PROCEEDINGS OF

ASMTA 2007

14TH INTERNATIONAL CONFERENCE

ON

ANALYTICAL AND STOCHASTIC MODELLING TECHNIQUES AND APPLICATIONS

4-6 June 2007
Prague, Czech Republic

Editors

Khalid Al-Begain
Armin Heindl
Miklos Telek

Co-sponsored by

IEEE UK and RI Computer Chapter

European Council on Modelling and Simulation

Society on Modelling and Simulation International

EC IST COST 290 Affiliated Conference

Proceedings of the 14th International Conference on Analytical and Stochastic Modelling Techniques and Applications

Editorial Enquiries
Professor Khalid Al-Begain
Chairman of the ECMS
Faculty of Advanced Technology,
University of Glamorgan
Pontypridd, Cardiff
CF37 1DL
United Kingdom
E-mail: kbegain@glam.ac.uk

In conjunction with
The 21st EUROPEAN CONFERENCE ON MODELLING AND SIMULATION
(ECMS 2007)

ISBN: **Hard Copy** **0-9553018-4-X**
978-0-9553018-4-1

CDROM **0-9553018-3-1**
978-0-9553018-3-4

(Set with Proceedings of ECMS 2007)

Although all papers published in these proceedings have gone through rigorous reviewing, responsibility of the accuracy of all statements rests solely with the author(s).

printed by **Digitaldruck Pirrot GmbH**
66125 Sbr.-Dudweiler, Germany

14th International Conference on Analytical and Stochastic Modelling Techniques and Applications

ADDITIONAL REVIEWERS

Andreas Binzenhoefer, University of Wuerzburg, Germany

Richard Boucherie, University of Twente, The Netherlands

Carla-Fabiana Chiasserini, School of Engineering of Turin, Italy

Michele Garetto, University of Turin, Italy

Marco Gribaudo, University of Turin, Italy

Andras Horvath, University of Turin, Italy

Tobias Hossfeld, University of Wuerzburg, Germany

Daniel Schlosser, University of Wuerzburg, Germany

Jon Schormans, Queen Mary University of London, UK

Yan Sun, Siemens/Nokia, China

J.I. van den Berg, TNO Telecom, The Netherlands

Preface

This is the 14th Analytical and Stochastic Modelling Techniques and Applications Conference. The conference that is becoming an important annual event in the fields of analytical modelling and performance evaluation in Europe.

Since last year many things have happened (good and sad). I start with the good news. This year has witnessed new blood joining the organisation of the conference. After the retirement of Gunter Bolch, who became the Honorary Chair of the conference, I invited Dr. Armin Heindl and Prof Miklos Telek to become the programme chairs for the conference. I now know that it was a very successful move. Both Armin and Miklos have done a wonderful job in promoting the conference in new circles resulting in doubling the number of submissions and in carrying through a very professional and rigorous reviewing and selection process. This resulted in the high quality programme of this year with an acceptance ratio just above 50%. The programme of ASMTA07 comprises 26 high quality papers organised into 7 sessions. Almost every paper was peer reviewed by three (in many cases by four) reviewers. The reviewers were truly wonderful this year, too, and in most of the cases the reviews provided valuable comments that contributed to increasing the quality of the final versions of the papers. In many cases, discussion panels were also organised where the reviews were not decisive.

I would like therefore to give a special thanks to both of them and also to all the members of the international programme committee for the excellent work in the reviewing process and the subsequent discussion panels during the selection process.

ASMTA has always been jointly organised with the European Conference on Modelling and Simulation the official conference of the European Council on Modelling and Simulation (also representing the SCS European Council). As a clear sign of the important role that ASMTA is playing in this setup, I have been elected as the Chairman for the Council in May 2006 during the ECMS06 conference in Bonn. I am very excited about this role and hope that I will be able to serve this wonderful community and raise the profile of the Council and the conferences.

Unfortunately, it is not all good news. ASMTA and the whole Queueing Theory community have suffered from the sudden death of Professor Pavel Bocharov. He has been one of the key figures in the field worldwide and a regular participant of ASMTA IPC and the conferences. We will definitely miss his questions and valuable comments during the sessions. This year's conference will be dedicated to his memory.

Let me finally wish you a very successful conference and also a pleasant stay in beautiful Prague. I am confident that the local organisers have made every effort to make it a memorable event. For that I give them my sincere thanks and appreciation.

Khalid Al-Begain
ASMTA07 Conference Chair
Chairman of the European Council for Modelling and Simulation
20th April 2007

Content

Session 4: Queueing systems and Networks

Session 5: Modelling and Analysis Techniques

Session 6: Network Models and Networking Issues

Session 7: Advanced Markovian Models

SCIENTIFIC PROGRAMME

SESSION 1

Traffic Modelling and Fitting Techniques

Explicit Phase-type Distribution Fitting in Laplace Transform Domain

Gábor Horváth
Department of Telecommunications
Budapest University of Technology and Economics
E-mail: ghorvath@hit.bme.hu

Abstract— **Several efficient and numerically stable methods are available to solve queueing systems where the random variables of the model (including the inter arrival and service times) are phase-type (PH) distributed. Thus, the approximation of measurement data by a PH distribution is an important task in Markovian performance modeling of practical systems.**

In this paper we introduce an explicit PH fitting method. This method performs the fitting in the Laplace transform domain in the sense that the Laplace transform of the pdf of the obtained PH distribution and the one of the target distribution are matched at the given points.

To evaluate its performance we applied this method on real traffic traces and found that – compared to some other available fitting methods – it provides good results by capturing both the body and the tail behaviour of the approximated distributions.

Keywords—**Distribution Fitting, Phase-Type Distribution**

I. Introduction

Matrix geometrical methods ([13]) are popular in the field of performance analysis due to their numerical tractability and flexible modeling capabilities. These methods provide efficient solutions for Markov chains having a regular structure. To model a practical system with a Markov chain and solve by a matrix geometric method the distributions of the random variables appearing in the system (e.g. inter-arrival times, packet sizes, etc.) have to be represented by phase-type (PH) distributions. Several methods have been published in the past to construct a PH distribution based on measurement data or based on a known but not Markovian distribution (e.g. Pareto, Weibull, etc.). The better the "quality" of the constructed PH is, the more accurate are the performance measures obtained from the Markov model of the system.

If we compare the published PH fitting methods, there is no clear winner, each method has its advantageous properties and its drawbacks. This motivates the research of new distribution fitting methods.

In this paper we introduce an explicit PH fitting approach. We show that PH fitting in Laplace transform domain can be done with a simple mathematical apparatus, and gives good results compared to other fitting methods.

The rest of the paper is organized as follows. Section II gives a brief summary on the PH distributions and its subclasses used by this paper. Section III overviews the PH fitting approaches published so far in the literature. Sections IV and V describe our fitting method in detail. The applicability of the method is demonstrated by numerical experiments in Section VI, finally Section VII concludes the paper.

II. Phase-Type Distributions

Phase-type (PH) distributions can be represented by an absorbing Markov chain. A PH distribution represents the distribution of the time the underlying Markov chain spends in the transient states till absorption, thus PH distributions are basically mixtures of the exponentially distributed sojourn times in the different states of the underlying Markov chain. PH distributions have two parameters: the generator of the transient part of the underlying Markov chain (denoted by $\boldsymbol{A}$) and the initial probability vector (denoted by α). The basic properties are the following:

- The probability density function (pdf) is:

$$f_{PH}(x) = \alpha\, e^{\boldsymbol{A}\,x}(-\boldsymbol{A})\,h.$$

(where h denotes a column vector of ones)

- The nth moment is the following:

$$m_n = n!\,\alpha\,(-\boldsymbol{A})^{-n}\,h. \tag{1}$$

- The Laplace transform of the pdf is:

$$f^*_{PH}(s) = \int_0^\infty f_{PH}(x)\,e^{-s\,x}dx = \alpha\,(s\boldsymbol{I} - \boldsymbol{A})^{-1}(-\boldsymbol{A})\,h.$$

Several subclasses of the PH distribution class have been introduced in the past for different purposes. Here we enumerate two of these subclasses that have been applied successfully in the past for distribution fitting.

A. Hyperexponential Distributions

HEDs (hyperexponential distributions) can be interpreted as a probabilistic choice between N exponential distributions (see Figure 1). With probability p_i the exponential distribution with parameter λ_i is chosen.

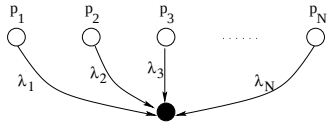


Fig. 1. Hyperexponential distribution

The advantage of this special PH structure is that it has a minimal number of parameters. (HEDs have

$2N-1$ parameters, while the full PH distribution class has N^2+N-1 parameters). Due to the special structure the pdf, its Laplace transform and the moments can be computed significantly simpler:

$$f_{HED}(x) = \sum_{i=1}^{N} p_i \, \lambda_i \, e^{-\lambda_i x}, \tag{2}$$

$$m_n = \sum_{i=1}^{N} p_i \, \frac{n!}{\lambda_i^n}, \tag{3}$$

$$f^*_{HED}(s) = \sum_{i=1}^{N} p_i \, \frac{\lambda_i}{s+\lambda_i}. \tag{4}$$

The simple formulas of these properties made HEDs popular among fitting methods in the past.

From the fitting problem point of view the disadvantages of HEDs are that the squared coefficient of variation is always larger than 1, and that the pdf is monotonically strictly decreasing. Thus, HEDs are not suited to approximate distributions which are more deterministic than the exponential or whose pdf is not monotonically decreasing.

B. Acyclic PH Distributions

PH distributions with acyclic transient generators are called APH (acyclic PH) distributions.

Cumani proved that every APH distribution can be transformed uniquely into a canonical form ([3]). This property makes it easy to check the equivalence of two APH distributions. These canonical forms have a minimal number of parameters ($2N-1$), which is advantageous because the fitting methods have to compute or to optimize less variables. Cumani introduced 3 canonical forms, in this paper we will refer to the first one shown by Figure 2.

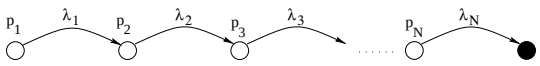


Fig. 2. Canonical form of APH distributions

Using the canonical form the Laplace transform of the pdf can be expressed by the following simple formula:

$$f^*_{APH}(s) = \sum_{i=1}^{N} p_i \prod_{j=i}^{N} \frac{\lambda_j}{s+\lambda_j}. \tag{5}$$

The APH distribution class is a wider class than the class of HEDs. Compared to HEDs, the bound of squared coefficient of variation is less strict (it is larger than $1/N$), and the pdf is not necessarily monotonically decreasing.

III. PH Fitting Methods

PH distribution fitting methods can be classified into two categories. Some of the fitting methods perform an optimization on the parameters of the PH, while the others compute these parameters directly by solving a system of (typically not linear) equations.

A. Optimization Based Fitting Methods

These methods start from an initial guess, and adjust the parameters of the PH iteratively to achieve the best possible fitting according to a subject function. For example, in the PHFIT tool ([7]) the subject function can be chosen by the user, the possibilities are: pdf area difference, cdf area difference and cross entropy. This tool operates on the APH distribution family.

The method in [9], [11], [10] matches the first three moments of the target distribution with a mixture of Erlang distributions, and performs an optimization on the remaining free parameters of the distribution according to a different subject function. This way this method is a mixture of the explicit and the optimization based algorithms.

An other family of optimization based methods are the EM-based fitting methods. The EM algorithm is an iterative method that aims to maximize the log-likelihood of the trace to fit. These algorithms start from an initial guess, and each iteration generates a new estimate such that the log-likelihood with the new estimate is larger than with the previous one. A number of EM-based fitting methods have been published for different subsets of the PH class. The output distribution is a HED in [12], it is a hyper-Erlang distribution in [16] and it is a general PH distribution in [1].

The drawbacks of these methods are that the evaluation of the subject function depends on the length of the trace, and that they might stick in a local maximum.

B. Explicit Fitting Methods

The difficulty in developing an explicit fitting method is that the equations to solve are not linear.

In [4] a HED fitting method is presented to approximate the tail behaviour of heavy-tailed distributions. This method solves eq. (2) at some points.

The most published methods regarding to explicit fitting are the moment matching methods. These methods solve the system of $m_i = \hat{m}_i$, where m_i is defined by eq. (1) and $\hat{m}_i$ are the moments of the distribution to approximate. A 2-state APH distribution is constructed based on 3 moments in [14]. Paper [2] presents an explicit method to construct the minimal order APH distribution given the first three moments. In [8] the parameters of an N-state APH distribution are calculated based on the first $2N-1$ moments.

In the upcoming sections we introduce an explicit fitting method that is based on the matching of the Laplace transforms of the pdf at several points, thus we solve the equations $f^*_{HED}(s_i) = f^*(s_i)$, where $f^*(s_i)$ corresponds to the distribution to fit. The result has a HED structure, but the method can be extended to provide an APH (or an even wider subclass of PH distributions) if necessary.

An other Laplace transform based fitting method has been published in [5] [1], where the first $2N-1$ derivatives of the transform ($2N-1$ terms of the power series

[1] Thanks for the anonymous reviewer for pointing this paper out

of $f^*(s)$) are matched. In that paper there is an other fitting method outlined (called Method 2), that is based on the matching of the Laplace transforms at several points, as we do in this paper. However – as this paper is independent on [5] – our method is different. In [5] the coefficients of the Laplace transform in rational fraction form are computed and an inverse transform is performed to obtain a generalized hyperexponential distribution (that is not necessarily Markovian), while our method computes the parameters of the approximating HED or APH distribution directly. (We further note that in [5] the focus is on the power series based method, thus the direct matching of Laplace transform has not been tested exhaustively).

Explicit fitting methods provide instant results, but also have some drawbacks. It is possible that the input combination can not be realized by a valid PH distribution. In this case, instead of providing the closest match, these methods simply fail. By increasing the number of phases (N), the chance of having an infeasible input combination increases. Thus, these methods work best with a small number of phases.

IV. HED Fitting in Laplace Transform Domain

In this section we present the details of a fitting method that works in the Laplace transform domain. The Laplace transform of the pdf of the distribution to fit is denoted by $f^*(s)$. We construct a hyperexponential distribution of order N such that the Laplace transform of its pdf (denoted by $f^*_{HED}(s)$) equals to $f^*(s)$ at the given points $s = s_1, s_2, \dots s_{2N}$. With other words, we determine the $2N$ parameters of the HED $\lambda_1, \dots, \lambda_N$ and $p_1, \dots, p_N$ such that the following $2N$ equations hold:

$$\sum_{n=1}^{N} p_n \frac{\lambda_n}{s_i + \lambda_n} = f^*(s_i), \quad \forall i = 1 \dots 2N, \qquad (6)$$

where $f^*(s_i)$ will be abbreviated by f_i in the sequel. To ensure that p_n is a distribution, thus $\sum p_n = 1$, s_1 is fixed to 0 and f_1 is fixed to 1.

Since the set of equations equations (6) are linear in p_n, we apply N reduction steps to obtain N equations that do not depend on p_n any more. A reduction step is basically a Gaussian elimination step. The kth reduction step is as follows:

1. for $\forall i = k \dots 2N$ multiply the ith equation
   by: $\begin{cases} \frac{s_i + \lambda_k}{s_i - s_{k-1}} & \text{if } k > 0 \\ s_i + \lambda_1 & \text{if } k = 1 \end{cases}$ (7)
2. Subtract the kth equation from
   equations $i = k+1, \dots, 2N$

Theorem 1: After the kth reduction step the left-hand sides of equations (6) are as follows:

$$\sum_{n=k+1}^{N} p_n \frac{s_i - s_k}{s_i + \lambda_n} \lambda_n \prod_{m=1}^{k} \frac{\lambda_n - \lambda_m}{\lambda_n + s_m}, \qquad (8)$$

the right-hand sides of equations (6) are as follows:

$$f_i \cdot (s_i + \lambda_k) \prod_{m=1}^{k-1} \frac{s_i + \lambda_m}{s_i - s_m} - \\ - \sum_{n=1}^{k} f_n \cdot (s_n + \lambda_n) \frac{s_i - s_k}{s_i - s_n} \prod_{m=1, m\neq n}^{k} \frac{s_n + \lambda_m}{s_n - s_m}, \qquad (9)$$

for $i = k+1, \dots, 2N$.

Proof: First we prove eq. (8) by induction. Assume eq. (8) holds for $k-1$. Now we apply the kth reduction step. Multiplying the ith equation by $\frac{s_i+\lambda_k}{s_i - s_{k-1}}$ gives:

$$\sum_{n=k}^{N} p_n \frac{s_i + \lambda_k}{s_i + \lambda_n} \lambda_n \prod_{m=1}^{k-1} \frac{\lambda_n - \lambda_m}{\lambda_n + s_m}.$$

By subtracting the kth equation from equations $i = k+1, \dots, 2N$ we get:

$$\sum_{n=k}^{N} p_n \underbrace{\left(\frac{s_i + \lambda_k}{s_i + \lambda_n} - \frac{s_k + \lambda_k}{s_k + \lambda_n} \right)}_{\frac{(s_i - s_k)(\lambda_n - \lambda_k)}{(s_i + \lambda_n)(s_k + \lambda_n)}} \lambda_n \prod_{m=1}^{k-1} \frac{\lambda_n - \lambda_m}{\lambda_n + s_m},$$

which equals eq. (8) since the term $n = k$ in the sum is zero.

Now we prove the second part of the theorem, again by induction. Assume eq. (9) holds for $k-1$. Now we apply the kth reduction step. After the multiplication and substraction we get:

$$f_i \cdot (s_i + \lambda_k) \prod_{m=1}^{k-1} \frac{s_i + \lambda_m}{s_i - s_m} - f_k \cdot (s_k + \lambda_k) \prod_{m=1}^{k-1} \frac{s_k + \lambda_m}{s_k - s_m} - \\ - \sum_{n=1}^{k-1} f_n \cdot (s_n + \lambda_n) \underbrace{\left(\frac{s_i + \lambda_k}{s_i - s_n} - \frac{s_k + \lambda_k}{s_k - s_n} \right)}_{\frac{(s_n + \lambda_k)(s_i - s_k)}{(s_i - s_n)(s_n - s_k)}} \prod_{m=1, m\neq n}^{k-1} \frac{s_n + \lambda_m}{s_n - s_m},$$

which – after some algebra – equals to eq. (9). ∎

Since the left-hand sides of equations $i = N+1, \dots, 2N$ equal to 0 after at the $k = N$th reduction step (see eq. (8)), we have N equations ($i = N+1, \dots, 2N$) that depend only on $\lambda_1, \dots, \lambda_N$:

$$0 = f_i \cdot (s_i + \lambda_N) \prod_{m=1}^{N-1} \frac{s_i + \lambda_m}{s_i - s_m} - \\ - \sum_{n=1}^{N} f_n \cdot (s_n + \lambda_n) \frac{s_i - s_N}{s_i - s_n} \prod_{m=1, m\neq n}^{N} \frac{s_n + \lambda_m}{s_n - s_m}. \qquad (10)$$

Now we introduce variables d_k, that are the following elementary symmetric polynomials of degree k in N variables:

$$\begin{aligned} d_0 &= 1, \\ d_1 &= \sum_{1 \le j \le N} \lambda_j, \\ d_2 &= \sum_{1 \le j < k \le N} \lambda_j \lambda_k, \\ d_3 &= \sum_{1 \le j < k < \ell \le N} \lambda_j \lambda_k \lambda_\ell, \end{aligned} \qquad (11)$$

and so on. By utilizing the properties of these symmetric polynomials we can translate the products to sums in eq. (10) as:

$$\prod_{m=1}^{N}(s_i+\lambda_m)=\sum_{j=0}^{N}s_i^{N-j}d_j. \tag{12}$$

With variables d_k the equations (10) become linear:

$$0=f_i\cdot a_i\sum_{j=0}^{N}s_i^{N-j}d_j-\sum_{n=1}^{N}f_n\cdot b_{i,n}\sum_{j=0}^{N}s_n^{N-j}d_j$$

$$\longrightarrow 0=\sum_{j=0}^{N}\underbrace{\left(f_i\cdot a_i\cdot s_i^{N-j}-\sum_{n=1}^{N}f_n\cdot b_{i,n}\cdot s_n^{N-j}\right)}_{c_{i,j}}d_j,$$

where we introduced supplementary variables a_i and $b_{i,n}$ to make the formulas shorter:

$$a_i=\prod_{m=1}^{N-1}\frac{1}{s_i-s_m}, \tag{13}$$

$$b_{i,n}=\frac{s_i-s_N}{s_i-s_n}\prod_{m=1,m\neq n}^{N}\frac{1}{s_n-s_m}. \tag{14}$$

Thus we have the following set of linear equations whose solution provides the d_k variables:

$$-c_{i,0}=\sum_{j=1}^{N}c_{i,j}\cdot d_j,\qquad i=N+1,\ldots,2N. \tag{15}$$

From the properties of elementary symmetric polynomials the λ_n intensities of the HED are the roots of the following polynomial:

$$P(\lambda)=\lambda^N-d_1\lambda^{N-1}+d_2\lambda^{N-2}-\cdots\pm d_N. \tag{16}$$

The roots of this polynomial can be computed symbolically using the Cardano formulas up to $N=4$, above that a numerical method has to be applied.

Having the λ_n intensities computed, the p_n probabilities can be easily calculated by a linear set of equations given by eq. (6) at $i=1,\ldots,N$. The following linear set of equations gives the p_n probabilities:

$$\begin{bmatrix}\frac{\lambda_1}{s_1+\lambda_1} & \cdots & \frac{\lambda_N}{s_1+\lambda_N}\\ \frac{\lambda_1}{s_2+\lambda_1} & \cdots & \frac{\lambda_N}{s_2+\lambda_N}\\ \vdots & \ddots & \vdots\\ \frac{\lambda_1}{s_N+\lambda_1} & \cdots & \frac{\lambda_N}{s_N+\lambda_N}\end{bmatrix}\begin{bmatrix}p_1\\p_2\\\vdots\\p_N\end{bmatrix}=\begin{bmatrix}f_1\\f_2\\\vdots\\f_N\end{bmatrix}. \tag{17}$$

The presented method is summarized in the following 3 steps:

1. First we compute the coefficients of the linear system of equations given by eq. (15) and solve it to obtain the d_k variables.
2. In the next step we obtain the λ_n intensities of the HED from variables d_k by computing the roots of polynomial eq.(16) symbolically or numerically.
3. In the last step we compute the p_n probabilities by solving the linear system given by eq. (17).

V. Obtaining a Markovian representation

In most of the cases the constructed PH distribution is plugged into a Markovian model to replace a distribution that is not Markovian or that is known only empirically. Therefore the result of a fitting method should have a Markovian representation, thus a valid PH distribution with positive intensities and valid initial probabilities.

If the algorithm presented by the previous section gives a valid HED, thus $p_i\in(0,1),\ \lambda_i>0, i=1\ldots N$, then we obtained the Markovian representation directly, in a hyper-exponential structure.

Otherwise we have to look for the Markovian representation in a wider class of distributions, first in the class of APH distributions. If the λ intensities are all valid (real and non-negative), but some of the probabilities are negative, we can try to transform the invalid HED to the canonical form of the APH class. With the transformation to APH the invalid probabilities may become valid, by resulting a valid APH distribution. During the transformation the λ intensities of the APH distribution remain the same as the ones of the original HED. This means that instead of performing the canonical transformation we simply have to keep the λ parameters and to replace eq. (17) with the following one to obtain the initial probabilities of the APH according to eq. (5):

$$\begin{bmatrix}\frac{\lambda_N}{s_1+\lambda_N} & \frac{\lambda_{N-1}}{s_1+\lambda_{N-1}}\frac{\lambda_N}{s_1+\lambda_N} & \cdots\\ \frac{\lambda_N}{s_2+\lambda_N} & \frac{\lambda_{N-1}}{s_2+\lambda_{N-1}}\frac{\lambda_N}{s_2+\lambda_N} & \cdots\\ \vdots & \ddots & \vdots\\ \frac{\lambda_N}{s_N+\lambda_N} & \frac{\lambda_{N-1}}{s_N+\lambda_{N-1}}\frac{\lambda_N}{s_N+\lambda_N} & \cdots\end{bmatrix}\begin{bmatrix}p_N\\\vdots\\p_2\\p_1\end{bmatrix}=\begin{bmatrix}f_1\\f_2\\\vdots\\f_N\end{bmatrix}. \tag{18}$$

If there is no valid APH representation either, there is still some hope: one can try to apply the results of [6], where the author computes an unicyclic representation from the eigenvalues (which we already computed: $\lambda_1,\ldots,\lambda_N$), that is a wider class than the APH class.

If all the symbolical methods fail to provide a Markovian representation, a heuristic iterative method published in [15] can be applied, that looks for a general PH representation of a ME distribution.

VI. Numerical Experiments

The fitting method is tested with two well-known traces that contain real measurement data and are available to download from the internet. The Laplace transform of the pdf of the trace at point s can be computed as follows:

$$f^*(s)=\frac{1}{L}\sum_{\ell=1}^{L}e^{-s\,x_\ell},$$

where L is the number of samples in the trace, and x_ℓ is the ℓth sample in the trace.

A. The lbl-tcp-3 trace

The lbl-tcp-3 trace contains the inter arrival times of packets of two hours of wide-area TCP traffic between

the Lawrence Berkeley Laboratory and the rest of the world.

We first fit the trace with different number of states. After a short experimentation we found a valid HED with $s_1, \ldots, s_N = 0, 1, 3, 5, 7, 9, 11, 13$. The experience was that – based on the visual comparison of the pdfs in linear and log-log scale – the HED with 2 states provided a visibly worse fit, while the ones with 3 and 4 states provided indistinguishable results (we omitted to include the plots in the paper). So we decided to use a 3-state HED in the sequel.

A crucial question is how to select the s_i points at which the Laplace transform of the HED and the one of the trace are matched. According to the limit theorems of the Laplace transform, the final value of a function is $f(\infty) = \lim_{s\to 0} sf^*(s)$, and the initial value can be obtained as $f(0+) = \lim_{s\to\infty} sf^*(s)$. Consequently if the s_i values are selected such that $sf^*(s)$ is small, then the tail, if they are such that $sf^*(s)$ is large, the body of the distribution will be fitted more accurately. We studied the effect of s_i values with the following three settings:

small		large		combined	
s_i	$s_i f^*(s_i)$	s_i	$s_i f^*(s_i)$	s_i	$s_i f^*(s_i)$
1	0.99	150	97.9	1	0.99
3	2.96	300	150.6	3	2.96
5	4.89	450	185.5	5	4.89
7	6.80	600	210.9	600	210.9
9	8.68	750	230.3	750	230.3

Instead of selecting s based on $sf^*(s)$, we selected the s values based on $f^*(s)$ only, because it is simpler. We took some s values by which $f^*(s)$ is around ≈ 0.9, and some others by which s is $\approx 0.4 - 0.2$. (Our fitting method provided a HED representation with all three cases).

The results are depicted by Figure 3. The plots confirm our arguments. The tail behaviour of the distribution can be captured by selecting small s_i values (see the log-log-scale plot), the body of the pdf can be captured better by selecting larger s_i values. According to the plots, the combined solution seems to be a good choice for fitting both the body and the tail of the pdf.

We compared the results of the algorithm to some other general purpose PH fitting tools. For the sake of fairness, these tools have been configured to fit with 3 states, too. The involved tools were GFIT, which is an EM-based iterative method (see [16]); MOMFIT, that performs symbolical moment matching (see [8]); and LTFIT, our presented method. The compared quantities are the mean value, the squared coefficient of variation (c_v^2) and the log-likelihood. The following table summarizes the results:

	Mean	c_v^2	Log-likelihood
trace	0.0041981	1.9923	-
MOMFIT	0.0041981	1.9923	4531094
GFIT	0.0041981	1.963	4531284
LTFIT	0.0041981	1.9922	4531861

As the table shows, our method performs well when

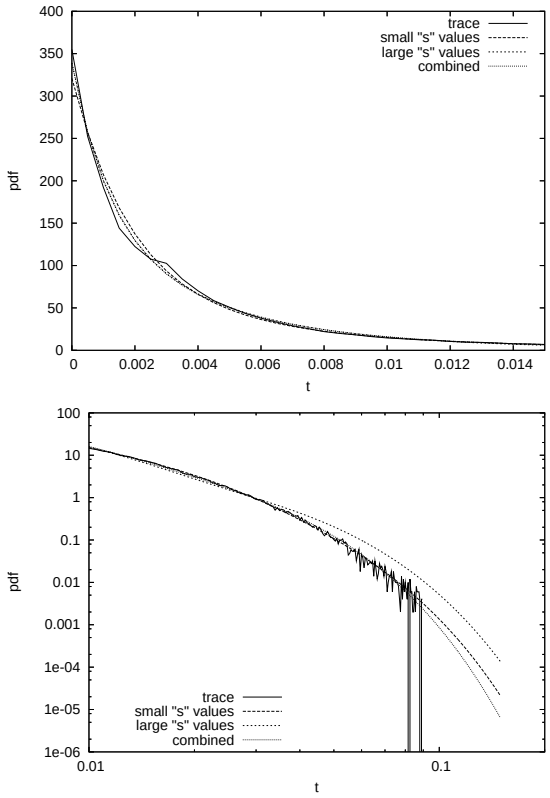


Fig. 3. Pdfs with different s_i points, linear and log-log scale

comparing these quantities. The most interesting result is that LTFIT gave the largest log-likelihood value, while the log-likelihood is the object of maximization in GFIT. The pdfs provided by these tools are available on Figure 4 for visual comparison.

B. The NASA trace

This trace contains around 1.9 million samples that are the number of bytes in the reply of HTTP requests to the NASA Kennedy Space Center WWW server in Florida.

A short experimentation was needed to find the proper s values. Our concept was – as in the previous example – to select s values by which $f^*(s)$ is large (around 0.9) to fit the body and also some s values by which $f^*(s)$ is smaller (like $0.2 - 0.4$) to fit the tail of the pdf. After some unsuccessful tries, with $s = 0, 0.0000001, 0.0000005, 0.000001, 0.0005, 0.001$ the fitting method provided a valid HED.

The following table compares the results to MOMFIT. Unfortunately GFIT did not converge for this trace (and gave "nan" results):

	Mean	c_v^2	Log-likelihood
trace	20455	14.155	-
MOMFIT	20455	14.155	$-1.9741e+07$
LTFIT	20455	14.151	$-1.9247e+07$

The log-likelihood is larger again with LTFIT, and the accuracy of the square coefficient of variation is sufficiently good as well. The pdfs are depicted by Figure 5.

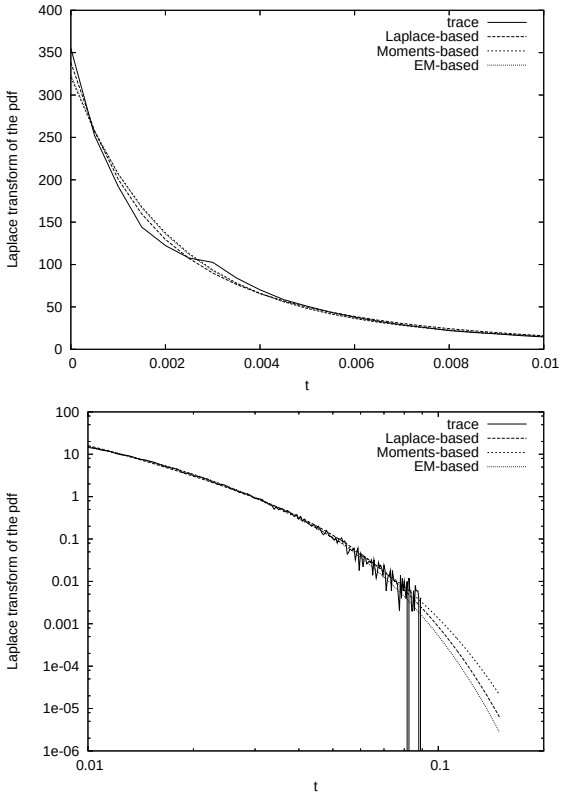


Fig. 4. Different fitting methods compared, linear and log-log scale

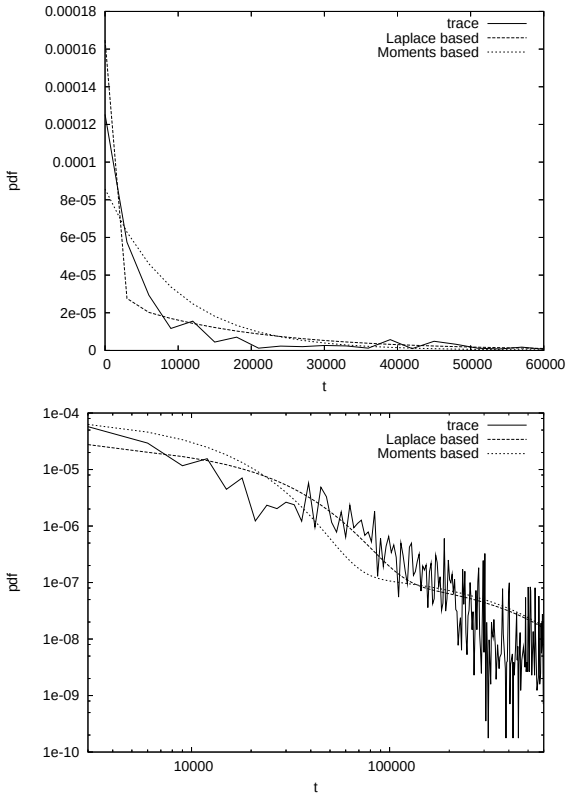


Fig. 5. Different fitting methods compared, linear and log-log scale

VII. Conclusion

In this paper we described a new PH fitting approach. We provided explicit formulas for HED fitting in Laplace transform domain, and showed how to extend the method for more general classes of PH distributions. Two numerical examples demonstrated that this approach performs well compared to the existing PH fitting methods.

However, there are still some open questions to solve. Future research has to be done to find an automatic method to select the proper s points at which the Laplace transforms are matched. Since the fitting method may fail at the selected s points (resulting a non-Markovian representation), developing a heuristic algorithm to move these points in order to find the closest feasible match is also a subject of future work.

References

[1] S. Asmussen, O. Nerman, and M. Olsson. Fitting Phase-type distributions via the EM algorithm. *Scand. J. Statist.*, 23(4):419–441, 1996.

[2] A. Bobbio, A. Horváth, and M. Telek. Matching three moments with minimal acyclic phase type distributions. *Stochastic Models*, 21:303–326, 2005.

[3] A. Cumani. On the canonical representation of homogeneous Markov processes modelling failure-time distributions. *Microelectronics and Reliability*, 22:583–602, 1982.

[4] A. Feldman and W. Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. *Performance Evaluation*, 31:245–279, 1998.

[5] C. M. Harris and W. G. Marchal. Distribution estimation using Laplace transforms. *Informs J. Computing*, 10(4):448–458, 1998.

[6] Qi-Ming He and Hanqin Zhang. A note on unicyclic representation of PH-distributions. *Stochastic Models*, 21:465–483, 2005.

[7] A. Horváth and M. Telek. PhFit: A general purpose phase type fitting tool. In *Tools 2002*, pages 82–91, London, England, April 2002. Springer, LNCS 2324.

[8] A. Horváth and M. Telek. Matching more than three moments with acyclic phase type distributions. *submitted for publication*, 2005.

[9] M. A. Johnson and M. R. Taaffe. Matching moments to phase type distributions: mixtures of Erlang distributions of common order. *Stochastic Models*, 5:711–743, 1989.

[10] M. A. Johnson and M. R. Taaffe. Matching moments to phase type distributions: density function shapes. *Stochastic Models*, 6:283–306, 1990.

[11] M. A. Johnson and M. R. Taaffe. Matching moments to phase type distributions: nonlinear programming approaches. *Stochastic Models*, 6:259–281, 1990.

[12] Rachid El Abdouni Khayari, Ramin Sadre, and Boudewijn R. Haverkort. Fitting world-wide web request traces with the EM-algorithm. *Performance Evaluation*, 52(2-3):175–191, 2003.

[13] G. Latouche and V. Ramaswami. *Introduction to matrix analytic methods in stochastic modeling*. SIAM, 1999.

[14] M. Telek and A. Heindl. Matching moments for acyclic discrete and continuous phase-type distributions of second order. *International Journal of Simulation Systems, Science & Technology*, 3(3-4):47–57, dec. 2002. Special Issue on: Analytical & Stochastic Modelling Techniques.

[15] M. Telek and G. Horváth. A minimal representation of markov arrival processes and a moments matching method. *submitted for publication*, 2007.

[16] A. Thümmler, P. Buchholz, and M. Telek. A novel approach for fitting probability distributions to trace data with the EM algorithm. *IEEE Trans. on Dependable and Secure Computing*, 3(3):245–258, 2006.

A Hybrid Algorithm for Parameter Fitting of Markovian Arrival Processes

Andriy Panchenko and Peter Buchholz
Computer Science Department, University of Dortmund
D-44221 Dortmund, Germany
{andriy.panchenko,peter.buchholz}@uni-dortmund.de

Keywords—**Stochastic Models, Markov Models, Queueing Systems and Network Models, Workload Modeling and Characterization**

Abstract—**Markovian arrival processes (MAPs) allow the modeling of general traffic processes and can be used to describe arrival or service processes in analytically solvable models. For the practical use of MAPs, the major problem is an appropriate parametrization of the MAP according to measured data traces. Parameter fitting of MAPs results in a complex nonlinear optimization problem and available fitting approaches either provide rough approximations with MAPs of order two or three, or yield large state spaces with dimensions over 50. The challenge is to build compact MAPs with up to 10 states that capture realistic traffic behavior. In this paper we present a new fitting approach which combines the established expectation maximization algorithm to fit the probability density function and an evolutionary algorithm for autocorrelation fitting. The new algorithm essentially improves the adoption of autocorrelation properties in comparison to pure EM algorithms.**

I. Introduction

Markovian arrival processes (MAPs) are used for some time to describe complex traffic processes in models of various systems like computer, communication or manufacturing systems [Horváth and Telek, 2002]. In practice one often has a sample of data resulting from measurements and the parameters of a MAP have to be chosen to capture the essential aspects of the data. Many important properties of observed data are based on its density function and autocorrelation behavior. The adequate model has to approximate these properties in a way that the pdf and autocorrelation functions of the MAP and of the observed data are similar. Unfortunately, the resulting optimization problem is of a high dimension, since the number of free parameters is $2n^2 - n$ for a MAP with n states and the problem is nonlinear. An important subclass of MAP is a phase type (PH) renewal process that describes only the distribution without correlation dependencies. A phase type (PH)-distribution used in PH renewal process contains $n^2 + n - 1$ free parameters which are often fitted using algorithms of the expectation maximization (EM) type [Asmussen et al., 1996, Buchholz, 2003] or by heuristic non-linear optimization methods [Horvath and Telek, 2002]. However, even for PH distributions a straightforward application of the approaches results in long fitting times. An analytical investigation of higher order MAPs seems to be unrealistic since even the simple two state case yields complex dependencies between the different measures as shown in [Heindl et al., 2006]. Thus, to fit parameters of MAPs no efficient and robust method is available yet. Of course, standard EM algorithms can be and have been extended to fit parameters of MAPs [Buchholz, 2003, Klemm et al., 2003], but the result is usually not satisfactory since fitting times are enormous and results are sensitive to the initial values of the parameters, if the order of the MAP goes beyond 2 or 3.

In this paper we present a hybrid approach which combines a standard EM approach to fit the probability density function of a PH distribution, extends the PH distribution into a MAP and uses afterwards an evolutionary algorithm to fit the coefficients of correlation. Since the autocorrelation fitting modifies the probability density function, the whole approach is iterated until a satisfactory parameter set has been generated.

The structure of the paper is as follows. In the next section we introduce MAPs and their basic properties. Then, in section 3, the hybrid fitting approach is presented. The quality of the approach is afterwards shown by means of some examples using real data traces from the Internet traffic archive [tra].

II. Markovian Arrival Processes

The following section gives a brief overview of PH distributions and MAPs, it shows how basic quantities like values of the probability density function (pdf) or lag k coefficients of correlation can be computed for a MAP and it defines the problem of parameter fitting. For further details about MAPs we refer to the literature [Fischer and Meier-Hellstern, 1993, Horváth and Telek, 2002].

A. Basic Definitions

A MAP introduced and investigated by Neuts[Neuts, 1981] is Markov process with state space $\mathcal{S} = \{1, \ldots, n\}$ and an irreducible generator matrix $\mathbf{Q}$. For a MAP description $\mathbf{Q}$ can be represented as $\mathbf{D_0} + \mathbf{D_1}$ where $\mathbf{D_1}$ is a nonnegative matrix, $\mathbf{D_0}$ has nonnegative elements outside the diagonal and $-\mathbf{D_0}(i,i) = \sum_{j:j\neq i} \mathbf{D_0}(i,j) + \sum_j \mathbf{D_1}(i,j)$. The behavior of a MAP is as follows: The process stays in state i for an exponentially distributed time with rate $-\mathbf{D_0}(i,i)$. Afterwards either a transition from $\mathbf{D_0}$ or a transition from $\mathbf{D_1}$ occurs. Transitions from $\mathbf{D_0}$ are internal and change the state of the MAP, i.e. with probability $\mathbf{D_0}(i,j)/-\mathbf{D_0}(i,i)$ the state changes to j. Transitions from $\mathbf{D_1}$ are external, which means that they generate an arrival into some environment. Thus, every time a transition from $\mathbf{D_1}$ occurs, a new arrival is generated and the sequence of arrivals forms the stochastic process describing the interarrival times.

Since $\mathbf{Q}$ is the generator matrix of an irreducible Markov process, its stationary distribution exists and can be computed as the solution of $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$ normalized to 1. The stationary distribution immediately after an arrival is given by $\boldsymbol{\tau} = \boldsymbol{\pi}\mathbf{D_1}/\boldsymbol{\pi}\mathbf{D_1}\mathbf{e}^T$, where $\mathbf{e}$ is a row vectors of ones.

MAPs can describe correlated arrival streams. If correlation is not needed, one can use PH-distributions which are characterized by matrix $\mathbf{D_0}$, arrival column vector $\boldsymbol{d}_1 = -\mathbf{D_0}\mathbf{e}^T$ and a row vector $\boldsymbol{\tau}$ for the initial distribution. In contrast to MAPs, a PH distribution starts after an arrival always with probability $\boldsymbol{\tau}(i)$ in state i and arrival rates are defined in vector $\boldsymbol{d}_1$. A PH distribution can be interpreted as a MAP with matrix $\mathbf{D_1} = \boldsymbol{d}_1\boldsymbol{\tau}$.

B. Analysis of MAPs

The main statistical properties of a MAP can be directly computed from the the matrices $\mathbf{D_0}$ and $\mathbf{D_1}$ [Horváth and Telek, 2002, Heindl et al., 2006]. The value of the probability density function (pdf) at time $t(> 0)$ is given by

$$f(t) = \boldsymbol{\tau} e^{\mathbf{D_0}t}\mathbf{D_1}\mathbf{e}^T \tag{1}$$

and the value of the distribution function can be computed from $F(t) = 1-\boldsymbol{\tau} e^{\mathbf{D_0}t}\mathbf{e}^T$. Since MAPs describe correlated streams, measures to capture the correlation structure are of major importance. The common measure is the lag k coefficient of autocorrelation which is defined as

$$r(k) = \frac{E\left[T^{(1)}\right]\boldsymbol{\tau}\left(\left(-\mathbf{D_0}^{-1}\mathbf{D_1}\right)^k \boldsymbol{m}^{(1)} - \mathbf{e}^T E\left[T^{(1)}\right]\right)}{E\left[T^{(2)}\right] - \left(E\left[T^{(1)}\right]\right)^2}, \tag{2}$$

where $\boldsymbol{m}^{(l)}$ is the vector of conditional l-th moments.

C. The Parameter Fitting Problem for MAPs

The fitting problem of MAPs describes the finding of parameters of a MAP such that the MAP generates some interarrival sequence which usually results from some observation of real processes. Let $\boldsymbol{t} = (t_1, ..., t_m)$ be some observed sequence of interarrival times which is denoted as a trace. A MAP should be generated such that its statistical properties approximate the properties of the observed trace. Two general possibilities of mapping exist. First one can try to fit the empirical moments and some autocorrelation lags of the trace by the MAP. However, this approach becomes very complex and is currently only possible for MAPs of order two [Heindl et al., 2006]. Alternatively, one can consider the likelihood $\mathcal{L}(\langle \mathbf{D_0}, \mathbf{D_1}\rangle, \boldsymbol{t})$ that the trace is generated from the MAP which is defined as

$$\mathcal{L}(\langle \mathbf{D_0}, \mathbf{D_1}\rangle, \boldsymbol{t}) = \boldsymbol{\tau}\left(\prod_{i=1}^{m} e^{\mathbf{D_0}t_i}\mathbf{D_1}\right)\mathbf{e}^T. \tag{3}$$

The likelihood yields the maximization problem $\max_{\mathbf{D_0},\mathbf{D_1}}$ $(\mathcal{L}(\langle \mathbf{D_0}, \mathbf{D_1}\rangle, \boldsymbol{t}))$ to maximize the pdf for all values from trace $\boldsymbol{t}$ considering the sequence of events.

For PH distributions the optimization problem is less complex since instead of matrices $\langle \mathbf{D_0}, \mathbf{D_1}\rangle$ only vector $\boldsymbol{\tau}$ and matrix $\mathbf{D_0}$ have to be found. However, even in this case only for restricted classes of matrices $\mathbf{D_0}$, like upper triangular matrices, standard optimization methods for nonlinear problems can be applied. For the general case algorithms of the expectation maximization type are used [Asmussen et al., 1996]. These algorithms are iterative and increase in each iteration the likelihood value such that they finally find some local optimum. In principle, the approach can also be applied for MAPs [Buchholz, 2003, Klemm et al., 2003], but the problem is the huge effort of the approaches. In each iteration, the likelihood value has to be computed using (3) and often the algorithm requires an enormous number of iterations until convergence is reached. The effort per iteration grows with the number of elements in the matrices $\mathbf{D_0}$ and $\mathbf{D_1}$ and with the number of elements in the trace. Traces of realistic traffic processes, as they can be found in the Internet archive [tra], contain several millions or even billions of values, but EM type algorithms can be applied for $10^4 - 10^5$ elements at most.

One way to reduce the effort is to consider only a subtrace, but as shown in [Buchholz, 2003] this usually yields bad results. The alternative is to fit the MAP according to some condensed information about the trace, like the moments, values of the pdf or the lag k coefficients of correlation. In [Buchholz and Panchenko, 2004b] a two-step EM algorithm is proposed that first fits a PH-distribution according to the empirical pdf of the trace and afterwards transforms the PH-distribution into a MAP which is modified to capture autocorrelations. Such a separation of fitting problems essentially reduces efforts in distribution fitting due to an usage of approximations of the empirical pdf at some points. The EM algorithm [Buchholz and Panchenko, 2004a] exactly fits the first moment and provides an appropriate quality in fitting higher moments and in approximating the tail of the pdf shape. The autocorrelation fitting time is also reduced because of caching in intermediate steps, but nevertheless remains still dependent of the trace length. Thus, the problems of long fitting times remains, even if the effort is reduced compared to a one step EM approach.

III. A Hybrid Approach for Parameter Fitting

As shown in the previous section, EM algorithms yield satisfactory results for pdf fitting, but require too much effort if the autocorrelation function has to be fitted. The reason for the long fitting time lies in the dependence of the efforts per autocorrelation fitting iteration on the number of elements in the trace. To overcome this problem and to reduce the fitting effort we use an objective function which computes the distance between empirical and modeled autocorrelation coefficients up to lag K. The advantage of such a distance computation is its independence of the trace length that allows one to perform fitting more efficiently. To adjust the MAP matrices we apply an evolutionary approach to fit the autocorrelation and combine this approach with the EM algorithm for pdf fitting. Below we briefly present a general framework of evolutionary algorithms and present the EA for autocorrelation fitting. Then we introduce multiobjective evolutionary algorithms that take into account both, the quality of distribution and correlation fitting. Finally we integrate the EA with EM into a hybrid algorithm and evaluate it by fitting of real data traces.

A. Evolutionary Algorithms for MAP Fitting

The general optimization problem can be defined as $\min y = f(\langle \mathbf{D_0}, \mathbf{D_1}\rangle, \boldsymbol{t})$ under the constraints $\mathbf{D_1}(i,j) \geq 0$ and $\mathbf{D_0}(k,l) \geq 0$ for $k \neq l$. The diagonal elements of $\mathbf{D_0}$ are implicitly given by the nondiagonal elements in $\mathbf{D_0}$ and

Algorithm 1 Framework of an evolutionary algorithm

$s = 0$
Initialize $\mathcal{P}_s$
Evaluate the elements in $\mathcal{P}_s$ according to function f
REPEAT
  Select μ individuals from $\mathcal{P}_s$ as parents
  Generate λ offspring from the parents and put them into new population $\mathcal{P}_{s+1}$
  Evaluate the new individuals according to function f and add the parents to $\mathcal{P}_{s+1}$
  $s := s + 1$
UNTIL not terminated

the elements in $\mathbf{D_1}$. For an evolutionary optimization approach let $\mathcal{P}_s = \left\{ \left\langle \mathbf{D_0}^{(1)}, \mathbf{D_1}^{(1)} \right\rangle, \ldots, \left\langle \mathbf{D_0}^{(\mu)}, \mathbf{D_1}^{(\mu)} \right\rangle \right\}$ be the MAP population in generation s. Using well known concepts from evolutionary optimization methods [Sarker et al., 2002], an optimization algorithm for MAP fitting according to function $f(.)$ can be defined as Algorithm 1.

The initialization of $\mathcal{P}_0$ can be done by randomly generating μ individuals which observe the above conditions for valid MAPs. Evaluation of f depends on the chosen function that describes the quality of MAPs. As parents the μ individuals from $\mathcal{P}_s$ with the best fitness, i.e. the smallest values $f(.)$, are chosen. The λ new offspring are generated by variation of individuals from the parent population. Individuals to be modified are chosen randomly. Afterwards one of the following variation operations is applied to the matrix $\mathbf{D_1}$.

The variation operators for MAP are constructed in such a way, that from a feasible individual another feasible individual is generated, i.e. after completion of variation operations each new individual satisfies the constraints (see subsection II-A). The construction of feasible individuals reduces the number of required constraint controls and individual correction operations. The following variation operations are conducted on the elements of a single matrix row or on the complete matrix $\mathbf{D_1}$. 1. By *row cyclic shift mutation* the elements in one randomly chosen row are cyclically shifted $l = random(1,\, n-1)$ times. 2. The *row element swap mutation* swaps two randomly chosen elements from a randomly chosen row. 3. By *row uniform mutation* two randomly chosen elements i and j from a randomly chosen row k and a value $w = uniform(0, 1)$ are chosen and the new elements in matrix $\mathbf{X}$ are given by $\mathbf{X}^{new}(k, i) = \mathbf{X}(k, i) + w \cdot \mathbf{X}(k, j)$ and $\mathbf{X}^{new}(k, j) = (1 - w) \cdot \mathbf{X}(k, j)$. 4. The *row element zero mutation* uses the previous operation with $w = 1$. 5. By *matrix row cyclic shift mutation* the rows are cyclically shifted $l = random(1,\, n-1)$ times. 6. The *matrix row swap mutation* swaps two randomly chosen rows. 7. The *matrix row swap crossover* generates new individuals from two matrices $\mathbf{X}^{(\alpha)}$ and $\mathbf{X}^{(\beta)}$ $(\alpha, \beta \in \{1, \ldots, \mu\})$ by swapping i-th rows $\mathbf{X}^{(\alpha)}(i\bullet)$ and $\mathbf{X}^{(\beta)}(i\bullet)$. Apart from the last one, all operations choose a single parent and generate one offspring. For the last operation two parents are chosen to generate two offsprings. Thus, first the operation is chosen, then one individual from the parent population is chosen or if the last operation is used two individuals are chosen, then the new individual is generated by applying the operation. Observe that all operations leave the number of states of the MAP unchanged such that the number of states becomes an input parameter of the method and a good MAP with a given number of states has to be found. This is the usual way of performing parameter fitting of MAPs or phase type distributions.

B. Evolutionary Autocorrelation Fitting

As shown in [Buchholz and Panchenko, 2004b] the separate fitting of pdf and autocorrelation structure of a MAP can be an alternative to a single step approach as used in [Buchholz, 2003, Klemm et al., 2003]. In the mentioned approach, an EM algorithm is used to optimize the likelihood function which results in long fitting times. Here we extend the approach by using an evolutionary algorithm for fitting the MAP parameters according to the lag k coefficient of correlation, whereas for the distribution fitting the EM algorithm from [Buchholz and Panchenko, 2004a] is applied.

Let $\langle \mathbf{D_0}, \mathbf{D_1} \rangle$ be the MAP resulting from pdf fitting. For fitting the parameters to the lag k coefficients of correlation only matrix $\mathbf{D_1}$ is modified, $\mathbf{D_0}$ keeps unchanged which implies also that the row sum of $\mathbf{D_1}$ is left unchanged. Thus, the variation operations introduced above are then applied to matrix $\mathbf{D_1}$ yielding a new matrix $\mathbf{D_1}^{(new)}$.

The presence of a single objective function essentially simplifies the selection of best individuals and reduces the complexity of the complete algorithm. As fitness criteria $f()$ we tried to apply simple objective functions that depict the autocorrelation fitting quality. The objective function is built as the weighted sum of the difference between the lag k coefficients of correlation of the trace and the MAP.

$$f_a(\langle \mathbf{D_0}, \mathbf{D_1} \rangle, \boldsymbol{t}) = \frac{1}{K} \sum_{k=1}^{K} w_k \left| r_{(\langle \mathbf{D_0}, \mathbf{D_1} \rangle)}(k) - \boldsymbol{r}_{(\boldsymbol{t})}(k) \right|, \tag{4}$$

where w_k is the weight of lag k, $r_{(\langle \mathbf{D_0}, \mathbf{D_1} \rangle)}(k)$ is the lag k coefficient of correlation of the MAP with matrices $\langle \mathbf{D_0}, \mathbf{D_1} \rangle$ and $\boldsymbol{r}_{(\boldsymbol{t})}(k)$ is the lag k coefficient of correlation of the trace. Observe that a modification of $\mathbf{D_1}$ implies that $\boldsymbol{\tau}$ is modified and therefore also the pdf is modified. This has the effect that the resulting MAP might approximate the autocorrelation well, but does not approximate the pdf.

Evaluation of the algorithm with the fitness function (4) provided satisfactory results in fitting of autocorrelation dependencies, but the price of this was a weak pdf approximation. Among the fitting results for various data traces we observed the following tendency: the better the autocorrelation dependencies are fitted the more coarse and inaccurate is the approximation of the pdf. This fact induced us to consider MAP fitting as a multiobjective problem where distribution and autocorrelation fitting quality have to be considered simultaneously. The hybrid approach presented in the next subsection uses a multiobjective goal function and is clearly superior to the algorithm with a single fitness function.

C. Multiobjective Evolutionary Algorithms

In multiobjective optimization problems (MOP) instead of a scalar value, a vector of fitness values is used. Thus,

the fitness function becomes $\boldsymbol{f} = (f_1, f_2, \ldots, f_R)$. Let $\boldsymbol{y}^{(i)} = \boldsymbol{f}(\left\langle \mathbf{D_0}^{(i)}, \mathbf{D_1}^{(i)} \right\rangle)$ be the vector of objective function values for $i-$th individuals. Between two elements $\boldsymbol{y}^{(1)}$ and $\boldsymbol{y}^{(2)}$ we might have the following relations:

1. $\boldsymbol{y}^{(1)} \succ \boldsymbol{y}^{(2)}$ ($\boldsymbol{y}^{(1)}$ dominates $\boldsymbol{y}^{(2)}$) if no component of $\boldsymbol{y}^1$ is smaller than the corresponding component of $\boldsymbol{y}^2$ and at least one component is greater,
2. $\boldsymbol{y}^{(1)} \succeq \boldsymbol{y}^{(2)}$ ($\boldsymbol{y}^{(1)}$ weakly dominates $\boldsymbol{y}^{(2)}$) if $\boldsymbol{y}^{(1)}(r) \geq \boldsymbol{y}^{(2)}(r)$ for all $r = 1, \ldots, R$, or
3. $\boldsymbol{y}^{(1)} \sim \boldsymbol{y}^{(2)}$ ($\boldsymbol{y}^{(1)}$ is indifferent to $\boldsymbol{y}^{(2)}$) if $\left(\boldsymbol{y}^{(1)} \not\succeq \boldsymbol{y}^{(2)}\right) \wedge \left(\boldsymbol{y}^{(2)} \not\succeq \boldsymbol{y}^{(1)}\right)$ holds.

For a set of solutions $\mathcal{Y}$, the Pareto front is defined as the subset of all elements from $\mathcal{Y}$ that are not dominated by another element from the set.

For MAP fitting, the different functions $f_.()$ define the values of the multidimensional fitness function. Usually it is sufficient to use two dimensional values resulting from $f_p()$ and $f_a()$ defined as (3) and (4) respectively. The solution of multiobjective problems is computed by an approach proposed in [Zitzler and Thiele, 1998] which is denoted as MOEA Strength Pareto Evolutionary Algorithm (SPEA) and implements an approach to optimization (minimization) of MOP by a mixture of existing and new techniques in order to approximate the Pareto-optimal set. As a framework for the fitting algorithm we take the original SPEA algorithm and its improved version SPEA2 [Zitzler et al., 2001].

The algorithm implements a so called *elitism* approach which keeps good solutions. For this purpose an archive population $\mathcal{A}$ is kept over all iterations. The archive population $\mathcal{A}$ includes a set of individuals which build a Pareto-optimal set i.e. it keeps the best (elite) individuals over all the generations and is updated by new nondominated offspring which have to be included in the Pareto set. In each iteration s from the actual population $\mathcal{P}_s$ and the archive $\mathcal{A}_s$ a new archive population $\mathcal{A}_{s+1}$ is generated by using all non-dominated individuals which build a new Pareto-optimal set. Afterwards weakly dominated individuals are removed and a clustering procedure is used to further reduce the number of elements in $\mathcal{A}_{s+1}$. From the new archive population a set of parents is chosen by tournament selection and with the variation operators new offspring are build to form $\mathcal{P}_{s+1}$. For details on fitness assignment and selection procedure see [Zitzler et al., 2001]. The algorithm terminates, if a maximal number of generations $s_{\max}$ has been reached.

D. The Complete Hybrid Approach

The structure of the complete hybrid approach is shown in Fig. 1. In the EM loop, the EM algorithm from [Buchholz and Panchenko, 2004a] is used to compute a matrix $\mathbf{D_0}$ for a given vector $\boldsymbol{d}_1$ and a given initial distribution $\boldsymbol{\tau}$ such that the pdf of the trace is approximated adequately at some predefined points $t_1, \ldots, t_M$. Initially $\boldsymbol{d}_1$ and $\boldsymbol{\tau}$ are generated randomly. The result of this step is a MAP $(\mathbf{D_0}, \boldsymbol{d}_1\boldsymbol{\tau})$.

The MAP is used as starting point of the EA loop. Thus, the MAP plus $\mu - 1$ MAPs resulting from applying the variation operations to the MAP form the initial population. From the population λ offspring are generated and the new

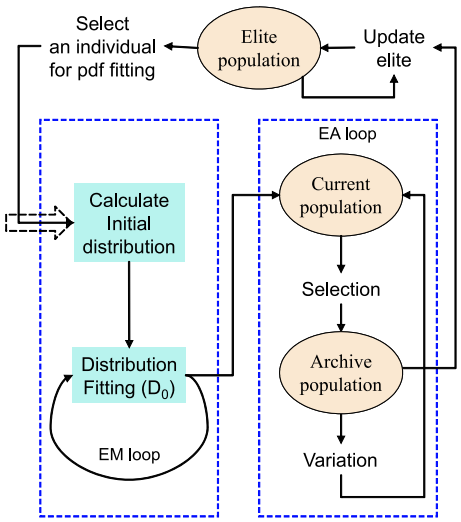


Fig. 1. Interaction between EM and MOEA

archive with μ individuals is built as described in the previous subsection. From the archive λ individuals are chosen via tournament selection and the EA loop starts again with this population. The process stops after a predefined number of generations s_{max} is reached. Experiments indicated that $\mu = 30$, $\lambda = 60$ are good choices to obtain good solutions in appropriate time.

After finishing the EA loop, the archive contains a set of MAPs with identical matrices $\mathbf{D_0}$, but different matrices $\mathbf{D_1}$. All these MAPs are Pareto optimal according to the used fitness functions. The archive is then combined with the elite population, a set that contains the Pareto optimal solutions from the previous iterations. After combination of the archive with the old elite population, the new elite population contains all Pareto optimal solutions. If necessary, the size of the elite population is reduced by the above mentioned clustering procedure. Observe that the individuals in the elite population may have different matrices $\mathbf{D_0}$ since the EM algorithm for pdf fitting modifies $\mathbf{D_0}$. As starting point for the next iteration, an individual with minimal distance to the middle of the elite population is chosen, if it has not already been used as initial individual for pdf fitting.

The algorithm stops if a maximum number of iterations has been performed or if the elite population is not modified in an iteration. After termination a set of MAPs has been generated and one of these MAPs is used as stochastic process to generate the trace. Different possibilities exist to choose one MAP from the elite population. One could use a MAP from the middle of the front or one could compute the likelihood values for all MAPs in the population and choose the MAP with the highest value. The latter approach requires some additional effort which is acceptable if the population is not too large.

IV. Examples

In this series of experiments we apply the two-step EM-MOEA fitting algorithm to obtain compact MAP models that capture pdf as well as autocorrelation dependencies of the well known pAug89 trace from the Internet traffic archive [tra]. The trace contains 10^6 elements, has nontrivial distribution shape and exhibits autocorrelation dependen-

	pAug	M_p	M_1	M_2	M_3	M_a	EM2s
$E[t]$	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$E[t^2]$	4.22	3.34	3.74	3.16	2.53	2.67	3.58
$E[t^3]$	64.8	30.9	34.1	22.8	13.4	13.6	33.7
Median	0.62	0.67	0.59	0.62	0.68	0.62	0.63
$Kurtosis$	153	60.7	43.3	34.3	25.9	18.5	55.5
H	0.79	0.64	0.673	0.65	0.656	0.67	0.51

TABLE I: Statistics of fitted MAPs for pAug89 trace

cies over 1000 lags. The original trace is scaled such that its mean value is $E[\boldsymbol{t}] = 1$.

The choice of free algorithm parameters was based on the previous results obtained in [Buchholz and Panchenko, 2004a,b] and on the series of experiments with MOEA. Previous experiments with distribution fitting indicated that an appropriate approximation of the pdf can already be obtained with $n = 5$ states. A further increase of the state space results in only minor improvements in the pdf quality, but improves the autocorrelation quality. Based on these results we keep the state space compact with $n = 5$ and apply the evolutionary algorithm for modification of the arrival rate matrix $\mathbf{D}_1$. As termination criteria for the EM algorithm we use a bound of $\epsilon = 5 \cdot 10^{-4}$ for the difference of values in the matrices of two subsequent iterations. The loop of the evolutionary algorithm is terminated if $s_{\max} = 150$ generations have been generated. The size of archive population equals $\mu = 30$ and the global elite population $\mathcal{E}$ is also of size 30. As variation operators the element swap mutation and the uniform mutation of matrix rows are applied.

The quality of the MAP models has been evaluated by comparing the simulation of a $\bullet/D/1/5000$ queue with the MAP models and the original traces as workloads under high and low utilization $\rho = 0.95$ and $\rho = 0.3$. For the comparison of average queue sizes under utilization ρ the coefficient $QR^{\rho}_{M_x} = E[q_{trace}]/E[q_{M_x}]$ which is defined as the ratio of the average queue sizes for the original trace and for the model M_x. The quality of the MAP models fitted by the EM-MOEA algorithm is compared with the results obtained by the pure EM approach [Buchholz and Panchenko, 2004b].

A. Fitting MAPs to the trace

The fitting procedure was conducted with the parameters defined above and the coefficients of autocorrelation for the objective function f_a were computed up to the lag $K = 300$. For a detailed analysis we select from the set Pareto optimal solutions three MAPs M_1, M_2 and M_3 with different tradeoffs between pdf and autocorrelation fitting, the MAP M_p, that has best quality of the pdf approximation, and the MAP M_a, with the best autocorrelation quality. For these individuals we use a variance-time plot test to estimate the values of the Hurst parameter (fig. 2). Based on the variance-time plot one can state that the MAP models exhibit self-similarity properties close to the pAug89 trace up to the aggregation level 10^3. The best approximation have the individuals M_1 and M_a which have values $H = 0.673$ and $H = 0.669$ for the Hurst parameter, respectively.

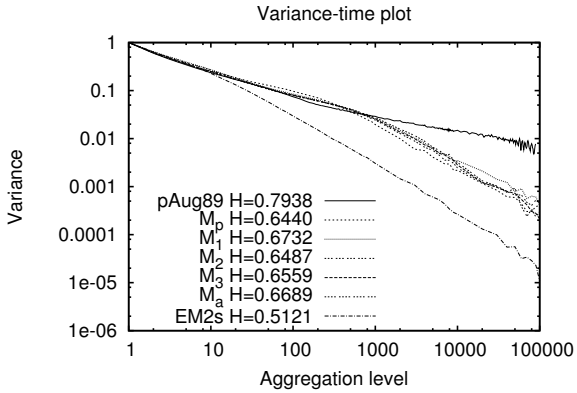


Fig. 2. Variance-time plot for pAug89 models

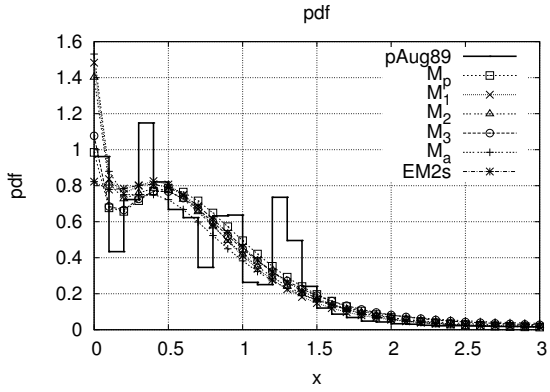


Fig. 3. Main part of pdf for selected MAPs

In table I the basic statistics are given for the selected individuals sorted from the individual M_p with best pdf capturing quality to the individual M_a with best autocorrelation capturing. The last column contains the results for the MAP $EM2s$ fitted by the pure EM algorithm. As one can see from this table the lower moments as well as skewness and kurtosis differ much from the trace statistics for the individuals (M_1, M_a) that exhibit good self-similarity properties. The quality of the h.t.d. properties is good for the individual M_p and for the MAP fitted by two-step EM-algorithm and can be evaluated oby comparing the kurtosis coefficients, which is interpreted as a measure of heavy tailness.

The advantage of the hybrid algorithm is obvious from comparison of the individuals M_p and EM2s, which have a similar quality of pdf fitting. The individual M_p has the worst quality of autocorrelation fitting compared to the other MAPs in the Pareto optimal set. But even for this individual Hurst parameter $H = 0.644$ is much better than for the $EM2s$ MAP model with $H = 0.512$.

B. Queue performance

Evaluation of the $\bullet/D/1/5000$ queue behavior shows that under high load conditions with $\rho = 0.95$ the best queue behavior provides the individual M_1, and under low queue utilization the best results are generated by M_p. But there are still some discrepancies between original and modeled queue behavior.

The variety of model qualities have an impact on the queue

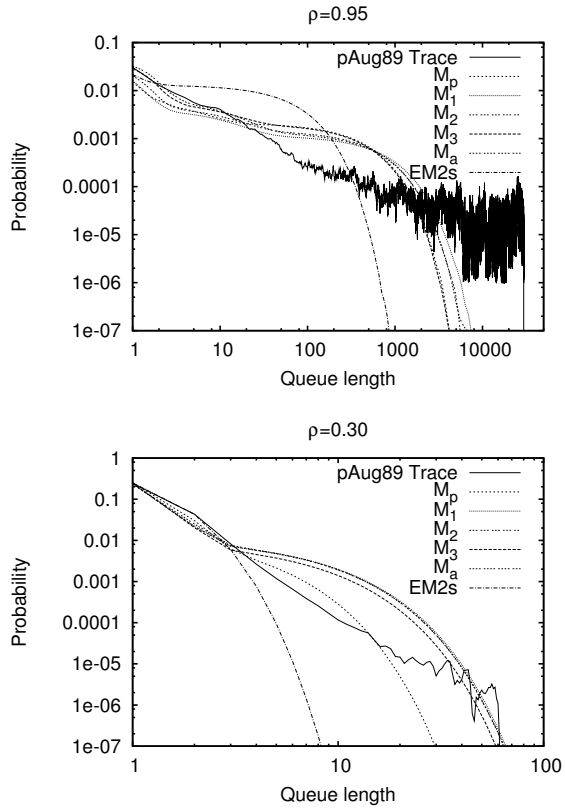


Fig. 4. Queue size distributions with $\rho = 0.95$ and $\rho = 0.3$

	pAug	M_p	M_1	M_2	M_3	M_a	EM2s
$\rho = 0.95$							
$E[q]$	1.0e4	4.1e2	7.0e2	5.5e2	3.8e2	5.3e2	6.8e1
$E[q^2]$	2.1e8	3.6e5	1.1e6	6.7e5	3.3e5	6.7e5	9.8e3
$CV(q)$	1.040	1.078	1.104	1.110	1.123	1.160	1.053
$\rho = 0.30$							
$E[q]$	0.395	0.391	0.624	0.610	0.526	0.618	0.358
$E[q^2]$	1.012	0.837	4.452	4.182	2.933	4.255	0.492
$CV(q)$	2.342	2.116	3.230	3.202	3.099	3.183	1.685

TABLE II: Statistics of queue size for pAug89 models

behavior. First, the third moment and the variance for modeled pdfs are less than for the pAug89 trace. Moreover, the original pdf has several peaks near the points 0.1, 0.4, 0.6, 0.8, 1.3 that can hardly be fitted by the MAP model. Secondly, the autocorrelation dependencies of the models at larger lags (over 1000) essentially differ from the pAug89 trace. Consequently, the models are not able to generate very long packet sequences (bursts) that cause queue overload or its high utilization. Nevertheless, the models fitted by the EM-MOEA algorithm are closer to pAug89 trace than the EM2s model (see figure 4 and table II).

V. Conclusion

We present a hybrid approach to fit the parameters of a Markovian Arrival Process (MAP) according to the values of a trace. The proposed fitting algorithm is independent of the trace length and requires only a compact description of the probability density function and the values of the autocorrelation function. The fitting process is decomposed into two parts, first the pdf is approximated using an algorithm of the expectation maximization type and afterwards the lag k coefficient of correlation is approximated by modifying parts of the MAP resulting from pdf fitting using an evolutionary algorithm. The applied fitness function results in a set of Pareto optimal solutions. One of these solutions can be used to describe the measured trace. Several examples show that the approach yields satisfactory results in an acceptable time.

References

The internet traffic archive. http://ita.ee.lbl.gov/html/traces.html.

S. Asmussen, O. Nerman, and M. Olsson. Fitting phase type distributions via the EM algorithm. *Scandinavian Journal of Statistics*, 23:419–441, 1996.

P. Buchholz. An EM-algorithm for MAP fitting from real traffic data. In P. Kemper and W. H. Sanders, editors, *Computer Performance Evaluation Modelling Techniques and Tools*, volume 2794 of *LNCS*, pages 218–236. Springer, 2003.

P. Buchholz and A. Panchenko. An EM-algorithm for fitting of real traffic traces to PH-distributions. In *Proc. Int. Conference PARELEC'04*, pages 283–288. IEEE Press, 2004a.

P. Buchholz and A. Panchenko. A two-step EM algorithm for MAP fitting. In C. A. et al., editor, *Proc. 19th Int. Symposium ISCIS'04*, volume 3280 of *LNCS*, pages 217–227. Springer, 2004b.

W. Fischer and K. S. Meier-Hellstern. The Markov-Modulated Poisson Process (MMPP) Cookbook. *Performance Evaluation*, 18(2):149–171, 1993.

A. Heindl, K. Mitchell, and A. van de Liefvoort. Correlation bounds for second-order MAPs with application to queueing network decomposition. *Performance Evaluation*, 63:553–577, 2006.

A. Horváth and M. Telek. Markovian Modeling of Real Data Traffic: Heuristic Phase Type and MAP Fitting of Heavy Tailed and Fractal Like Samples. In *Performance 2002*, LNCS 2459, pages 405–434. Springer, 2002.

A. Horvath and M. Telek. Phfit: A general purpose phase type fitting tool. In *Performance Tools 2002*, volume 2324 of *LNCS*, pages 82–91. Springer, 2002.

A. Klemm, C. Lindemann, and M. Lohmann. Modeling IP Traffic Using the Batch Markovian Arrival Process (extended version). *Perf. Evaluation*, 54:149–173, 2003.

M. F. Neuts. *Matrix-Geometric Solutions In Stochastic Models*. John Hopkins University press, London, 1981.

R. Sarker, M. Mohammadian, and X. Yao, editors. *Evolutionary Optimization*. Kluwer, 2002.

E. Zitzler and L. Thiele. An Evolutionary Approach for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, May 1998.

E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report TIK Report Nr. 103, Computer Engineering and Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, May 2001.

MIXED MODELING OF THE CORRELATION STRUCTURE OF VIDEO TRAFFIC

M.E. Sousa-Vieira
Department of Telematics Engineering
University of Vigo
Campus Universitario Lagoas-Marcosende s/n, 36210 Vigo, SPAIN
E-mail: estela@det.uvigo.es

KEYWORDS

M/G/∞ Processes, Persistent Correlations, Traffic Modeling, VBR Video, Discrete Event Simulation.

ABSTRACT

With the increasing popularity of multimedia applications, video data represents a large portion of the load in modern networks. Consequently, adequate models of video traffic, characterized by a high burstiness and a strong positive correlation, are very important for the performance evaluation of network architectures and protocols. In this paper, we propose a new model for the characterization of correlated video traffic, based on the M/G/∞ process. We approximate the autocorrelation function by a combination of functions, and obtain the distribution for the service time of the M/G/∞ queueing system that gives rise to a process with such correlation structure. After analyzing the capability of the model to capture empirical correlations, we study its goodness, applied to VBR MPEG video, to predict queueing performance by simulation.

1. INTRODUCTION

VBR video traffic, characterized by a high burstiness and a strong positive correlation, represents an important part of the load in modern networks. So, adequate models for this type of traffic are needed for networks design and performance evaluation. On the other hand, recent indications of persistent correlations in various types of network traffic (Beran et al. 1995; Garrett and Willinger 1994; Leland et al. 1994; Paxson and Floyd 1995) have contributed to an important shift in the stochastic modeling of traffic, since its impact on the performance metrics may be drastic (Conti et al. 1996; Erramilli et al. 1996; Li and Hwang 1993; Norros 1994), and the validity of traditional processes, like Markovian or AutoRegressive, is in doubt, because modeling long-term correlations through these processes requires many parameters, whose interpretation becomes difficult. Because of this, the use of new classes of stochastic processes for network traffic modeling purposes, able to display forms of correlation as diverse as possible by making use of few parameters (parsimonious modeling) is essential.

Some of these processes, commonly used in simulation studies, are Fractional Gaussian Noise (FGN), Fractional AutoRegressive Integrated Moving Average (F-ARIMA) and M/G/∞. The main drawback of FGN and F-ARIMA processes is that only off-line approximate methods for synthesizing traces (López et al. 2000) are efficient enough to be of practical use, being their complexity $O(n\log(n))$, with n the length of the sample path, and this off-line generation can restrict the simulation time horizon of experiments due to memory requirements.

The M/G/∞ process is a stationary version of the occupancy process of an M/G/∞ queueing model. Its viability for modeling network traffic can be attributed to several factors. Firstly, many forms of time dependence can be obtained by means of varying the service time distribution. This makes this process a good candidate for modeling many types of correlated traffic, such as VBR video. Secondly, queueing analytical studies are sometimes feasible (Parulekar 2001; Resnick and Samorodnitsky 1999; Tsoukatos and Makowski 1997), and when they are not, it has important advantages in simulation studies (Krunz and Makowski 1998; Poon and Lo 2001), such as the possibility of on-line generation. Furthermore, there exists a trivial method of producing exact sample paths of the process with complexity $O(n)$: it suffices to simulate the M/G/∞ queue, sampling the occupancy of the system at integer instants.

After examining the empirical autocorrelation function of some VBR video sequences, we opt for approximating it by a combination of functions, smoothing the transitions between them by means of cubic spline interpolation and we obtain the appropriate distribution for the service time of the M/G/∞ queueing system that provides such correlation structure. The resulting generator is easy to initialize in steady state.

It is well-known that the tail of the marginal distribution plays an important role in performance evaluation (Grossglauser and Bolot 1996). To model the empirical marginal distribution of real video sequences, we need to transform the Poisson marginal distribution of the M/G/∞ process into a more appropriate form, such as the Lognormal distribution, and small values of the arrival rate λ of the Poisson input pro-

cess are inappropriate for the transformantion process (Poon and Lo 2001). On the other hand, the complexity of the generator is an increasing function of λ, that is directly related to the mean value of the M/G/∞ process. As the resulting distribution for the service time has subexponential decay, we can apply the method that we proposed in (Sousa et al. 2006) in order to get a highly efficient generator of synthetic traces.

Finally, it is important to highlight that although there are many reasons to argue that the validation of a source model requires that its distribution and autocorrelation functions match well their empirical counterparts, while using as few parameters as possible, the ultimate goal is to predict queueing performance. In most cases, the goodness of a model is determined by comparing model predictions, typically simulation statistics using a queueing system, and the corresponding statistics using the empirical data as the traffic source. As the last contribution of the paper, we validate our model showing its capability to capture the marginal distribution and correlation structure of VBR MPEG video traffic, and study its goodness to predict queueing performance by simulation.

The remainder of the paper is organized as follows. In Section 2 we give an overview of the M/G/∞ process. The M/G/∞ based model is explained in Section 3. In Section 4 we show the capability of our solution for capturing empirical correlation structures and in Section 5 we show the results related to queueing performance. In Section 6 we apply the model to the generation of traffic with Long-Range Dependence (LRD) properties. Finally, the conclusions are summarized in Section 7.

2. M/G/∞ PROCESS

An interesting self-similar process is the occupancy process of an M/G/∞ queueing system. In such queueing model, customers arrive according to a Poisson process with rate λ to a system of infinitely many servers, and their service times constitute a sequence of continuous i.i.d. random variables distributed as the random variable S with finite mean value $\mathsf{E}[S]$.

In (Cox and Isham 1980) the authors show that the number of customers, or busy servers, in the system at any instant t, $\{X(t); t \in \Re^+\}$, has a Poisson marginal distribution with mean value $\mathsf{E}[S]$ and its autocorrelation function depends on the distribution of the service time.

We are interested on the discrete-time version of $\{X(t); t \in \Re^+\}$, that is, $X \stackrel{\Delta}{=} \left\{X_n \stackrel{\Delta}{=} X(n); n = 1,2,\ldots\right\}$. The most natural approach to generate it is to simulate the queue in discrete-time, since its simulation will be more efficient (Suárez et al. 2002).

2.1. Discrete-time Model

Let $A = \{A_n; n = 1,2,\ldots\}$ be a renewal stochastic process, where A_n is a Poisson random variable with mean value λ and represents the number of arrivals at instant n; let $\{\{S_{n,i}; i = 1,\ldots,A_n\}; n = 1,2,\ldots\}$ be a renewal stochastic process where $S_{n,i}$ is distributed as a positive-valued discrete random variable S with finite mean value $\mathsf{E}[S]$, and corresponds to the service time of the i-th arrival at instant n. If the following conditions hold:

- the initial number of users X_0 is a Poisson random variable of mean value $\lambda\mathsf{E}[S]$,
- their service times $\left\{\widehat{S}_j; j = 1,\ldots,X_0\right\}$ are mutually independent and have the same distribution as the residual life of S, $\widehat{S}$:

$$\Pr\left[\widehat{S} = k\right] = \frac{\Pr[S \geq k]}{\mathsf{E}[S]},$$

then the stochastic process $X = \{X_n; n = 1,2,\ldots\}$ is strict-sense stationary and ergodic, and satisfies:

- it has Poisson marginal distribution and mean value $\mathsf{E}[X] = \lambda\mathsf{E}[S]$,
- its autocorrelation function is $\mathrm{r}_k = \Pr\left[\widehat{S} > k\right] \forall k$.

So, the autocorrelation structure of X is completely determined by the distribution of $\widehat{S}$ or, ultimately, by the distribution of S. Varying this distribution many forms of time dependence can be obtained.

In (Krunz and Makowski 1998) the authors show that an $\Re^+$-valued sequence $\{\mathrm{r}_k; k = 0,1\ldots\}$ can be the autocorrelation function of the stationary M/G/∞ process, with integrable S, if and only if r_k is decreasing and integer-convex, with $\mathrm{r}_0 = 1 > \mathrm{r}_1$ and $\lim_{k\to\infty} \mathrm{r}_k = 0$, in which case the distribution of S is given by:

$$\Pr[S = k] = \frac{\mathrm{r}_{k-1} - 2\mathrm{r}_k + \mathrm{r}_{k+1}}{1 - \mathrm{r}_1} \quad \forall k > 0. \tag{1}$$

3. M/G/∞ BASED MODEL

With this method, we model the autocorrelation of the M/G/∞ process as a combination of functions:

$$\mathrm{r}_k = \begin{cases} y_1[k] & k \leq \mathsf{K}_1 \\ y_2[k] & \mathsf{K}_1 < k \leq \mathsf{K}_2 \\ \ldots & \\ y_{\mathsf{M}}[k] & k > \mathsf{K}_{\mathsf{M}-1}. \end{cases}$$

In order to smooth the transitions, and get a distribution for the service time with subexponential decay, we apply cubic spline interpolation. The goal is to obtain interpolation functions that are continuous in the second derivative, both within

an interval around the transition points, K_i, and at its boundaries. The resulting autocorrelation function is:

$$r_k = \begin{cases} y_1[k] & k \le K_{1,1} \\ y_{1\_2}[k] & K_{1,1} \le k \le K_{1,2} \\ y_2[k] & K_{1,2} \le k \le K_{2,1} \\ y_{2\_3}[k] & K_{2,1} \le k \le K_{2,2} \\ \dots & \\ y_{M-1\_M}[k] & K_{M-1,1} \le k \le K_{M-1,2} \\ y_M[k] & k \ge K_{M-1,2}, \end{cases} \quad (2)$$

with:

$$y_{i-1\_i}[k] = A_{i-1}[k]\,y_{i-1}[K_{i-1,1}] + B_{i-1}[k]\,y_{i-1}[K_{i-1,2}] + C_{i-1}[k]\,y''_{i-1}[K_{i-1,1}] + D_{i-1}[k]\,y''_{i-1}[K_{i-1,2}],$$

where:

$$A_{i-1}[k] = \frac{K_{i-1,2} - k}{K_{i-1,2} - K_{i-1,1}}, \quad B_{i-1}[k] = 1 - A_{i-1}[k],$$

$$C_{i-1}[k] = \frac{1}{6}(A^3_{i-1}[k] - A_{i-1}[k])(K_{i-1,2} - K_{i-1,1})^2,$$

$$D_{i-1}[k] = \frac{1}{6}(B^3_{i-1}[k] - B_{i-1}[k])(K_{i-1,2} - K_{i-1,1})^2,$$

and where $y''_{i-1}[K_{i-1,1}]$ and $y''_{i-1}[K_{i-1,2}]$ are the second derivatives of the autocorrelation function at $K_{i-1,1}$ and $K_{i-1,2}$, all for $2 \le i \le M$.

Finally, from (1) and (2) we have the distribution of the service time, $\Pr[S = k]$.

4. MODELING VBR VIDEO TRACES

4.1. Modeling the Correlation Structure

The correlation behavior of the examined empirical VBR video traces (Fitzek and Reisslein 2001) can be adequately captured by:

$$r_k = \begin{cases} e^{-\beta_1 k^{\alpha_1}} & k \le K_{1,1} \\ y_{1\_2}[k] & K_{1,1} \le k \le K_{1,2} \\ e^{-\beta_2 k^{\alpha_2}} & k \ge K_{1,2}, \end{cases} \quad (3)$$

with $\beta_1 > 0$, $\beta_2 > 0$, $0 < \alpha_1 \le 1$, $0 < \alpha_2 < 1$ and $\alpha_2 < \alpha_1$.

In this case:

$$y''_1[K_{1,1}] = e^{-\beta_1 K_{1,1}^{\alpha_1}} \left[\beta_1^2 \alpha_1^2 K_{1,1}^{2(\alpha_1-1)} - \beta_1 \alpha_1 (\alpha_1 - 1) K_{1,1}^{\alpha_1 - 2}\right],$$

$$y''_2[K_{1,2}] = e^{-\beta_2 K_{1,2}^{\alpha_2}} \left[\beta_2^2 \alpha_2^2 K_{1,2}^{2(\alpha_2-1)} - \beta_2 \alpha_2 (\alpha_2 - 1) K_{1,2}^{\alpha_2 - 2}\right].$$

The distribution for the service time, $\Pr[S \le k]$, is:

- $1 - (e^{-\beta_1 k^{\alpha_1}} - e^{-\beta_1 (k+1)^{\alpha_1}})(1 - e^{-\beta_1})^{-1}$; $k < K_{1,1}$
- $1 - (e^{-\beta_1 k^{\alpha_1}} - y_{1\_2}[k+1])(1 - e^{-\beta_1})^{-1}$; $k = K_{1,1}$
- $1 - (y_{1\_2}[k] - y_{1\_2}[k+1])(1 - e^{-\beta_1})^{-1}$; $K_{1,1} < k < K_{1,2}$
- $1 - (y_{1\_2}[k] - e^{-\beta_2 (k+1)^{\alpha_2}})(1 - e^{-\beta_1})^{-1}$; $k = K_{1,2}$
- $1 - (e^{-\beta_2 k^{\alpha_2}} - e^{-\beta_2 (k+1)^{\alpha_2}})(1 - e^{-\beta_1})^{-1}$; $k > K_{1,2}$.

The mean is:

$$E[S] = \frac{1}{1 - e^{-\beta_1}}.$$

And the distributionn of the residual life, that is necessary to initialize the process in steady state is:

$$\Pr\left[\widehat{S} \le k\right] = \begin{cases} 1 - e^{-\beta_1 k^{\alpha_1}} & k \le K_{1,1} \\ 1 - y_{1\_2}[k] & K_{1,1} \le k \le K_{1,2} \\ 1 - e^{-\beta_2 k^{\alpha_2}} & k \ge K_{1,2}, \end{cases}$$

4.2. Modeling the Marginal Distribution

The M/G/∞ process has Poisson marginal distribution whose tail drops faster than that of the empirical distribution of real video sequences. The tail of the marginal distribution plays an important role in performance evaluation (Grossglauser and Bolot 1996). So, we need to transform the Poisson marginal distribution into a more appropriate one.

In this respect, in (Huang et al. 1995) authors demonstrate that Gaussian processes enjoy the attractive advantage that their marginal distribution can be changed by a sufficiently regular transformation without modifying the long-range correlation structure. For this reason, it is desirable to generate processes with large enough mean value, because the Poisson distribution tends to Gaussian when the mean increases.

The change of distribution will be done by the following transformation:

$$T(x) = F_Y^{-1}(F_X)$$

where F_X is the original marginal distribution and F_Y is the target marginal distribution.

Regarding the appropriate distribution, we choose the Lognormal distribution because gives rise to good approximations of the GOP (Group of Pictures) sizes for a wide range of MPEG video sequences (Rose 1997), and we are interesting in modeling this type of traffic.

4.3. Estimation of the Parameters of the Model

The application of this model to the generation of traces with a correlation structure similar to that of the real sequences we are interested in requires the estimation of the parameters of the model.

One possibility is to estimate them by means of least square fitting of the empirical autocorrelation function. Another one, that we will investigate in a following work, is to use the Whittle estimator (Taqqu and Teverovsky 1998; Whittle 1953).

5. FITTING EMPIRICAL TRACES

We have applied the model to generate MPEG-4 encoded video traffic at the GOP level. For the sake of brevity, we present only the results for the empirical trace *"Jurassic Park I"*, although the conclusions are valid for all the traces we have analyzed (Fitzek and Reisslein 2001).

For this trace we choose $\alpha_1 = 1$ and $\alpha_2 = 0.5$, and we obtain $\beta_1 = 0.061875$, $\beta_2 = 0.210721$, $K_1 = 11$, $K_{1,1} = 7$ and $K_{1,2} = 23$.

In Figure 1 we plot the resulting matching of the autocorrelation function for the first lags and in Figure 2 we show the distribution for the service time, $\Pr[S = k]$.

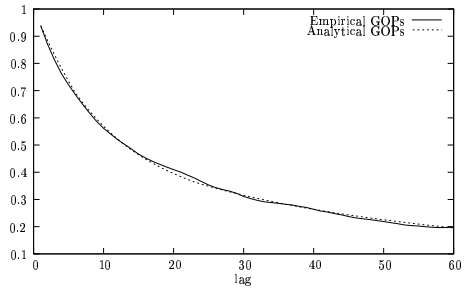


Figure 1. Autocorrelation matching (*"Jurassic Park I"*)

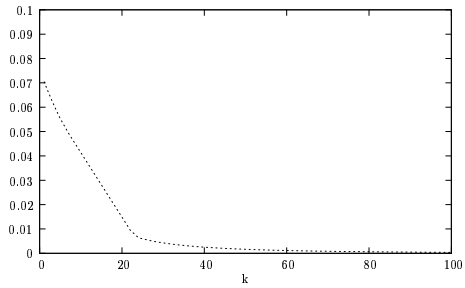


Figure 2. Distribution for the service time (*"Jurassic Park I"*)

In Figure 3 we represent the autocorrelation function of a synthetic trace of the resulting M/G/∞ process. We can observe a good fit for that corresponding to the empirical trace.

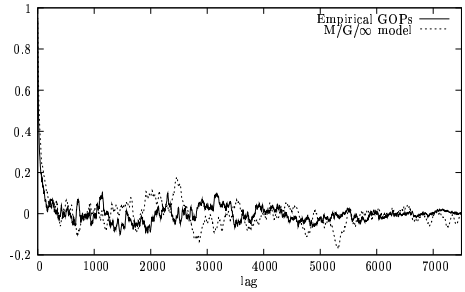


Figure 3. Autocorrelation matching of the M/G/∞ process (*"Jurassic Park I"*)

5. QUEUEING PERFORMACE

To verify the appropriateness of our model, we have made exhaustive simulation experiments of a single-server FIFO queueing system with finite buffer. GOPs arrive at the switch at a constant rate. The capacity of the server or the utilization factor, and the buffer size are system variables. We have analyzed two performance statistics of the system: loss probability and mean delay. Many values of the system variables were used in the simulation experiments.

In Figures 4 and 5 we show the results for a buffer size of 0.6 MBits and utilization in the range 0.4 - 0.9, for six synthetic traces (we generate them varying the seed of the generator). We could observe a good agreement between the model results and the results using the empirical trace (continuous line). Slight differences are reasonable, because of the short length of the sequence, 9000 GOPs.

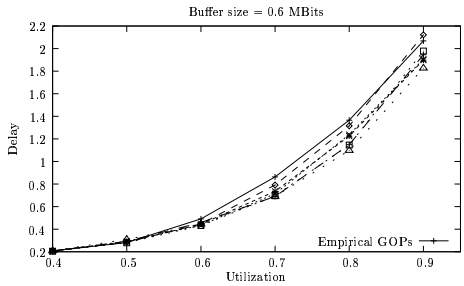


Figure 4. Mean delay results varying the utilization (*"Jurassic Park I"*)

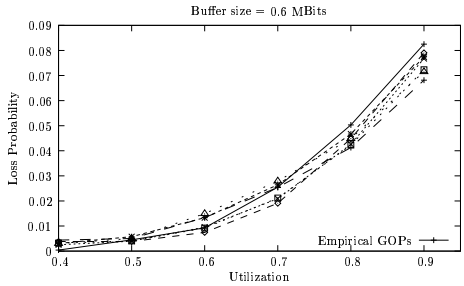


Figure 5. Loss probabibity results varying the utilization (*"Jurassic Park I"*)

6. APPLICATION OF THE MODEL TO THE GENERATION OF TRACES WITH LRD PROPERTIES

6.1. LRD Processes

It is said that a process exhibits LRD when its autocorrelation function is not summable (or, equivalently, its spectral density has a singularity at the origin), i.e., $\sum_{k=1}^{\infty} r_k = \infty$, like in those processes whose autocorrelation function decays hy-

perbolically:

$$\exists \beta \in (0,1) \left| \lim_{k\to\infty} \frac{r_k}{k^{-\beta}} = c_r \in (0,\infty) \right. .$$

Applied to the traffic of modern networks, the LRD property implies the existence of persistent correlations and their presence at multiple time scales (Cox 1994).

The M/G/∞ process belongs to the class of LRD processes if the service-time has infinite variance, as it may happen in heavy-tailed distributions.

A discrete random variable S has a heavy-tailed distribution if it complies with:

$$\lim_{k\to\infty} \frac{\Pr[S>k]}{k^{-\alpha}} = L[k] \quad 0<\alpha<\infty,$$

with L a function that varies slowly, i.e.:

$$\lim_{k\to\infty} \frac{L[kx]}{L[k]} = 1 \quad \forall x>0.$$

6.2. Example

As an example of VBR traffic whose correlation behavior can be adequately approximated by a combination of functions that gives rise to an LRD process, we consider DCT intra-coding video traffic, at the frame level. We present the results for the empirical trace *"Star Wars"* (Krunz and Makowski 1998).

We consider the following autocorrelation function:

$$r_k = \begin{cases} e^{-\beta_1 k^{\alpha_1}} & k \le K_{1,1} \\ y_{(1\_2)}[k] & K_{1,1} \le k \le K_{1,2} \\ k^{-\beta_2} & k \ge K_{1,2}, \end{cases} \qquad (4)$$

with $\beta_1>0$, $\beta_2>0$ and $0<\alpha_1\le 1$.

In this case:

$$y_1''[K_{1,1}] = e^{-\beta_1 K_{1,1}^{\alpha_1}} \left[\beta_1^2\alpha_1^2 K_{1,1}^{2(\alpha_1-1)} - \beta_1\alpha_1(\alpha_1-1)K_{1,1}^{\alpha_1-2)}\right],$$

$$y_2''[K_{1,2}] = \beta_2(\beta_2+1)K_{1,2}^{-\beta_2-2}.$$

For this trace we choose $\alpha_1 = 0.5$ and we obtain $\beta_1 = 0.075802$. β_2 is related with the Hurst parameter (Hurst 1951) ($H = 1-\frac{\beta_2}{2}$). We estimate it using the aggregated Whittle estimator (Taqqu and Teverovsky 1998) and we obtain $\beta_2 = 0.4$. The transition point is $K_1 = 1485$ and $K_{1,1} = 618$ and $K_{1,2} = 1735$.

In Figure 6 we plot the resulting matching of the autocorrelation function. Although we have obtained a good fit, it could be improved at first lags considering a combination of (3) and (4).

In Figure 7 we can see the distribution for the service time of the resulting M/G/∞ process.

Finally, in Figure 8 we represent the autocorrelation function of a synthetic trace of the resulting M/G/∞ process. Again, we can observe a good fit for that corresponding to the empirical trace.

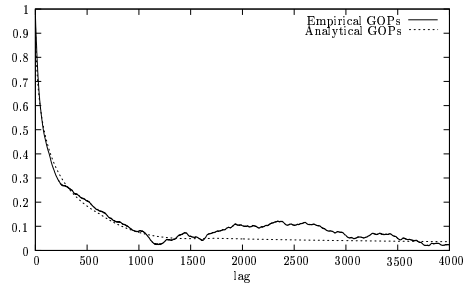


Figure 6. Autocorrelation matching (*"Star Wars"*).

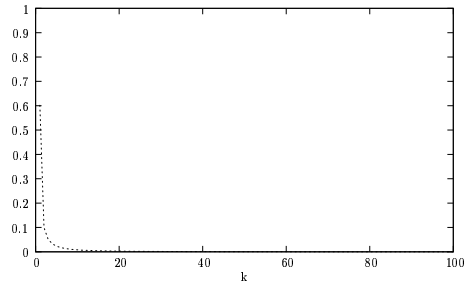


Figure 7. Distribution for the service time (*"Star Wars"*).

6. CONCLUSIONS

In this paper we have proposed a new model for VBR video traffic based on the M/G/∞ process. Our solution enjoys several features that make it very attractive to model this type of traffic: the possibility of on-line generation, a highly efficient simulation model of the M/G/∞ queue that allows to deal with a wide range of input parameters and the possibility to capture both the short-term and the long-term correlation behavior of traffic in a parsimonious way. As an example, we apply our model to capture the correlation structure of VBR MPEG video traffic. Simulation results of performance metrics show that both the synthetic and the real traces exhibit similar enough queueing behavior. In a following work, we are going to investigate the application of the Whittle estimator to our model.

REFERENCES

Beran, J., Shreman, R., Taqqu, M.S., Willinger, W., 1995. "Long-Range Dependence in Variable-Bit-Rate Video Traffic". IEEE Transactions on Communications, 43-2/4, 1566–1579.

Conti, M., Gregori, E. Larsson, A., 1996. "Study of the Impact of MPEG-1 Correlations on Video Sources Statistical Multiplexing". IEEE Journal on Selected Areas in Communications, 14-7, 1455-1471.

Cox, D.R., Isham, V., 1980, *Point Processes*. Chapman and Hall.

Cox, D.R., 1984. "Long-Range Dependence: A Review". In Statistics: An Appraisal. Iowa State University Press, 55–

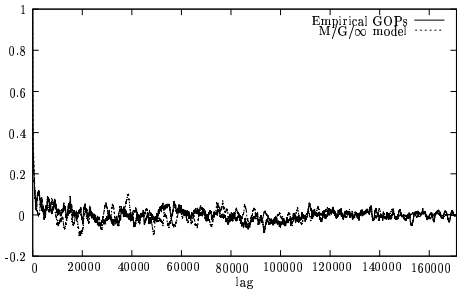


Figure 8. Autocorrelation matching of the M/G/∞ process (*"Star Wars"*).

74.

Erramilli, A., Narayan, O., Willinger, W., 1996. "Experimental Queueing Analysis with Long-Range Dependent Packet Traffic". IEEE/ACM Transactions on Networking, 4-2, 209–223.

Fitzek, F.H.P., Reisslein, M., 2001. "MPEG-4 and H.263 Video Traces for Network Performance Evaluation". IEEE Network, 15-6, 40–54.

Garrett, M.W., Willinger, W., 1994. "Analysis, Modeling and Generation of Self-similar VBR Video Traffic". In Proceedings of the ACM SIGCOMM'94, London, UK, 269–280.

Grossglauser, M., Bolot, J.-C., 1996. "On the Relevance of Long-Range Dependence in Network Traffic". In Proceedings of the ACM SIGCOMM'96, Stanford, CA, USA, 269–280.

Huang, C., Devetsikiotis, M., Lambadaris, I., Kayevol, A.R., 1995. "Modeling and Simulation of Self-similar Variable Bit Rate Compressed Video: A Unified Approach". In Proceedings of the ACM SIGCOMM'95, Cambridge, UK, 114–125.

Hurst, H.E., 1951. "Long-term Storage Capacity of Reservoirs". Transactions of the American Society of Civil Engineers, 116, 770–799.

Krunz, M., Makowski, A., 1998. "Modeling Video Traffic Using M/G/∞ Input Processes: A Compromise between Markovian and LRD Models". IEEE Journal on Selected Areas in Communications, 16-5, 733–748.

Leland, W.E., Taqqu, M.S., Willinger, W., Wilson, D.V., 1994. "On the Self-similar Nature of Ethernet Traffic (extended version)". IEEE/ACM Transactions on Networking, 2-1, 1–15.

Li, S.Q., Hwang, C.L., 1993. "Queue Response to Input Correlation Functions: Discrete Spectral Analysis". IEEE/ACM Transactions on Networking, 1-5, 317–329.

López, J.C., López, C., Suárez, A., Fernádez, M., Rodríguez, R.F., 2000. "On the Use of Self-similar Processes in Network Simulation". ACM Transactions on Modeling and Computer Simulation, 10-2, 125–151.

Norros, I., 1994. "A Storage Model with Self-similar Input". Queueing Systems, 16, 387–396.

Parulekar, M., 2001. *Buffer Engineering for M/G/∞ Input Processes*. Ph.D. Thesis. University of Maryland, College Park, MD, USA.

Paxson, V., Floyd, S., 1995. "Wide-area Traffic: The Failure of Poisson Modeling". IEEE/ACM Transactions on Networking, 3-3, 226–244.

Poon, W., Lo, K., 2001. "A Refined Version of M/G/∞ Processes for Modeling VBR Video Traffic". Computer Communications, 24-11, 1105–1114.

Resnick, S., Samorodnitsky, G., 1999. "Activity Periods of an Infinite Server Queue and Performance of Certain Heavy-tailed Fluid Queues. Queueing Systems, 33-1/3, 43–71.

Rose, O., 1997. "Simple and Efficient Models for Variable Bit Rate MPEG Video Traffic". Performance Evaluation, 30-1, 69–85.

Sousa, M.E., Suárez, A., Fernández, M., López, C., Rodríguez, R.F., 2006. "A Highly Efficient M/G/∞ Generator of Self-similar Traces". 2006 Winter Simulation Conference, Monterey, CA, USA, 2146–2153.

Suárez, A., López, J.C., López, C., Fernández, M., Rodríguez, R.F, Sousa, M.E., 2002. "A New Heavy-tailed Discrete Distribution for LRD M/G/∞ Sample Generation". Performance Evaluation, 47-2/3, 197–219.

Taqqu, M.S., Teverovsky, V., 1998. "On Estimating the Intensity of Long-Range Dependence in Finite and Infinite Variance Time Series". In A Practica Guide to Heavy Tails. R. Adler, R. Feldman and M.S. Taqqu Eds. Birkhauser, 177-217.

Tsoukatos, K.P., Makowski, A.M., 1997. "Heavy Traffic Analysis for a Multiplexer Driven by M/G/∞ Input Processes". In Proceedings of the 15th International Teletraffic Congress, Washington, DC, USA, 497–506.

Whittle, P., 1953. "Estimation and Information in Stationary Time Series". Arkiv Matematick, 2-23, 423–434.

AUTHOR BIOGRAPHY

M.E. SOUSA-VIEIRA is an assistant professor in the Department of Telematics Engineering at University of Vigo. She received a telecommunication engineering degree from University of Vigo in 1998. Her current research interests are in the area of performance evaluation in communication networks, more specifically simulation methodology and traffic modeling. Her e-mail address is <estela@det.uvigo.es>.

PARAMETER ESTIMATION FOR SEMI-MARKOV MODELS FOR MOBILE VIDEO TRAFFIC

Sebastian Kempken
Wolfram Luther
Abteilung Informatik
Universität Duisburg-Essen
Duisburg, Germany
{kempken, luther}@informatik.uni-duisburg.de

Gerhard Haßlinger
T-Systems ENPS
Darmstadt, Germany
gerhard.hasslinger@t-systems.com

ABSTRACT

Semi-Markov models can be used to represent a wide spectrum of data traffic sources in communication networks. Furthermore, both efficient and reliable analysis techniques are available. In this work, we extend a newly proposed technique to estimate the parameters of a discrete-time semi-Markov process from given video traces for mobile devices. We consider the distribution and the autocorrelation of the trace and of the semi-Markov model, respectively, to study the fitting procedure according to those characteristics. This new technique relies on an evolutionary optimization approach employing genetic programming. This way, the quality of the models can be significantly improved in comparison to traditional methods.

KEYWORDS

Reliable algorithms, stochastic traffic modeling, genetic programming, semi-Markov processes

INTRODUCTION

Video content for mobile devices is enjoying more and more popularity as the number of possible applications increases: Third generation networks allow video telephony and real-time streaming of video contents, both via traditional broadcasting and, at the same time, as user-generated content on emerging and established web sites like YouTube[1]. Hence, an increasing amount of mobile video traffic in service-integrated communication networks needs to be reflected in traffic modeling techniques.

Nodes in these communication networks - that is, routers and switches - receive data over a number of input links and distribute it over one or more output links. A particular output line is chosen according to forwarding strategies defined by routing protocols like OSPF and BGP both standardized by the Internet engineering task force. Furthermore, buffers must also be considered in the routing process. In general, each router disposes of an output buffer for each outgoing connection. In this way, it is able to store all data packets and can forward them again after an appropriate delay in case of transmission collisions or temporary overload situations.

[1] http://www.youtube.com

We consider time-slotted systems an efficient method to model and analyze such a router. This approach is often taken in performance analysis of telecommunication systems (Li, 1991; Haßlinger, 1997, 2000). The time is divided into time slots of equal length L and the amount of data arriving at or leaving the router is traced per slot. Usually, the output capacity C of a router may be considered constant, so that an amount of data CL can be transmitted in each time slot. In contrast, the quantity of arriving data is often highly variable and, in the case of video transmissions, also shows a high level of autocorrelation. For this reason, we consider semi-Markov processes (SMP) to model the arrival process to reflect these characteristics (Kempken and Luther, 2006). An analysis of such models is possible using either matrix-analytical (Latouche and Ramasvami, 1999) or factorization techniques (Li, 1991; Haßlinger, 2000; Luther and Traczinski, 2006; Kempken et al., 2006; Traczinski et al., 2005). Furthermore, there are several modeling techniques for video traffic based on semi-Markov processes (Rose, 1997; Haßlinger and Takes, 2003; Krunz, 2000; Salvador et al., 2003). We consider the accurate representation of both the distribution and the autocorrelation to be key characteristics for the quality of such a model. In terms of the latter criterion, video traffic shows a high level of long-term correlation (Rose, 1997). This is difficult to include in semi-Markov models since the autocorrelation function of an SMP shows exponentially decaying behavior. However, we find that by choosing the parameters of an SMP carefully, the quality of the model can be significantly improved. In (Kempken and Luther, 2007), we presented a modeling approach for high definition video data that relies on an evolutionary optimization strategy and shows favorable results. In this paper, we extend this technique to handle the special characteristics of mobile video traffic.

In the following section, we describe the typical characteristics of video streams for mobile devices. Next, we formally introduce semi-Markov processes and explain our approach to the task of parameter estimation of the SMP model. A numerical example is given before we conclude the results.

MOBILE VIDEO

Mobile video contents are most often delivered in the 3GP container format, using MPEG-4 video encoding

and Adaptive Multi Rate (AMR) or Advanced Audio Coding (AAC) for the audio stream. Since the audio stream is coded with a constant bitrate in almost all cases, we focus on modeling the video data, which is coded using a variable bitrate scheme.

The MPEG-4 Part 2 video standard (MPEG-4), defines a coding technique for compressed video streams (cf. Pereira (2002)). The compression is achieved by reducing both the spatial and temporal redundancies. The former refers to the compression of a single frame using entropy coding and transforms; the latter considers changes on a frame-to-frame level. The compression here is done by predicting future frames using motion vectors. Hence, we may distinguish three types of frames:

1. I-frames: These use intraframe coding only. The compression is achieved through such methods as the discrete cosine transform and entropy coding.
2. P-frames: In addition to intraframe coding techniques, these frames are predicted using motion compensation referring to the preceding I- or P-frame.
3. B-frames: These bidirectional frames code the differences between the actual picture and an interpolation between the preceding and subsequent P- or I-frames.

The usual case is that B-frames have the lowest bandwidth requirements, and I-frames are the largest frames in a sequence. The video coding standard requires the frames to be in a certain order, for example "IBBBPBBB" or "IBBPBBPBBPBB". Such a sequence is called a group of pictures (GoP). However, the length of such a group may vary within a stream up to a given limit to achieve higher compression. Video streams for mobile devices often make use of this technique to accomplish lower bandwidth requirements. Furthermore, they often feature low resolutions and frame rates.

A video traffic stream can be considered at several levels (cf. Rose (1997)):

- Sequence layer: the whole video sequence (minutes to hours)
- Scene layer: given in intervals where the content of the pictures is almost the same (seconds to minutes)
- GoP layer: defined by a group of pictures (tenths of a second)
- Frame layer: considers the period of a single frame (hundredths of a second)

Since we intend to provide a reliable model in terms of distribution and autocorrelation, we use a low level of the video stream as a basis for the modeling approach.

On the frame level, the coding scheme together with the typical size relations of the frame types leads to an autocorrelation behavior typical of MPEG video streams. Furthermore, long-term dependency is typical of video streams. The sub-exponential decay of the autocorrelation function is difficult to model using a Markov chain-based approach since this leads to an exponentially decaying autocorrelation function (Feldmann and Whitt, 1998; Kempken and Luther, 2006).

In the modeling of MPEG-based video streams, we therefore face two challenges. On the one hand, we must be able to reproduce typical periodic autocorrelation behavior while, on the other hand, we have to find a way to model occurring long-term dependencies using an exponentially decaying autocorrelation.

If the length of the groups of pictures is constant, it is also feasible to model on the GoP level (cf. Rose (1997)). To do so, we use a given video trace to determine the mean size of a GoP and the mean size of each frame inside the GoP. If, for instance, the GoP consists of 12 frames, we yield the same number of scaling factors by dividing the mean frame size by the mean size of all GoPs. These scaling factors can now be used to reconstruct the sizes of particular frames from the size of a GoP, which results from the semi-Markovian model. However, if the length of the GoPs vary, as is the case for most video streams intended for mobile devices, this approach is not applicable. In the following, we present a technique for modeling the frame sizes directly, including the periodic behavior of the autocorrelation.

SEMI-MARKOV MODELS

We introduce a discrete-time homogeneous semi-Markov process (SMP) to describe the workload behavior of a given queueing system (cf. Kleinrock (1975)). A family of random variables $\{(R_t, \sigma_t)|t \in \mathbb{N}_0\}$ denotes such a process if

$$P\left(R_{t+1} = a, \sigma_{t+1} = j | R_k = a_k, \sigma_k = i_k\right) =$$

$$P\left(R_{t+1} = a, \sigma_{t+1} = j | \sigma_t = i_t\right), 1 \leq k \leq t$$

holds for all $t \in \mathbb{N}$ and $\{\sigma_t\}$ is a Markov chain. In our case, R_t denotes the t-th workload change. In the following, we restrict ourselves to the case of a finite Markov chain. If the chain $\{\sigma_t\}$ has M states, the semi-Markov process is called SMP(M).

To define a semi-Markov process completely, the transition matrix $P := (p_{ij})$ of the underlying chain with M^2 distribution functions $f_{ij}(t)$, $i, j = 1, \ldots, M$ must be known. $f_{ij}(t)$ is the distribution of the process values in case of a transition from state i to state j.

A simplification of the transition is achieved using a special case of a semi-Markov process (SSMP):

$$P\left(R_{t+1} = a, \sigma_{t+1} = j | R_k = a_k, \sigma_k = i_k, 1 \leq k \leq t\right) =$$
$$P\left(R_{t+1} = a, \sigma_{t+1} = j | \sigma_t = i_t\right) =$$
$$p_{i_t j} P\left(R_{t+1} = a | \sigma_t = i_t\right)$$
$$f_{i_t j}(t) = p_{i_t j} f_{i_t}(t)$$

Please note that every SMP with M states can always be described as an SSMP with M^2 states. To do so, we identify each transition from state i to state j of the SMP with a state (i, j) of the SSMP (cf. Haßlinger (1997)). Using special case SMPs, we are able to adjust the autocorrelation and the distribution of the process independently: The autocorrelation relies only on the state-dependent mean values (cf. Relation 6).

To make efficient analysis of the SMP feasible, we require the embedded Markov chain to be aperiodic and irreducible and the transition matrix to be diagonalizable.

In this case, we can denote the autocorrelation of the SMP as an exponential sum

$$\mathcal{A}_R(n) = \sum_{j=2}^{M} \alpha_j \lambda_j^n \tag{1}$$

with λ_j being the eigenvalues of the transition matrix (cf. Kempken and Luther (2006)). The first eigenvalue λ_1 of a stochastic transition matrix is always 1. The coefficients α_j can be derived from a Vandermonde-type equation system from a number of given autocorrelation values using a part of Prony's well-known method (cf. Hildebrand (1974)).
In the following, we denote the original data as G_j describing the size of the particular frame in bytes with j being its index.

PARAMETER ESTIMATION

We intend to derive the parameters of a semi-Markov model - that is, the transition probabilities and the state-dependent discrete distributions - in such a way that the distribution and autocorrelation of the process and the empirical data match as closely as possible. We employ an evolutionary optimization technique using genetic programming.

Evolutionary Optimization

This approach does not derive the parameters of a SSMP model from given data in a deterministic way but instead efficiently tests new parameters and checks which fit best. To do so, a *population* is defined. Each member of this population (*individual*) represents a possible parameter configuration, and *fitness* values are computed for each one. The members with a high fitness level are reproduced, *mutated* and combined with each other (*crossover*). They form the members of the next generation of the population. This depicts a simple but efficient optimization technique for a broad range of problems. In the following, we describe how we adapt this idea to our problem of finding the parameters of an SSMP model. Further details about genetic programming techniques can be found, for instance, in (Goldberg, 1989) or (Whitley, 1994).
The main problem in deriving SSMP parameters from a given empirical data series is the assignment of values to states of a Markov chain. If such an assignment is given, the transition probabilities, the stationary probabilities and the state-dependent distributions can be computed from the given data G_j by counting. Let t_{ij} denote the number of transitions from state i to state j; then the probabilities are estimated by

$$p_{ij} := \frac{t_{ij}}{\sum_{k=1}^{M} t_{ik}}. \tag{2}$$

The steady state probabilities may be determined by solving the following system of equations:

$$\sum_{k \in Z} p_k p_{kj} = p_j \text{ for all } j \in Z \tag{3}$$

$$\sum_{j \in Z} p_j = 1 \tag{4}$$

To model the state-dependent distributions, we look at the distributions of the actual values for each state according to the particular frame-state assignment.
If N denotes the number of values and M the number of states of a SSMP model, then there are M^N possible frame-state assignments. For a typical data series as examined in the following section, we have $N = 3100$ and $M = 4$. Hence, it is infeasible to test all combinations. However, we attempt to search this huge parameter space using genetic algorithms.
The total number of states M of the SSMP model to be determined is a parameter for the optimization process. In previous experiments (Kempken and Luther, 2007), we found that even a small number of states is sufficient to give a good approximation of the autocorrelation behavior of the empirical data set.
We code each possible assignment alternative as a sequence of the corresponding state indices. In the example mentioned, one scheme is therefore defined by 3100 values between 1 and 4. The idea is to optimize the assignment scheme so that the autocorrelation of an SSMP derived from it and the autocorrelation of the empirical data series prove the best match. The autocorrelation function of a data series $G_j, j \in \{1, \ldots, N\}, n \in \{0, \ldots, N-1\}$ is estimated by

$$\mathcal{A}_G(n) = \frac{\sum_{j=1}^{N-n} (G_j - E(G))(G_{j+n} - E(G))}{\sigma_G^2 (N-n)}. \tag{5}$$

The corresponding values for the SSMP model can be derived from the parameters of the model (cf. Kempken and Luther (2006)) or by using

$$\mathcal{A}_S(n) = \frac{\sum_{i=1}^{M} \sum_{j=1}^{M} p_i E_i(S) p_{ij}^{(n)} E_j(S) - E^2(S)}{\sigma_S^2}.$$

$$\mathcal{A}_S(0) = 1 \tag{6}$$

Here, $E(S)$ denotes the overall expectation value of the semi-Markov process S and $E_i(S)$ the expectation value if S is in state i, which is the state dependent mean value. The n-step transition and the stationary probabilities are given by $p_{ij}^{(n)}$ and p_i respectively.
To evaluate the quality of an assignment scheme and the corresponding SSMP model S of the empirical data series G, we use the following function:

$$g_G(S) := \frac{1}{\sum_{i=1}^{N} (\mathcal{A}_G(i) - \mathcal{A}_S(i))^2} \tag{7}$$

In terms of genetic programming, the intention is to find an individual (parameter configuration) that maximizes the above fitness function.
We initialize a population of individuals by randomly taking values from the range of the state indices. These random assignment schemes are to be evaluated and optimized iteratively. Each iteration consists of several steps:

1. For each individual, construct an SSMP model with M states according to the given assignment scheme.
2. Compute the fitness of each individual according to the rating function given in Relation 7.
3. Build up a new generation of individuals.
(a) The best individual - that is, the one with the highest fitness value - is taken into the next generation unchanged.
(b) A *mating pool* is constructed from a selection of the population. Each individual has a probability of being selected for the mating pool proportional to its fitness value.
(c) With given probabilities, some mutators are applied to the members of the mating pool (see below).
(d) Genetic material is exchanged between random members of the mating pool (crossover, see below).
(e) The changed and unchanged members of the mating pool form the next generation of the population.

Since we cannot decide when the optimal configuration has been reached, we could extend the optimization process infinitely. However, we abort the optimization when no further improvement in the maximum fitness value of the population has been achieved for a given number of generations. The parameters corresponding to the individual with the highest fitness value are returned as result of the genetic approach.

The genetic operators (mutation and crossover) and their probabilities are important to the optimization process. In our case, we used four types of mutation operators, each with a probability of 0.2:

- Swap: two random values in the sequence are exchanged.
- Reverse: a randomly chosen, continuous part of the sequence is reversed.
- Shuffle: a randomly chosen, continuous part of the sequence is shuffled.
- Block: a randomly chosen, continuous part of the sequence is replaced by a block of identical random values.

In our experiments, we apply the so-called 2-point crossover: a randomly chosen continuous part of the genetic sequence is replaced by a sequence taken from another individual with the same length and relative position.

An interesting enhancement of this approach is to start with a modified initial population: Instead of creating all individuals randomly from scratch, we can include the assignments according to other modeling approaches, e.g. the simple or scene-based modeling method proposed by Rose (1997). Because they are included in the sequence population and the best sequence is kept in every iteration step, the results yielded by the evolutionary optimization process are guaranteed to be of at least of the same quality. However, this is only feasible if the number of states of the approaches match.

In order to create a complete queue model, we have to compute the discrete state-specific distributions for the arrival process. The SSMP approach allows us to adapt the autocorrelation and the state-specific distributions separately, since the autocorrelation depends only on the mean values per state. Hence, we have to preserve the mean value when performing a discretization of the size distribution. To do so, we apply the following approach.

We denote the number of values S_k to be discretized into d equidistant steps by K and the difference between the discretization points as

$$\Delta = \frac{\max_{j\in 1,\ldots,N}(G_j)}{d-1}.$$

We yield d points $s_l = l\Delta, l = 0,\ldots,d-1$. We denote the discretized variable by S_Δ. The probability $P(S_\Delta = s_l)$ is influenced by each value that differs less than Δ from s_l:

$$P(S_\Delta = s_l) = \sum_{k=1;|S_k-s_l|<\Delta}^{K} \frac{\Delta - |S_k - s_l|}{K\Delta} \qquad (8)$$

Modeling the Periodic Behavior

The method presented above delivers a good approximation of the autocorrelation of the given empirical data. However, if we intend to reflect the periodic behavior resulting from the GoP coding scheme, a much higher number of states is needed. In order to do so, we propose some enhancements to the optimization technique presented.

The MPEG-4 video standard we considered requires a fixed limit for the maximum length of any group of pictures $\hat{G}$. This parameter has to be given or determined in advance along with an assignment of every single frame to an index number specifying the position of the particular frame within the current group of pictures. These values can be read directly from a given video file. The index numbers serve as state identifiers; thus a Markov chain may be constructed that reflects the periodic behavior. The number of states is given here by the maximum GoP length $\hat{G}$. However, this leads to a very rough approximation of the periodic behavior. In order to improve the model, we introduce an additional number of "sub-states" M available in every state of the Markov chain described. The new model therefore consists of $M \times \hat{G}$ states. Figure 1 gives an illustration of this process.

GoP sequence (fixed)	1	2	3	4	5	1	2	3	4	1
	&	&	&	&	&	&	&	&	&	&
sub-state sequence (subject to optimization)	1	1	2	2	1	3	1	2	2	1
	=	=	=	=	=	=	=	=	=	=
Resulting sequence (used for autocorrelation)	1	5	10	14	17	3	5	10	14	1

Fig. 1. Combination of GoP sequence and sub-state sequence

The sequence of sub-states is optimized by the evolutionary technique presented. In every iteration step, the sub-state index $(1\ldots M)$ and the corresponding GoP-index $(1\ldots\hat{G})$ are combined to a unique state number $(1\ldots M\hat{G})$. We compute the autocorrelation of the resulting process to evaluate the fitness of the sub-state sequence. The sequence of GoP-index states, however, remains the same in every step.

This way, we are able to fit the autocorrelation function of the model to the corresponding values of the data trace without losing the characteristic periodic behavior. However, this approach involves a high number of states and therefore requires a much higher computational effort. Whether the reflection of the periodic behavior is worth this additional expense or the approximation of the autocorrelation resulting from the basic approach presented previously is sufficient depends on the application of the model.

EXAMPLE AND COMPARISON

As an example, we consider an episode of the German video-blog "Ehrensenf"[2], which is available in different video formats including 3GP for mobile devices. The episode to be modeled was published on February 5, 2007 and includes 3102 frames at a rate of 15 frames per second, resulting in a length of about three and a half minutes. The audio is encoded using the AMR codec with a constant bitrate and therefore ignored in the following considerations. The video stream uses the MPEG-4 part 2 video codec, and the particular frame sizes range from 32 to 2226 Bytes. The maximum size of a GoP is 30 frames. Another example is the movie trailer for "The Da Vinci Code", which is available on the internet[3]. This clip lasts for 1510 frames at a rate of 11 frames per second, resulting in a length of 2 minutes and 20 seconds. The audio is also encoded using the AMR codec. The video stream, however, is encoded according to the H.263 standard with frame sizes ranging from 19 to 7297 bytes. The maximum size of a group of pictures is 12 frames.

To get the frame sizes from the files, we implemented a *rawfileparser* tool that reads these values from a variety of video files based on QuickTime (MOV / 3GP), Windows Media (ASF) and Video for Windows (AVI) container formats.

We applied both the basic modeling technique and the enhanced approach to reflect the periodic behavior. The optimization process was aborted in both cases when no improvement was found for 100 iterations. The basic approach creates a semi-Markov model with four states, while the enhanced approach requires $4 \cdot 30 = 120$ and $4 \cdot 12 = 48$ states, respectively. We considered the autocorrelation function on the interval from $\mathcal{A}(0)$ up to $\mathcal{A}(99)$. The remaining difference to the empirical autocorrelation

$$\sum_{i=0}^{99} |\mathcal{A}_S(i) - \mathcal{A}_G(i)|$$

is given in Table 1. The functions themselves are depicted in Figures 2 and 3.

We see that the evolutionary optimization approach proposed is able to model the general behavior of the autocorrelation function using a small number of states. In (Kempken and Luther, 2007), we successfully applied this technique to modeling the long-time autocorrelation of high definition video streams. Furthermore, with the proposed enhancements, we are able to reflect the periodic characteristics of the autocorrelation at the frame level.

However, the enhanced approach requires much more computational effort. In the examples considered, the basic approach typically requires about a minute to optimize the parameters. In contrast, the enhanced approach takes about six hours due to the increased state space. In each case, the test system has been a desktop PC with a 3 Ghz Pentium 4 processor.

Clip name	Basic	Enhanced
Ehrensenf	7.1275 (4)	7.0872 (120)
The Da Vinci Code	4.2941 (4)	2.1463 (48)

Table 1: Remaining error for autocorrelation on $\mathcal{A}(0) \ldots \mathcal{A}(99)$. The respective number of states is given in parentheses.

CONCLUSION AND FURTHER WORK

The analysis of communication networks strongly depends on appropriate models for the traffic considered. In this paper, we have proposed a new technique for modeling video traffic in mobile applications using discrete-time semi-Markov processes employing genetic programming. We also presented an enhancement to this technique that allows us to reflect the periodic behavior of the autocorrelation of frame sizes within GoPs - but at the expense of a much higher computational effort. We have applied both methods to modeling two exemplary empirical video traces.

Our methods allow the resulting semi-Markov model to reflect long-time correlations in a more accurate way than other techniques, as it has been pointed out in previous work (Kempken and Luther (2007)). Using the newly proposed enhancement to our approach, we are able to reflect the periodic behavior of the autocorrelation at the frame level accurately, but at the cost of a much higher number of states, and consequently a higher computational effort.

As further work, we intend to optimize the genetic programming approach proposed in this work and to apply it to other types of traffic as well. We plan to analyze the resulting queueing models using such tools as our *InterVerdiKom* (cf. Kempken et al. (2006)), which implements a number of efficient and reliable factorization methods for both transient and steady state workload analysis of semi-Markov queue models. This way, we intend to evaluate the actual impact of the increase in modeling accuracy of the approach proposed in this paper and to find out if the huge increase in computational effort required by the enhanced approach pays off.

ACKNOWLEDGEMENTS

This research was carried out in the recent project 'Traffic Modeling in Multiservice Networks and Analysis of Resource Requirements for Quality of Service Support' funded by the German Research Council (DFG). The authors would also like to thank Holger Pretzsch, who is

[2]http://www.ehrensenf.de

[3]http://www.free-3gp-video.com

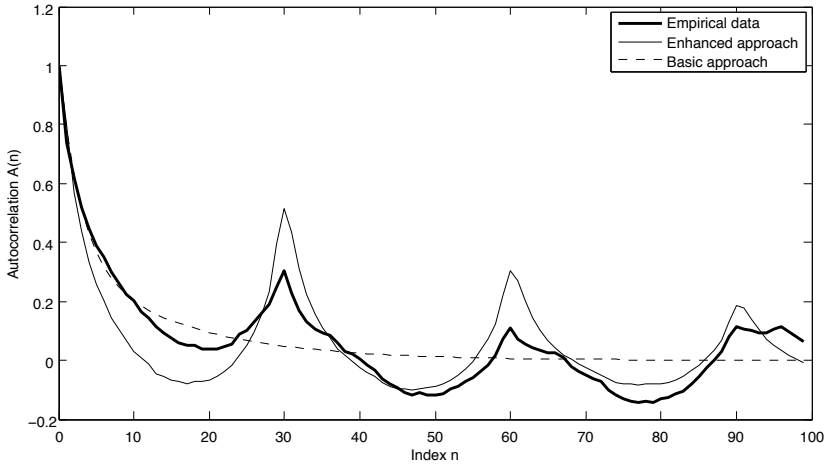


Fig. 2. Autocorrelation of "Ehrensenf" clip

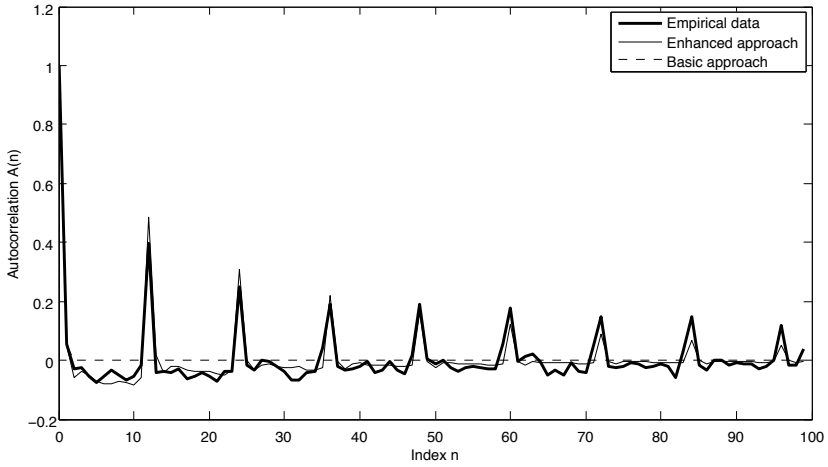


Fig. 3. Autocorrelation of "The Da Vinci Code" clip

student in computer science in Duisburg, for his work on the file parsing tool.

REFERENCES

Feldmann, A. and Whitt, W., 1998. "Fitting mixtures of exponentials to long-tail distributions to analyze network performance models". *Performance Evaluation* 31, 245–279.

Goldberg, D. E., 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, Boston, MA.

Haßlinger, G., 1997. "Semi-Markovian modelling and performance analysis of variable rate traffic in ATM networks". *Telecommunication Systems* 7, 281 – 298.

Haßlinger, G., 2000. "Waiting times, busy periods and output models of a server analyzed via Wiener-Hopf factorization". *Performance Evaluation* 40, 3–26.

Haßlinger, G. and Takes, P., 2003. "Real time video traffic characteristics and dimensioning regarding QoS demands". In "Proceedings of the 18th International Teletraffic Congress", 1211–1220.

Hildebrand, F. B., 1974. *Introduction to numerical analysis*. McGraw-Hill Book Co., New York.

Kempken, S. and Luther, W., 2006. "Verified methods in stochastic traffic modeling". In "Reliable Implementation of Real Number Algorithms: Theory and Practice", LNCS. Springer, to appear.

Kempken, S. and Luther, W., 2007. "Modeling of H.264 high definition video traffic using discrete-time semi-Markov processes". In "Proceedings of the 20th International Teletraffic Congress", to appear.

Kempken, S.; Luther, W. and Haßlinger, G., 2006. "A tool for verified analysis of transient and steady states of queues". In "Proceedings of the First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS '06)", electronic resource.

Kleinrock, L., 1975. *Queueing Systems*, vol. 1/2. Wiley.

Krunz, M. M., 2000. "The correlation structure for a class of scene-based video models". *IEEE Transactions on Multimedia* 2, no. 1, 27 – 36.

Latouche, G. and Ramasvami, V., 1999. *Introduction to matrix analytic methods in stochastic modeling*. ASA-SIAM.

Li, S.-q., 1991. "A General Solution technique for Discrete Queueing Analysis of Multi-Media Traffic on ATM". *IEEE Transactions on Communication* 39, 1115–1132.

Luther, W. and Traczinski, D., 2006. "Reliable Computation of Workload Distributions using Semi-Markov Processes". In "Proceedings of the 13th International Conference on Analytical and Stochastic Modelling Techniques and Applications", 111–116.

Pereira, F., 2002. *The MPEG-4 Book*. Prentice Hall PTR.

Rose, O., 1997. "Simple and efficient models for variable bit rate MPEG video traffic". *Performance Evaluation* 30, 69 – 85.

Salvador, P.; Valadas, R. and Pacheco, A., 2003. "Multiscale Fitting procedure using Markov Modulated Poisson Processes". *Telecommunications Systems* 23, no. 1-2, 123–148.

Traczinski, D.; Luther, W. and Haßlinger, G., 2005. "Polynomial Factorization for Servers with Semi-Markovian Workload: Performance and Numerical Aspects of a Verified Solution Technique". *Stochastic Models* 21, 643–668.

Whitley, D., 1994. "A genetic algorithm tutorial". *Statistics and Computing* 4, no. 2, 65–85.

SESSION 2

Network Traffic and Protocols

PERFORMANCE-RELEVANT NETWORK TRAFFIC CORRELATION

Hans-Peter Schwefel
Center for Teleinfrastruktur
Aalborg University
email: hps@kom.aau.dk

Imad Antonios
Dept. of Computer Science
Southern Connecticut State Univ.
email: antoniosi1@southernct.edu

Lester Lipsky
Dept. of Comp. Sci. & Eng.
University of Connecticut
email: lester@engr.uconn.edu

ABSTRACT

Correlation structure is an important metric to consider when modeling the performance of network traffic. Particularly, the presence of long-range dependence (LRD) in the input process may, under some circumstances, lead to poor queueing performance. In this paper, we first aim to characterize the conditions under which the presence of LRD is performance relevant. We define variations of an ON/OFF-type process that employ a truncated power-tail (TPT) distribution, and analyze their correlation structure in relation to queueing performance. Our analytic results show that the correlation structure in some cases is very sensitive to the presence of a background Poisson process, and that while other model variations exhibit LRD, it is only those with TPT-distributed ON times that queueing performance is poor. These results lead us to propose a procedure for extracting performance-relevant correlation properties, whose effectiveness we demonstrate via simulation experiments using synthetic and measured traffic.

Keywords: ON/OFF models, long-range dependence, queueing models, autocorrelation.

INTRODUCTION AND MOTIVATION

Innumerable studies of Internet traffic have shown that it exhibits *self-similarity* and *long-range dependent* (LRD) properties (see [Leland et al. 1994]), meaning that it is highly varying, and bursty over a wide range of time scales. This poses challenges in understanding the factors underlying it, and thus makes it difficult to develop models to predict network performance. Within the large body of research on the subject, a bulk of the work has concentrated on constructing statistical models that reproduce the correlation structure of the measured traffic [Melamed 1991, Norros 1995], and studying the effects of generated traffic on queueing performance behavior via simulation. The main conclusion was that the presence of LRD in traffic carries important implications for network performance. Another body of research was concerned with uncovering the physical factors that underlie self-similarity and LRD [Crovella and Bestavros 1996, Veres and Boda 2001]. The factors identified include file-size distribution, file transmission time distribution, user think times, and TCP congestion control mechanism, which were used to construct analytic performance models with matching statistical properties of measured traffic. One of the important long-range dependent traffic models is the aggregation of ON/OFF sources with heavy-tailed ON periods [Crovella and Bestavros 1996, Willinger et al. 1995].

Since LRD properties have been recognized not to be sufficient predictors of queueing performance, it is important to characterize the conditions under which their presence has a significant bearing on network performance, which is the goal of this paper. The starting point of our inquiry is the definition of a rich analytic traffic model of ON/OFF-type sources with four variations whose correlation structure we examine. In particular, we consider the autocorrelation structure of packet inter-arrival times as well as the counting process associated with the model variations. To account for settings where non-LRD traffic is aggregated with LRD traffic, the model variations allow for a background Poisson process.

Summary of results: Our results of the coefficient of autocorrelation for the inter-packet times show that not all models exhibit LRD properties, whereas for the counting process LRD properties are present in all model variations with TPTs. Our calculations also indicate that the correlation structure is highly sensitive to the superposition of a background Poisson process only for inter-packet times, but not for the counting process. Our analytic results for the queueing behavior reveal that some models with LRD in their inter-packet times

or counting process do not show significant delays – it is only the models in which TPTs are used for the ON periods that poor queueing behavior, as measured by average queue lengths, can be observed for utilizations well below 1. Power tails in the OFF periods only become performance relevant when the utilization is close to saturation. The mere existence of LRD is hence not a performance-relevant aspect. This conclusion motivated the investigation of an alternative correlation metric that is based on a filtered counting process. An exploration of the proposed measure using both synthetically generated and measured traffic reveals its effectiveness in destroying the correlation structure for traces whose associated queueing performance is not poor, thus making it a more reliable predictor of performance than the presence of LRD properties.

A SIMPLE ANALYTIC ON/OFF MODEL

ON/OFF models have been widely used for modeling of bursty traffic for the last 20 years [Heffes 1980]. After the identification of long-range dependent phenomena in network traffic [Leland et al. 1994], such models have been extended to include heavy-tail distributions [Dumas and Simonian 2000] or their truncated counterparts [Schwefel and Lipsky 1999]. The latter reference introduces an aggregated ON/OFF model with general Matrix-Exponential (ME) ON periods (see Appendix or [Lipsky 1992] for a detailed treatment of ME distributions), represented by a complex, but structured Markov Modulated Poisson Process (MMPP). For the purpose of this paper, we will only present the structure of single-source ON/OFF models. These, however, can be generalized to describe the aggregation of traffic from multiple sources by expressing their structure as the product of the state spaces of single-source models as shown in [Schwefel 2000].

The model variations considered make use of truncated power-tail distributions introduced in [Greiner et al. 2000] (see Appendix for brief description); when these describe the ON periods only, the model is referred to as TPT-ON. If it is for the OFF periods only the model is TPT-OFF, and when both the ON and OFF periods are TPT distributed the model is TPT-ALL. As a case for comparison, a fully exponential ON/OFF model (EXP) is also used. It is important to note that for the range of TPT distributions considered, the results would be indistinguishable from using a full power-tail for finite sample experiments, whether these are based on measurement or simulation.

All model variations are special cases of the ME-ALL model as presented in the following. The ME-ALL model is an MMPP model which generates inter-packet times from a single ON/OFF source, with peak-rate λ_p and long-term average rate $\lambda = \lambda_p * E\{ON\}/(E\{ON+OFF\})$ aggregated with some background Poisson process with rate λ_{bg}. The ME-ALL model uses Matrix-Exponential OFF periods, $< \mathbf{p}_{off}, \mathbf{B}_{off} >$, and Matrix-Exponential ON periods with representation $< \mathbf{p}_{on}, \mathbf{B}_{on} >$. The modulating Markov process then has the following generator matrix $\boldsymbol{\mathcal{Q}}$, and the corresponding Poisson packet rates on the diagonal of $\boldsymbol{\mathcal{L}}$:

$$\boldsymbol{\mathcal{Q}} = \left[\begin{array}{c|c} -\mathbf{B}_{off} & \mathbf{B}_{off}\boldsymbol{\varepsilon}'_{off}\,\mathbf{p}_{on} \\ \hline \mathbf{B}_{on}\boldsymbol{\varepsilon}'_{on}\mathbf{p}_{off} & -\mathbf{B}_{on} \end{array}\right],$$

$$\boldsymbol{\mathcal{L}} = \left[\begin{array}{cc|c} 0 & & \\ & \ddots & \\ & 0 & \\ \hline & & \lambda_p\mathbf{I} \end{array}\right] + \lambda_{bg}\boldsymbol{\mathcal{I}}\,.$$

Note that the individual blocks may have different dimensions. $\boldsymbol{\varepsilon}'$ is a column vector with all elements equal to 1 of the corresponding dimension, and $\boldsymbol{\mathcal{I}}$ is the unit matrix of dimension $dim(B_{on})+dim(B_{off})$. The ME-ALL process has a mean inter-packet time of $E\{X\} = 1/(\lambda + \lambda_{bg})$.

An ME-ALL/M/1 queue is a special case of an MMPP/M/1 queue and can be represented as a Quasi-Birth-Death Process with the standard matrix-geometric solution [Latouche and Ramaswami 1993]. LRD properties manifest themselves in the correlation structure of a stochastic process, which can be computed for the inter-packet times from an MMPP using standard methods, see [Meier-Hellstern and Fischer 1992].

In addition to the inter-packet time process (X_i), the counting process can also be used to describe traffic. In this paper, we use the interval-based version of the counting process, where $N_i(\Delta)$ is the random variable representing the number of packet arrivals in interval $[i\Delta, (i+1)\Delta]$. The covariance of the counting process $N_i(\Delta)$ of an MMPP for an interval size $\Delta > 0$ is obtained in [Neuts 1989].

The TPT-ALL model uses truncated power-tail distributions for both the ON period and the OFF period. The use of an exponential distribution $< p_{OFF} = 1, B_{off} = 1/Z >$ for the OFF periods results in the TPT-ON model, and the use of an equivalent exponential distribution in the ON periods instead causes the model to reduce to the TPT-OFF model. When both ON and OFF periods are exponential, the model reduces to the classical 2-state ON/OFF model [Heffes 1980].

CORRELATION AND QUEUEING PERFORMANCE

Common approaches in traffic modeling typically attempt to fit the first two moments of the traffic trace; capturing the correct mean is essential, since it determines the utilization of any performance model, which in turn can have a critical impact on stationarity. Fitting the variance is fully justified for service time distributions, since the basic performance metric of average queue length in a simple queueing model such as M/G/1 is directly related to the variance of the service time. On the other hand, the variance of an arrival process is of little impact on performance behavior in realistic utilization ranges well below 1. Of primary interest are long-range dependent properties [Melamed 1991, Norros 1995], which have more recently been recognized as being critical in guiding the development of traffic models.

Correlation Structure and Long-Range Dependence

We consider the correlation structure of the four model variations both in terms of their inter-packet time and counting processes. While studies on traffic LRD have mostly focused on the latter, using the inter-packet time process is primarily motivated by its immunity to potential sampling problems caused by the additional counting process parameter of interval size.

We first examine the case for inter-packet times. Figure 1 shows the results of analytic computations of the autocorrelation coefficients when a background Poisson stream is merged with the output of each of the model variations. In all models, the arrival rate of the ON/OFF model is $\lambda = 1$, and the contribution of the background process is the additional rate $\lambda_{bg} = 0.1$. Except for EXP, in which short-range dependence (SRD) can be observed, all models exhibit LRD properties. Looking at the TPT-OFF model, we find that the magnitude of the autocorrelation coefficients is greater than that for TPT-ON. This is surprising since the TPT-OFF model without the presence of the background Poisson traffic is a renewal process: the samples are either created by a Poisson process of rate λ_p (during the ON period), or by a convolution of an exponential residual time of the ON period together with a TPT-distributed ON period and another exponentially distributed time until first packet in the ON period. The absence of background traffic also renders EXP a renewal process, while for TPT-ALL and TPT-ON LRD properties are still present. These results indicate that only when the ON period is TPT distributed that LRD properties appear for inter-packet times.

Additional experiments with very small values for λ_{bg} in the order of 10^{-5} and smaller show that even

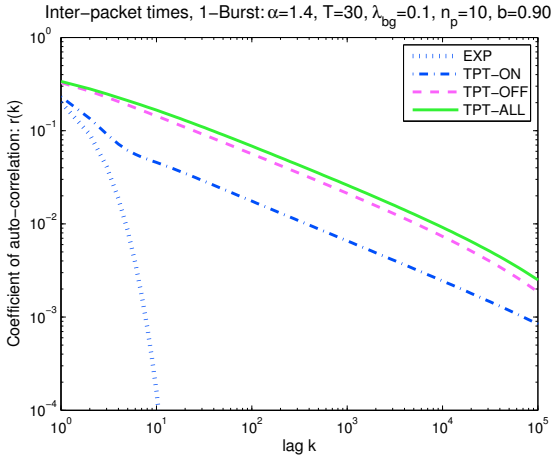


Figure 1: **Correlation structure of the inter-packet times with 10% background Poisson traffic:** The TPT-OFF model shows positive correlation only in the presence of background Poisson traffic. The TPT-ALL and TPT-ON models show LRD in both scenarios. Curves without background traffic are not shown since two of them, EXP and TPT-ALL, are renewal.

such small background Poisson rates are sufficient to keep the LRD properties of the inter-packet times for the TPT-OFF model. Hence, any small overlay process will make these models appear to be LRD.

The results of the computation of the autocorrelation coefficient for the counting process are shown in Figure 2. The TPT-ALL, TPT-ON and TPT-OFF models all exhibit LRD properties, and the magnitude of the coefficients is in decreasing order for the respective model. The EXP model only exhibits short-range dependence. As opposed to the computations for the inter-packet times, the background Poisson stream has virtually no effect on the correlation structure.

The behavior of the different processes is summarized in the following table (LRD = long-range dependence, SR = short-range correlation, 0 = renewal process):

model / scenario	$\lambda_{bg} = 0$	$\lambda_{bg} \to 0$	$\lambda_{bg} > 0$
inter-packet times			
EXP	0	0	SR
TPT-OFF	0	LRD	LRD
TPT-ON	LRD	LRD	LRD
TPT-ALL	LRD	LRD	LRD
counting process			
EXP	SR	SR	SR
TPT-OFF	LRD	LRD	LRD
TPT-ON	LRD	LRD	LRD
TPT-ALL	LRD	LRD	LRD

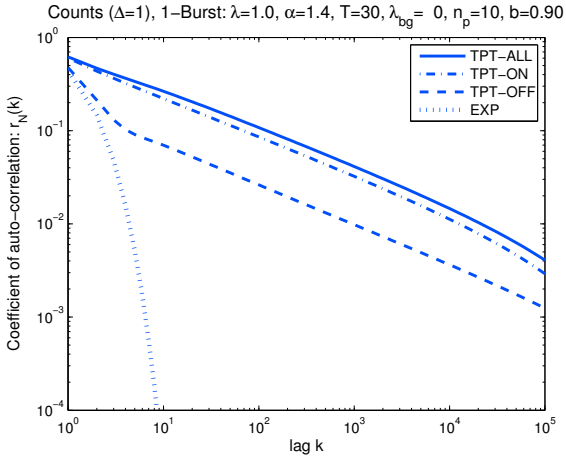


Figure 2: **Correlation structure of the counting process for $\Delta = 1$:** All three models that use TPT distributions show LRD properties. The background Poisson traffic has negligible impact on the correlation structure: the curves for $\lambda_{bg} = 0.1$ are visually indistinguishable and hence not shown here.

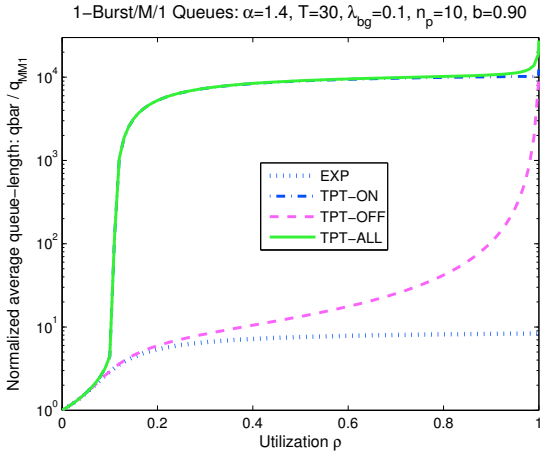


Figure 3: **Queueing behavior of different ON-OFF/M/1 queues with 10% background traffic:** Shown is the average queue length normalized to the mean queue length $\rho/(1-\rho)$ of an M/M/1 queue. The TPT-ON and TPT-ALL model show a blow-up around $\rho = 0.1$, while the TPT-OFF and EXP model show rather low queue lengths for typical network utilization values $\rho < 0.8$. The TPT distributions in the OFF periods only becomes relevant for $\rho \to 1$. The existence of background Poisson traffic does not have a relevant influence here, despite its strong influence on correlation structure of the inter-packet times.

Queueing Performance

Since all four models, TPT-ALL, TPT-ON, TPT-OFF, and EXP have an MMPP representation, analytic results for the queue length distribution and the average queue length when using these processes as arrival to an exponential server can be obtained via known matrix-algebraic methods [Latouche and Ramaswami 1993, Neuts 1981]. In this section, we analyze the different performance behavior as measured by average queue length for different server rates ν, corresponding to different utilizations $\rho = (\lambda + \lambda_{bg})/\nu$.

As Figure 3 shows, three different ranges of utilization values need to be distinguished:

1. For low values of $\rho < 0.1$ (in the given parameter setting), all four models show the same, low average queue length.

2. At $\rho = 0.1$ the normalized average queue length of the TPT-ALL and TPT-ON model increases rapidly. These so-called blow-up points were already identified for the TPT-ON model in [Schwefel and Lipsky 1999]. They occur when the packet-arrival rate during ON periods is higher than the service rate, namely when $\lambda_p + \lambda_{bg} > \nu$ or equivalently $\rho > (\lambda+\lambda_{bg})/(\lambda_p+\lambda_{bg})$. The TPT-OFF and the EXP model do not show these blow-up points.

3. For high utilization $\rho \to 1$, TPT-ON and EXP show a horizontal asymptote of the normalized mean queue length, indicating growth according to $(1-\rho)^{-1}$. The TPT-OFF and TPT-ALL models however show blow-ups in their average queue lengths that grow faster than the M/M/1 queue when $\rho \to 1$. This growth can be explained by the analysis of GI/M/1 queues with GI=TPT as discussed in [Greiner et al. 2000].

Note that the existence of background Poisson traffic, despite its potentially large influence on correlation structure of the inter-packet times, has only limited impact on the queueing behavior. In the case where there is no background Poisson traffic, performance is nearly indistinguishable from that shown in Figure 3. Large amounts of background traffic may however shift the blow-up points of the TPT-ON and TPT-ALL model as quantitatively described in Item (2) above.

In summary, the analytic performance results in terms of average queue lengths clearly show that although the TPT-OFF model with some background Poisson traffic exhibits long-range dependence with a large tail-exponent (large absolute values of the autocorrelation curve) and extremely high variance, its queueing performance is rather well behaved. It is only for high utilization values $\rho \to 1$ that this model causes high queue lengths. The TPT-ON and TPT-ALL models on the other hand are nearly equivalent in terms of queueing behavior in that power tails in the ON periods are the performance dominating element and lead to these blow-up points when the utilization is well below 1.

A PERFORMANCE-RELEVANT CORRELATION METRIC

The previous sections show that autocorrelation of the inter-packet time process is in any case not a good indicator of potential poor queue performance, since the presence of LRD in this process is rather sensitive to the existence of a background Poisson process or any other overlapping process. Autocorrelation of the counting process in turn is less sensitive to such disturbance by a background process, but it still does not provide a performance-relevant metric, since either ON periods or OFF periods could cause such LRD if heavy-tail distributions are present, while only heavy-tailed ON periods may cause poor performance behavior for utilizations well below 100%. The reason for the poor performance, whether it is measured by average queue length/delay or by buffer overflow probability, is the existence of heavy-tailed over-saturation periods. Therefore, it is only the periods of high traffic that are performance critical, even though long-range dependence can also be created by subsequent intervals of heavy-tailed length with low traffic.

Measuring the empiric distribution of the duration of over-saturation periods directly may however not show heavy-tailed distributions due to potential sensitivity toward small-scale traffic variations. The advantage of the correlation properties is the robustness toward such small scale variations. As a consequence, this section proposes and evaluates an approach that is based on the autocorrelation properties of a filtered counting process, such that the impact of periods of low traffic is removed.

The Filtering Method

The approach works as follows: Any maximum-length sequence of subsequent values of the counting process with low traffic intensity $N_i(\Delta)$,...,$N_{i+k}(\Delta) < N^*$ is replaced by a single value, for which we use the average of this sequence. In pseudocode this filtering of the measured counts $N_i(\Delta)$ into its counterpart $M_j(\Delta)$ can be expressed as:

Algorithm 0.1: FILTER($N[], M[], N^*$)

$j \leftarrow 1; i \leftarrow 1;$
while $i <= length(N)$
do
- **if** $N[i] < N^*$
  - **then**
    - $M[j] = N[i]; j \leftarrow j + 1;$
    - $i \leftarrow i + 1;$
  - **else**
    - $k \leftarrow 1$
    - **while** $N[i + k] < N^*$
      - **do** $k \leftarrow k + 1$
    - $M[j] = average(N[i], ..., N[i + k - 1])$
    - $j \leftarrow j + 1;$
    - $i \leftarrow i + k$

Choice of threshold: An important parameter in the algorithm above is the threshold N^*, which determines whether the interval is identified as 'low traffic' or 'high traffic'. In the general case, when only the traffic measurements are available, this value is suggested to be set to $N^* = \lambda\Delta = E(N)$, the expected value of the number of arrivals in an interval of size Δ. In case that it is known that the traffic is being fed into a network with bottleneck capacity ν, the intuitively obvious choice of the threshold is $N^*_\nu = \nu\Delta$, which in stable scenarios, namely when $\lambda < \nu$, results in a larger threshold.

Instead of investigating LRD properties of the counts N_i, the LRD properties of the filtered (and somewhat shorter, except for pathological cases) sequence M_j should be investigated, in which LRD properties as caused by periods of low traffic have been removed.

The filtering approach above can alternatively be applied to the inter-arrival time process. In this case, periods of low traffic correspond to a sequence of relatively large inter-packet times, those that exceed $1/\lambda$, or $1/\nu$ if ν is known. When applied to inter-packet times, the algorithm uses the first value as the threshold and replaces sequences of inter-packet times exceeding it with a single value being the threshold.

Application to Synthetic Traces

Figure 4 demonstrates the effectiveness of this approach. The figure shows the analytically computed autocorrelation function of the counting process of both the TPT-ON and TPT-OFF models (dashed lines), together with the empiric correlation function for the original simulated counts (solid line) and the corresponding filtered counts (dotted). Figure 4 is plotted on a log-log scale to reveal the LRD properties, while the linear scale in Figure 5 accentuates the effect of the filtered count on the correlation structure. These results show that the correlation structure of the filtered counts M_j only exhibits LRD properties for models with TPT-distributed ON periods, exactly those with poor performance behavior.

Application to Traffic Measurements

In this section, we aim to further support the effectiveness of the filtering approach in identifying performance-relevant correlation by considering traffic traces and associating them to simulated traffic. We first computed the autocorrelation function for the interarrival process of the Bellcore trace (BC-pAug89) used in [Leland et al. 1994] and its filtered version. The correlation structure of the filtered process, as shown in Figure 6, strongly exhibits LRD, denoting its relevance for queueing performance. It even shows a much clearer straight line behavior in

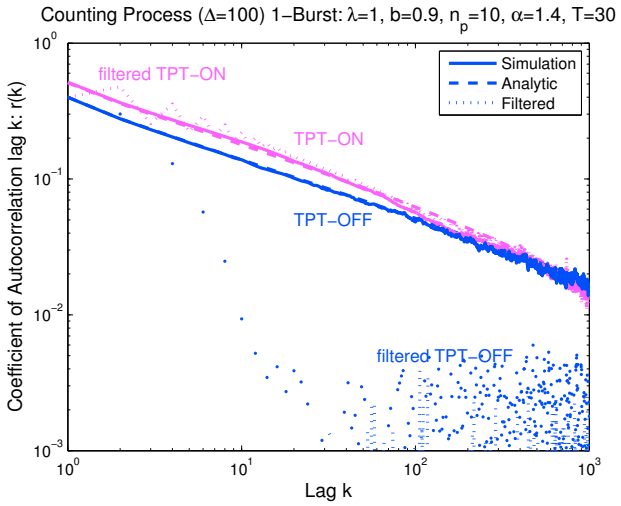


Figure 4: Autocorrelation coefficient computations and simulations for the counting process of a 1-Burst model and its processed variant (log-log scale): LRD properties in the TPT-OFF model are destroyed by the pre-processing, but not in the TPT-ON model.

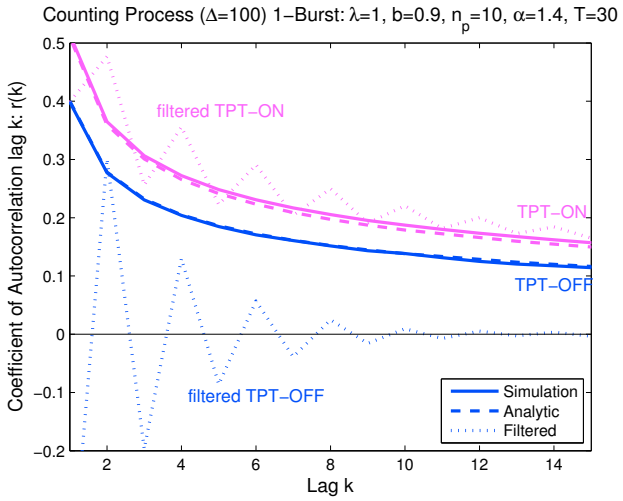


Figure 5: Same as Fig. 4 but on a linear scale to accentuate the effect of the filtering process on each of the models.

the log-log plot, indicating the presence of performance relevant correlation. This is confirmed by the results of a trace-driven queueing simulation for this measurement shown in the curve marked by squares in Figure 7. The figure shows the mean queueing delay as a function of utilization assuming exponential service times. The queueing delay caused by such a trace is comparable to that of an N-Burst simulation with 5 sources and TPT-distributed ON durations (marked by crosses). Note that the queueing delay is substantially larger (by a factor of 100 or more) than in a comparable simulation of Poisson traffic for utilizations above 50%. The sample size for the simulated data was chosen to be the same as the trace, $n = 1e6$.

Based on these results, it can be stipulated that the Bellcore data represents an example of a dataset with LRD caused by ON-OFF like behavior with power-tailed ON periods, which is consistent with the poor queueing behavior. In fact, without having done any detailed parameter fitting, the queueing behavior closely matched that of a TPT-ON model with a small number of sources.

A second set of packet data is also shown in Figure 7, namely a DEC trace of UDP packets (dec-pkt-4) available from [Danzig et al.]. The performance behavior of that trace as obtained in the trace-driven simulation (marked by circles) is very close to an M/M/1 queue, with some stronger deviations for ρ above 90%. Since the measurement originates from backbone traffic, an attempt to model it with a large number of TPT-ON sources, here $N = 500$ (marked by stars in Figure 7), yields remarkably close resemblance to the queueing delays of the UDP trace.

Since the method to identify the performance-relevant correlation is designed for traffic generated from a small number of sources (N in the order of $1, ..., 20$), it is well suited for analyzing traffic with a low degree of multiplexing (e.g. access networks). As we have shown in Figure 7 and previously in [Schwefel and Lipsky 1999], it is for such traffic that the performance impact of LRD is most detrimental.

SUMMARY AND CONCLUSIONS

In this paper, we examined the autocorrelation structure of a variation of analytical ON/OFF-type models that employ TPT distributions. Our computations indicate that the presence of a background Poisson process has a very strong impact on the coefficient of correlation of the inter-packet times for the TPT-OFF model. For the counting process of these models, the autocorrelation structure was found to exhibit LRD properties for the TPT-ON, TPT-ALL, and TPT-OFF models. The autocorrelation coefficients here are however rather insensitive to the background process. A queueing analysis of the traffic models revealed that only those with an ON period that is TPT distributed, namely TPT-ON and TPT-ALL, suffer from poor performance for a realistic utilization range where $\rho < 0.8$. We proposed and explored an alternative correlation metric that extracted performance-relevant aspects of the counting process. Through simulation experiments based on both measured and synthetic traffic traces, we demonstrated the effectiveness of this approach in destroying the correlation structure of models whose associated queueing performance are not poor, making it a more reliable metric than the mere recognition of LRD.

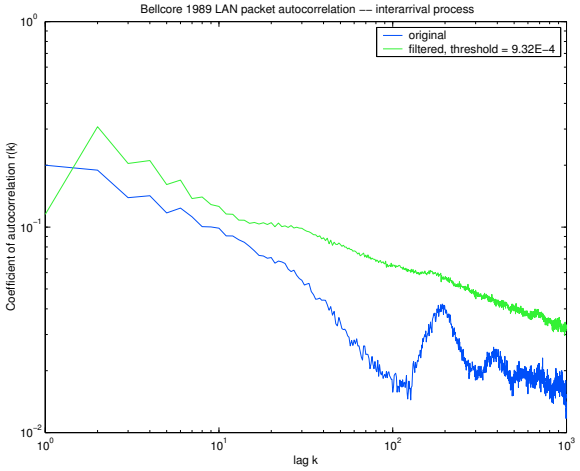


Figure 6: **Autocorrelation structure of actual and filtered Bellcore Ethernet packet trace (BC-pAug89) traffic:** The filtered variant clearly reveals performance-relevant correlation.

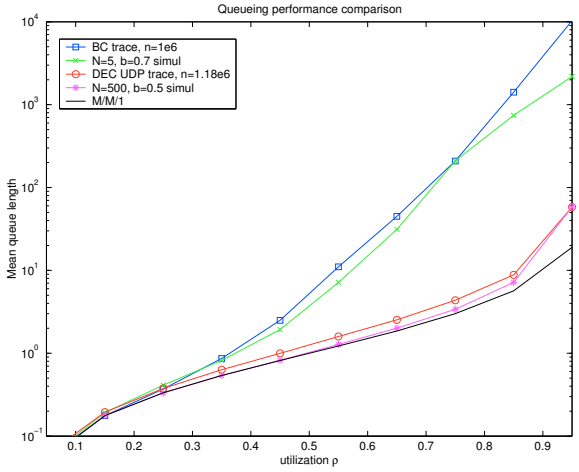


Figure 7: **Queueing performance of measured and synthetic traffic:** The queueing performance of the Bellcore trace (BC-pAug89) is comparable to that of an N-Burst model with $N = 5$, while the queueing behavior corresponding to the UDP packets from the dec-pkt-4 trace closely matches that of highly multiplexed synthetic traffic ($N = 500$) and the M/M/1 queue.

ACKNOWLEDGMENTS

The authors would like to acknowledge Robert Sheehan at the University of Connecticut for his helpful comments that stimulated the development of the filtering process.

APPENDIX

Matrix Exponential (ME) Distributions: Using the notation of [Lipsky 1992], the vector-matrix pair $< \mathbf{p}, \mathbf{B} >$ represents a ME distribution with reliability function $R(x)$, and density function $f(x)$ in the following way:

$$\begin{aligned} R(x) &= \mathbf{p}\exp(-x\mathbf{B})\boldsymbol{\epsilon}', \\ f(x) &= -\frac{R(x)}{dx} = \mathbf{pB}\exp(-x\mathbf{B})\boldsymbol{\epsilon}' \end{aligned}$$

where $\boldsymbol{\epsilon}'$ is a column-vector with all components being 1.

The moments of the distribution come out by integration:

$$\mathbb{E}\left\{X^k\right\} = k!\,\mathbf{p}\,\mathbf{B}^{-k}\boldsymbol{\epsilon}' \qquad (1)$$

Truncated Power-Tail (TPT) Distributions: The T-phase ME representation of a TPT distribution with exponent α is given by [Greiner et al. 2000]: let

$$0 < \theta < 1, \quad \text{and} \quad \gamma := \left(\frac{1}{\theta}\right)^{1/\alpha} > 1\,\cdot$$

Then, let $< \mathbf{p_T}, \mathbf{B_T} >$ be the ME representation of a T-phase Hyperexponential distribution with

$$\mathbf{p}_T = \frac{1-\theta}{1-\theta^T}[1,\, \theta,\, \theta^2,\, \cdots .\theta^{T-1}], \quad \text{and}$$

$$\mathbf{B}_T = \mu\ \mathbf{diag}\left(1, \gamma^{-1}, ..., \gamma^{-(T-1)}\right)\,\cdot$$

The parameter, μ, is a positive constant that can be chosen to set the mean of the distribution according to Eq. (2).

The expected value follows directly from Eq. (1)

$$\mathrm{E}(X_T) = \mathbf{p_T}\,\mathbf{B_T}^{-1}\,\boldsymbol{\epsilon}'_{\mathbf{T}} = \frac{1}{\mu}\,\frac{1-\theta}{1-\theta^T}\,\frac{1-(\gamma\theta)^T}{1-\gamma\theta} \qquad (2)$$

The so-called power-tail range of the TPT distribution is given in [Schwefel and Lipsky 1999]:

$$\mathrm{Rng}\,(\mathbf{B}_T) = \frac{\gamma^{T-1}}{\mu} =: \overline{x_T}\,. \qquad (3)$$

The following table shows the ratio $\mathrm{Rng}\,(()\,\mathbf{B}_T)/E(X)$ for $\alpha = 1.4$ and $\theta = 0.5$ and for values of T, as used in the main part of the paper:

T	1	10	20
$\mathrm{Rng}\,(\mathbf{B}_T)$ /E(X)	1	35.9	$4.46 \cdot 10^3$
...	30	40	50
...	$6.20 \cdot 10^5$	$8.74 \cdot 10^7$	$1.23 \cdot 10^{10}$

References

[Crovella and Bestavros 1996] Mark Crovella and Azer Bestavros: *Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes*. Proceedings of the ACM Sigmetrics, 1996.

[Danzig et al.] P. Danzig, J. Mogul, V. Paxson, and M. Schwarz: *The Internet Traffic Archive, <http://ita.ee.lbl.gov/>*

[Dumas and Simonian 2000] V. Dumas and A. Simonian: *Asymptotic bounds for the fluid queue fed by sub-exponential ON/OFF sources.* Advances in Applied Probability **32**, No 1. March 2000.

[Greiner et al. 2000] Michael Greiner, Manfred Jobmann, and Lester Lipsky: *The Importance of Power-tail Distributions for Telecommunication Traffic Models.* Operations Research **47**, No. 2, pp 313-326, March 1999.

[Heffes 1980] H. Heffes: *A class of data traffic processes - covariance function characterization and related queueing results.* Bell Syst. Tech. J. 59, pp. 897-930. 1980.

[Latouche and Ramaswami 1993] Guy Latouche and V. Ramaswami: *A Logarithmic Reduction Algorithm for Quasi-Birth-Death Processes.* Journal of Applied Probability **30**, pp. 650-674, 1993.

[Leland et al. 1994] Will Leland, Murad Taqqu, Walter Willinger, and Daniel V. Wilson: *On the Self-Similar Nature of Ethernet Traffic (Extended Version).* Proc. of IEEE/ACM Trans. on Networking, **2**, 1. February 1994.

[Lipsky 1992] Lester Lipsky: *QUEUEING THEORY: A Linear Algebraic Approach.* MacMillan Publishing Company, New York, 1992. 2nd edition to appear in Springer.

[Meier-Hellstern and Fischer 1992] Kathy Meier-Hellstern and Wolfgang Fischer: *MMPP Cookbook.* Performance Evaluation **18**, p. 149-171. 1992.

[Melamed 1991] B. Melamed: *TES: A class of methods for generating autocorrelated uniform variates.* ORSA Journal on Computing **3**, no. 4, pp. 317-329. 1991.

[Neuts 1981] Marcel Neuts: *MATRIX-GEOMETRIC SOLUTIONS IN STOCHASTIC MODELS.* John Hopkins University Press, London, 1981.

[Neuts 1989] Marcel Neuts: *STRUCTURED STOCHASTIC MATRICES OF M/G/1 TYPE AND THEIR APPLICATION.* Dekker, New York, 1989.

[Norros 1995] I. Norros: *On the Use of Fractional Brownian Motion in the Theory of Connectionless Networks.* IEEE JSAC **13**, no. 6, pp. 953-962. 1995.

[Schwefel and Lipsky 1999] Hans-Peter Schwefel and Lester Lipsky: *Impact of aggregated, self-similar ON/OFF traffic on delay in stationary queueing models (extended version).* Performance Evaluation 43, pp. 203-221, 2001.

[Schwefel 2000] Hans-Peter Schwefel, *Performance Analysis of Intermediate Systems Serving Aggregated ON/OFF Traffic with Long-Range Dependent Properties,* PhD Dissertation, Technische Universität München, 2000.

[Veres and Boda 2001] Andras Veres, and Miklós Boda, *The Chaotic Nature of TCP Congestion Control.* Proceedings of the ACM Sigcomm, 2001.

[Willinger et al. 1995] Walter Willinger, Murad Taqqu, Robert Sherman, and Daniel Wilson: *Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level (Extended Version).* Proceedings of the ACM Sigcomm, 1995.

AUTHOR BIOGRAPHIES

HANS-PETER SCHWEFEL is an Associate Professor and head of the IP Labs at Aalborg University, Denmark. His research focuses on IP based wireless networks with main interest in performance and dependability aspects. Before he joined Aalborg University, he was a project manager at Siemens Information and Communication Mobile, supervising research projects and responsible for the development of technical concepts for next generation mobile networks. He obtained his doctoral degree in the area of IP traffic and performance modeling from the Technical University in Munich, Germany. For his research activities, he also spent extended periods of time at the University of Florence, Italy, the University of Connecticut, USA, and at ATT Labs, USA.

IMAD ANTONIOS received his Ph.D. in Computer Science from the University of Connecticut under the guidance of Lester Lipsky. In 2003 he joined the Department of Computer Science at Southern Connecticut State University, USA, as Assistant Professor. His research interests are in the performance and dependability modeling of systems.

LESTER LIPSKY received his Ph.D. in Theoretical Atomic Physics from the University of Connecticut in 1965. In 1968 he joined the Department of Computer Science at the University of Nebraska, Lincoln, and became Full Professor in 1976. Presently, (as of 1987) he is Professor of Computer Science and Engineering at the University of Connecticut, becoming Emeritus in 2005. Professor Lipsky's major research interests are in the analytical modeling of the performance of computer systems and networks, and related mathematical and numerical problems. He has authored or co-authored over 150 research articles. His book, "QUEUEING THEORY: A Linear Algebraic Approach" was published by Macmillan, NY in 1992. The second edition will be published sometime in 2007 by Springer-Verlag.

Load Transformations of Markovian Arrival Processes

Stephan Heckmüller, Bernd E. Wolfinger

Dept. of Computer Science, TKRN, University of Hamburg
Vogt-Kölln-Str. 30, D-22527 Hamburg, Germany
E-mail: {heckmueller|wolfinger}@informatik.uni-hamburg.de

Abstract— **Realistic load models are indispensable for the valid performance evaluation of today's complex computer networks. Within these networks, the provisioning of a service is typically achieved by a number of intermediate processing steps, each of which influences load characteristics as seen by succeeding service stations. In this paper, by using the concept of load transformation, we provide methods for explicitly modeling these processing steps. We give general methods for modeling load transformation and present transformations for markovian arrival processes. We demonstrate that two important transformation processes, fragmentation and sliding window mechanism, can be expressed as transformations of markovian arrival processes, and thus retaining analytic tractability. We propose algorithms to match the number of fragments per packet in a markovian setting and investigate the influence of fragmentation on state sojourn time. By comparison between the transformed analytic models and trace data obtained by simulation we demonstrate the validity of our approach.**

Keywords— **Load Modeling, Markovian Arrival Processes, Queueing Systems, Fragmentation, Sliding Window Mechanism**

I. Introduction

Network loads induced by contemporary computer applications in general exhibit very complex statistical properties, which have a strong influence on performance measures of real systems. Besides the development of new performance evaluation techniques and measurement methods considerable progress has been made in the field of realistic description of network loads.

Unfortunately, the characteristics of load as seen at higher layers are altered significantly by processing steps within the transmitting computer network. Among these are fragmentation, header generation, congestion avoidance mechanisms and traffic smoothing. In order to achieve a more realistic representation of loads at lower network layers, in this paper, we propose to model explicitly the influence of these processing steps, which we call *load transformations*. We believe that the concept of load transformation presented here provides a valuable tool for realistic load modeling for different interfaces especially when collecting load measurements is complicated or even impossible. This applies to performance evaluation during a system's design phase and in many cases to load characterizations for lower interfaces. In particular, we are able to model the transformed load based on the characterization of the untransformed load and on knowledge regarding the transformation process.

We show that load models which belong to the class of Batch Markovian Arrival Processes can be transformed directly for a number of important transformations, i.e. fragmentation with delay and sliding window mechanisms. Thus, we achieve a high degree of accuracy in description and retain analytic tractability as well, which would not be the case for simulative transformations.

The concept of load transformation has been studied during the past years [WZHB02], [Bai99], [Zad01]. The load description technique given by the authors emphasizes the interface separating the service system from its environment and permits the specification of load models depending on the service system under consideration. As a direct consequence the authors derive the distinction between primary and secondary load which allows the formal definition of load transformations. In [WZHB02] analytic load transformation processes in computer networks are investigated. Here, the authors concentrate on the analytic description of packet length transformation caused by fragmentation and header generation. Moreover, load transformation by means of simulation is investigated in [Bai99]. Up to our knowledge no transformation procedure unifying timing and length transformation has been given yet.

Batch Markovian Arrival Processes (BMAP) were first introduced with alternative notation in [Neu79]. The author uses the term *versatile Markovian Point Process*. BMAPs in the formulation used in this paper were introduced in [Luc91]. Since then considerable effort has been made to investigate *Batch Markovian Arrival Processes* (BMAP). Recent works include for example the blocking probability of $BMAP/G/1$ Queueing System [CW06], the departure process from a $BMAP/G/1$ Queue [FC01] or the workload distribution for systems with threshold [LB06]. Here the server is assumed to stay idle until workload, i.e. the number of requests waiting for service, exceeds a certain threshold. Other works extend the $BMAP/G/1$ Queueing System: In [Hof01] the $BMAP/G/1$ Queueing System is extended in order to allow the arrival stream to depend on the system's state, i.e. on the number of customers in the system.

In [KLL03] BMAPs are used to model network traffic at IP-Level. The authors give procedures to match

model parameters with measured data. Several modeling techniques for video traffic using $BMAP$s were proposed. In [BC92] variable bit rate sources are modeled by means of BMAPs. In [HSB04] video traffic is modeled by discrete $BMAP$s in order to investigate the queueing behaviour. The authors use the results for buffer dimensioning.
The rest of the paper is organized as follows: In section II we introduce our definition of load transformation. Moreover we recapitulate results concerning transformed packet lengths and present procedures for transforming the timing of concrete arrival sequences. We investigate the influence of packet length distribution on the arrival process after transformation and demonstrate the influence of interarrival times of fragments on load characteristics. In section III we present our transformation procedures for Batch Markovian Arrival Processes for fragmentation with delay and sliding window mechanisms and show that both are able to capture the characteristics of the modeled procedure very well. Additionally, we propose algorithms to match the mean number of fragments per packet within the transformed model and incorporate the modified interarrival times into the transformed model. Finally, we give conclusions in section IV.

II. Load Transformations

Within networks of service stations the load offered to a single station is in general influenced by the service offered by other stations. This is especially true for modern computer networks, which are organized in layers and interconnected systems. Here, each preceding station influences the load offered to its successor and by that transforms, in our terminology, *primary load* into *secondary load*.
In order to describe these transformations in a formally rigouros manner, we define load as shown in Definition 1 [Wol99].

Definition 1: Load $L = L(E, S, IF, T)$ is defined as the sequence of requests, which are provided to the service system S during interval T by its environment E. Requests are passed via interface IF, which separates the service system from its environment. $\diamond$

The load L can be described by a sequence of arriving requests r_i arriving during time interval T. For a well defined load L we define the arrival process as tuples of request r_i and corresponding arrival time t_i

$$\{(r_i, t_i) | r_i \in \mathcal{R}_i, t_1 \leq t_2 \leq ... \leq t_N, t_{1,...,N} \in T\} \quad (1)$$

Single requests r_i could for instance represent data transmission requests or connection setup requests. The concept presented here is not limited to the modeling of computer networks. Considering database systems we could for instance model transactions as requests. Moreover to specify requests r_i we additionally introduce *request types* (e.g. packet transmission request) and *request attributes* (e.g. packet length or destination address). Based on Definition 1 we define four types of transformation

1. We define request transformation as a function T_R, which maps the sequence of primary load requests $R^p = (r_1^p, ..., r_N^p)$ to the sequence of secondary load requests $R^s = (r_1^s, ..., r_K^s)$ for a given transformation.

$$T_R : R^p \rightarrow R^s$$

2. Transformation of timing is defined as a function mapping interarrival times of primary load $T^p = (t_1^p, ..., t_N^p)$ to interarrival times of secondary load $T^s = (t_1^s, ..., t_K^s)$

$$T_T : T^p \rightarrow T^s$$

3. If the transformation of request attributes cannot be achieved independently of timing, we additionally use

$$T'_R : R^p \times T^p \rightarrow R^s$$

4. Likewise, if the transformation of timing cannot be achieved independently of request attributes, we use

$$T'_T : R^p \times T^p \rightarrow T^s$$

Note that due to the generality of Definition 1, we are able to take into account the state of the service station, which is of great importance for load-dependent transformations. In addition, we are also able to model request generation respectively request elimination, because K does not necessarily have to equal N. Moreover, the elements of R^p and R^s don't need to be of the same type.

A. Transformation of attributes

In the following section we examine transformation of request attributes, which in general comprises a wide range of network processes from routing procedures to header generation. Here, we concentrate on the transformation of packet length.
Besides header generation, which can be represented by an additive component, one of the most frequently used length transformations in computer networks is fragmentation. For a given maximum length M a request of length l (e.g. corresponding to a packet to be sent) is divided into $\lceil \frac{l}{M} \rceil$ fragments. The first $\lfloor \frac{l}{M} \rfloor$ fragments are of length M. The length of the last fragment depends on the system under consideration. For systems where the last fragment is expanded (*Padding*) to maximum fragment length (e.g. ATM [HHS98]) the resulting fragment length forms a constant random variable. Otherwise the fragment length distribution $F_F^L(x)$ resulting from packet length distribution $F_P^L(x)$ is given by [WZHB02]:

$$F_F^L(x) = \begin{cases} 0, & x \leq 0 \\ \sum_{i=0}^{\infty} \frac{1}{\beta} \left(F_P^L(iM + x) - F_P^L(iM) \right), & 0 < x \leq M \\ 1, & x > M \end{cases} \quad (2)$$

Here β denotes the mean number of fragments, which can be expressed as:

$$\beta = \sum_{i=1}^{\infty} i(F_P^L(iM) - F_P^L((i-1)M))$$

In Figure 1 the resulting fragment lengths for $M = 1500$ and various length distributions of primary load typically used in load modeling are shown. We choose the corresponding parameters to achieve a mean value of 5000 for all three distributions. As expected, all distributions reach 1 for $X > M$. Note that the differences between the length distributions of secondary load are very small, although there is hardly any similarity between primary load's length distribution.
Further evaluation with different parameters and distributions confirmed that the length distribution of primary load has little influence on the fragment length distribution if the probability $P(X < M)$ is low. In contrast, the influence of the packet length distribution on the distribution of the number of fragments per packet[1] is rather strong. We will get back to this in section III, where we give transformations for fragmentation of markovian arrival processes.

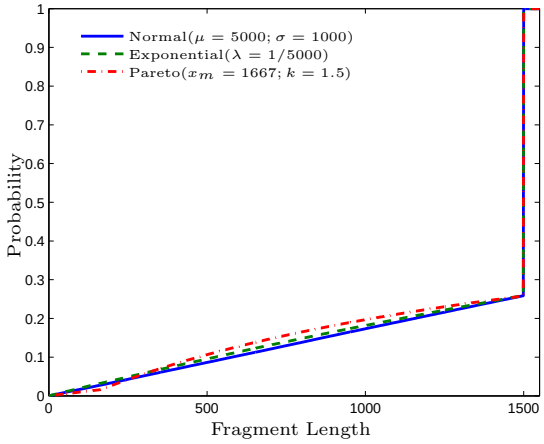


Fig. 1. Fragment Length Distribution for different Packet Length Distributions ($M = 1500$)

B. Transformation of Interarrival Times

Besides attribute transformation many procedures in computer networks do significantly change the timing, especially the request interarrival times, which in turn strongly influences the behaviour of following service stations. This is also the case for fragmentation processes, for which we have neglected the timing aspects up to this point. As fragmentation is not a timeless process it induces non zero interarrival times between fragments. Depending on the amount of time between the fragment arrivals this has an impact on load characteristics and performance measures. Furthermore the interarrival times between fragments are very important for modeling congestion avoidance mechanisms like those inherent in *TCP*, in which case we have to take into consideration the state of all networks being part of the overall end-to-end path.
In the case of fragmentation the timing transformation is not limited to delay transformation, but consists of two transformation procedures. First, packet arrivals at primary load interface possibly induce multiple arrivals at the same time instant, when processing delay is neglected.
For that reason, we introduce fragmentation transformation for arrival instants as:

$$t_i^s = t_j^p, j = \underset{N>0}{\operatorname{argmin}} \left(\sum_{k=1}^{i} l_k^s \leq \sum_{l=0}^{N} l_l^p \right) \tag{3}$$

Here l_i^p respectively l_i^s denotes the size of the i^{th} arrival of primary respectively secondary load. If processing delay is modeled explicitly fragmentation transformation is followed by a delay transformation. This transformation maps the arrival instant of packet i of untransformed load to the corresponding arrival instant of packet i of secondary load by adding a possibly variable delay. For a given sequence of arrival instants $T_i^p = (t_1, ...t_N, t_1 \leq t_2 ... \leq t_N)$ the transformed sequence T_i^s is given by Equation (4).

$$t_i^s = t_i^p + D(i) \tag{4}$$

We can now define delay transformation for request fragmentation recursively. For constant fragmentation time τ_f it is:

$$t_i^s = t_i^p + D(i) = max(t_{i-1}^s + \tau_f, t_i^p + \tau_f \cdot N(i)) \tag{5}$$

Here $N(i)$ denotes the fragment counting function, that is $N(i) = k$, if fragment i is the k^{th} fragment within the burst of all the fragments caused by a single packet. Therefore it exhibits an irregular sawtooth pattern. If we furthermore assume that fragmentation of a packet is always completed until the next arrival instant at primary load level, then, Equation (5) reduces to

$$t_i^s = t_i^p + N(i) \cdot \tau_f \tag{6}$$

To demonstrate the effect of τ_f on the performance of succeeding service stations we computed queue length observed at arrival instants at a $5Mb/s$ link transmitting a fragmented video stream with mean data rate of $1.1Mb/s$. Empirical queue length distribution is presented in the left part of Figure 2 for $M = 1500$ and $\tau_f = \{10^{-3}, 10^{-5}\}$. To emphasize the influence of τ_f the right part of Figure 2 shows the difference between both distributions. The maximum can be found in the region with the highest density and is approximately 0.17.

Concerning the range of values for τ_f it has to be remarked, that not only the time it takes for fragmentation itself is of interest. If we wish to examine, for example, a scenario, where fragments are to be transmitted via a bottleneck link transmission delay can also be taken into consideration. Given this we could easily have $\tau_f = 0.01$ which would reduce burstiness significantly and justifies the necessity of fragmentation modeling. Moreover, as mentioned above, congestion avoidance mechanisms, e.g. in case of *TCP*, can be seen as delay transformations on fragmented network loads, which puts further emphasis on the importance

[1] In the following, we will refer to this distribution as fragment distribution.

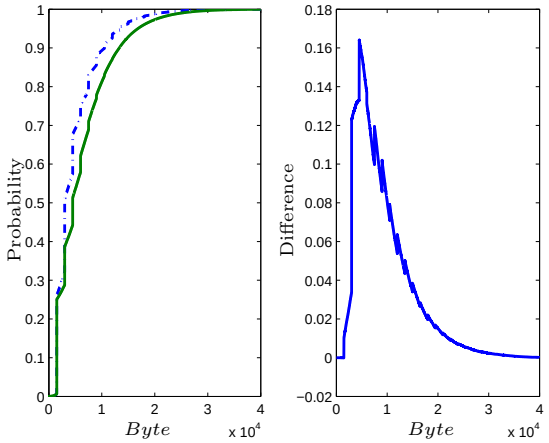


Fig. 2. Queue length distribution at arrivals for $\tau_f = 10^{-3}$ (dashed line) and $\tau_f = 10^{-5}$ (solid line) and the difference between both distributions

of explicit consideration of fragmentation.
As an example for congestion avoidance mechanisms we give the delay transformation based on Equation (4) for a more complex procedure: *sliding window mechanism*. In this case delay is not only caused by fragmentation time but also by the time it takes for acknowledgements to arrive when the window is closed. Assuming that this time is dominated by the *Round-Trip-Time* (RTT) we get

$$t_i^s = max(t_i^p + N(i) \cdot \tau_f, \lfloor \frac{i-1}{W} \rfloor \cdot RTT, t_{i-1}^s + \tau_f) \quad (7)$$

Here W represents the constant window size. The first argument to max in Equation (7) represents the case when the window mechanism imposes no restriction. If this restriction is present, however, this means that request i is delayed up to the next multiple of RTT, which is exactly argument two. Finally argument three represents backlog at time t_i^p. The procedure described above applies for scenarios with approximately constant *Round-Trip-Time*.

III. Transformations of Markovian Arrival Processes

While it is of course possible to implement load transformations in a straightforward algorithmic approach this has some disadvantages: Firstly, secondary load obtained via transformation of concrete sequences of arrivals is in general not suitable for queueing theorical methods, which makes it difficult to obtain performance measures like waiting time distribution or blocking probability. Secondly, even if the primary load model is a stochastic one secondary load has to be determistic because the formulas given above can only be applied to concrete sequences of requests. This implies a loss of generality. Thus, it would be worthwile describing load transformations as a function which preserves analytic tractability.
On the other hand we have to avoid too harsh restrictions concerning modeling capabilities. In order to achieve both goals, we choose the class of *Batch Markovian Arrival Processes* (see Definition 2) [Luc93], which is at the same time suitable for the realistic description of a wide range of network applications and moreover it allows analytical calculations.

Definition 2: Let the *Batch Markovian Arrival Process* (BMAP) be defined as a markovian arrival process with infinitesimal generator matrix:

$$\mathbf{Q} = \begin{pmatrix} D_0 & D_1 & D_2 & D_3 & \dots \\ & D_0 & D_1 & D_2 & \dots \\ & & D_0 & D_1 & \dots \\ & & \vdots & & \ddots \end{pmatrix}$$

With $D_{0,\dots,\infty}$ being $(m \times m)$ matrices defined by:

$$(D_0)_{ii} = -\lambda_i, \quad 1 \le i \le m$$

$$(D_0)_{ij} = \lambda_i p_i(0,j), \quad 1 \le i,j \le m \wedge i \ne j$$

$$(D_k)_{ij} = \lambda_i p_i(k,j), \quad 1 \le i,j \le m \wedge k > 0$$

$$\sum_{\substack{j=1 \\ j \ne i}}^{m} p_i(0,j) + \sum_{k=1}^{\infty}\sum_{j=1}^{m} p_i(k,j) = 1, \quad 1 \le i \le m$$

◇

Here $p_i(k,j)$ denotes the probability that – given the process is in state i – a transition to state j occurs with a batch arrival of size k. Thus, we are able to model arbitrary discrete distributions with possibly infinite support. The overall rate with which the process leaves state i is given by λ_i. It is worthwile noting that $BMAP$ processes are closed with respect to aggregation. That is, we can also describe transformations of aggregated loads or aggregated transformed loads.
In the following, load transformation is therefore restricted to those transformations mapping BMAPs to BMAPs, which leads to an alternative definition of transformation. Since for the description technique under consideration packet lengths and interarrival times are described by one single model we restrict ourselves to one transformation type:

$$T_{BMAP} : \mathcal{BMAP}^p \rightarrow \mathcal{BMAP}^s \quad (8)$$

A. Fragmentation

In section II we examined the effects of fragmentation mechanisms and their influence on timing behaviour. Here we give analogous transformations for fragmentation and delay, which are described by means of BMAPs. We first consider the effect of transformation on the stochastic processes describing packet sizes and interarrival times.
The length of packets generated when a transition from state i to state j occurs follows an i.i.d. discrete distribution, thus the transformed distribution is given by Equation (2), too. Furthermore, we assume that fragment interarrival time τ_f is constant with $\tau_f > 0$ (which could be a realistic assumption, e.g. in voice/video communications). This leads to a transformation of

interarrival times, which changes the spacing between the last respectively first fragment induced by successive arrivals of primary load. In section III-C we give an explicit formula for the distribution of the transformed interarrival times in terms of interarrival distribution of primary load and fragment distribution. For now we assume the mean time for fragmentation in state i is given by $E[T_i^F]$ and give the BMAP transformation for fragmentation procedures in Definition 3. Moreover, the number of fragments generated in state i is given by random variable L_i^F.

Definition 3: Let the primary load be defined by $BMAP$ B. The secondary load after fragmentation with maximum packet length M is modeled as $BMAP_F^S$ with the set of *generating states*

$$G = \{g_i, 0 < i \leq m\},$$

the set of *waiting states*

$$W = \{w_{i,j}, \forall i, j. \exists k. p_i(j,k) > 0\}$$

and matrices $D_{0,\ldots,\infty}$

$$(D_k)_{w_{i,j},g_j} = \begin{cases} \frac{\lambda_j}{1-\lambda_j \cdot E[T_i^F]}, & k = 0 \\ 0, & k \neq 0 \end{cases}$$

$$(D_k)_{w_{i,j},w_{i,j}} = \begin{cases} -\frac{\lambda_j}{1-\lambda_j \cdot E[T_i^F]}, & k = 0 \\ 0, & k \neq 0 \end{cases}$$

$$(D_k)_{g_i,g_i} = \begin{cases} \lambda_f \cdot \left(1 - \frac{1}{E[L_i^F]+1}\right), & k = M \\ 0, & k \neq M \wedge k \neq 0 \\ -\lambda_f, & k = 0 \end{cases}$$

$$(D_k)_{g_i,w_{i,j}} = \begin{cases} \lambda_f \cdot p_i^{TF}(k,j) \cdot \frac{1}{E[L_i^F]+1}, & k \leq M \\ 0, & k > M \end{cases}$$

$\diamond$

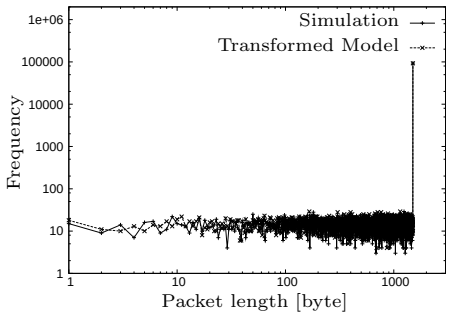
Each state of $BMAP$ B results in two or more states of the transformed process. While states $w_{i,j}$ represent the actual transition rates of the $BMAP$ describing primary load, states g_i model the generation of fragments. Because fragmentation is not modeled as a timeless process we have to increase the rate with which states $w_{i,j}$ are left. This is accomplished by reducing the mean sojourn time by mean fragmentation time $E[T_i^F]$. As we will see in section III-C, the transformed sojourn time is in general not exponentially distributed anymore. To achieve a more realistic model behaviour we could match the transformed distribution with a phase-type distribution and replace $w_{i,j}$ by a set of states.
Transition rates are chosen as shown in Definition 3 to match the original fragment distribution in its mean. The resulting number of fragments per packet is geometrically distributed with parameter $p = \frac{1}{E[L_i^F]+1}$. Generation of full length fragments occurs with a transition from g_i to g_i, while the last fragment arrives with the transition from g_i to its corresponding waiting state $w_{i,j}$. Its size is given by $p_i^{TF}(k,j)$. In order to limit complexity, we assume that the number of fragments generated by a transition from state i to state j does only depend on the current state i.
The interarrival time between fragments is exponentially distributed with rate λ_f. For constant fragment interarrival time τ_f (see Section II) we use $Erlang-k$ distributed sojourn times for generating states instead. Depending on how large we choose k an arbitrary close fit to the constant interarrival time is possible. So there is clearly a tradeoff between model accuracy and complexity. In the following we use $k = 5$.
To evaluate our model, we compared the empirical distribution of fragment size and interarrival times resulting from the transformed $BMAP_F^S$ with those resulting from load generated according to the untransformed $BMAP$. In the latter case fragmentation was done by means of formula (2) respectively (5). As an example, we present $BMAP_1$ modeling an *MPEG*-video stream of a soccer game. It consists of three states representing I-, P- and B-Frames, respectively. Transitions are chosen in order to model the GOP-pattern $IBBPBBPBBPBB$ stochastically. The batch sizes generated in each state are normally distributed with different means and variances shown in table I, where L^P denotes the random variable describing packet length and L^F the random variable describing the number of fragments. Transition rate is 40 for all transitions. The values given are chosen in accordance to trace data of a real *MPEG*-video stream [MPE]. We do not model different degrees of motion intensity by a higher number of states, since this does not influence the validity of our model.

In Figure 3(a) the histogram of fragment lengths is plotted for two fragmented arrival sequences. One was generated according to the transformed model and the other was fragmented by means of the algorithmic approach given in section II. The comparison shows that our model is clearly capable of matching the fragment length distribution. Concerning interarrival times, as expected, our model can not capture the constant fragmentation time exactly for $\tau_f = 10^{-3}s$ (see Figure 3(b)). This is not a major problem for two reasons: Firstly, the constant fragmentation time can be matched more closely by increasing k. Secondly, the constancy is itself an approximate assumption in order to model reality. Real system typically do not exhibit this behaviour in a strict sense. Finally, 3(c) shows interarrival times $\tau_f = 10^{-5}s$. Within the scales under considerations almost no differences are visible.

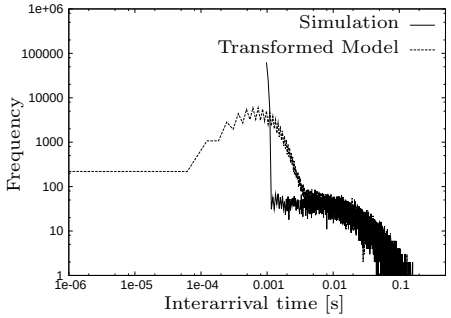
Our model is thus capable of capturing the main char-

State	$E[L^P]$	$Std[L^P]$	$E[L^F]$	$Std[L^F]$
I	8396.7	2211.7	5.5978	1.4745
P	4863.2	1880.6	3.2421	1.2537
B	6360.6	2299.4	4.2404	1.5329

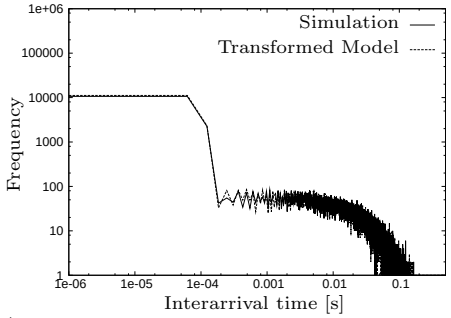
TABLE I: Mean and standard deviaton of packet distribution generated in state i of $BMAP_1$ and of resulting fragment distribution



(a)Comparison of fragment length



(b)Comparison of interarrival Times for $\tau_f = 10^{-3}s$



(c)Comparison of interarrival Times for $\tau_f = 10^{-5}s$

Fig. 3. Evaluation of fragment transformation

acteristics of fragmented load, i.e. the changed distribution of interarrival times and of packet lengths. As we mentioned before, because of the assumption of constant fragmentation time, the transformed interarrival time distribution can only be approximative. Using states with $Erlang - k$ we can nevertheless achieve arbitrary close approximations at the cost of higher model complexity.

B. Fitting the Number of Fragments

While the transformation from packet length distribution to the distribution of fragments per packet is exact for exponentially distributed packet lengths (aside from inherent rounding issues), other distributions can per definition only be matched in their mean. As it is well known, packet lengths produced by many applications do not coincide with the exponential distribution [PF95]. Rather heavy-tailed or normally distributed traffic is of greater interest. In the following we therefore give methods to match the actual distribution of the number of fragments by mixtures of geometric distributions for both distribution types. These mixtures give an immediate mapping to the state pairs $(g_i, w_{i,j})$ introduced in Definition 3.

B.1 Heavy-Tailed Distributed Traffic

In order to match heavy-tailed packet length distribution we use a discrete version of the algorithm presented in [FW97]. With this algorithm (see Equations (9)-(13)) heavy-tailed discrete distributions can be approximated by a fixed number of geometric distributions. The procedure is a recursive one, which matches the discrete distribution at quantiles $c_k < c_{k-1} < ... < c_1$ with $1 < b < \frac{c_i}{c_{i+1}}$. During recursion step i the complemented cdf is computed as the difference between the complemented cdf of step $i-1$ and the geometric distribution with parameter $p_{0_{i-1}}$.

$$F_i^c(x) = F_{i-1}^c(x) - p_{i-1} \cdot (1 - p_{0_{i-1}})^x \tag{9}$$

$$p_{0_i} = 1 - \left(\frac{F_i^c(bc_i)}{F_i^c(c_i)} \right)^{\frac{1}{c_i(b-1)}} \tag{10}$$

$$p_i = \frac{F_i^c(c_i)}{(1 - p_{0_i})^{c_i}} \tag{11}$$

$$p_k = 1 - \sum_{i=1}^{k-1} p_i \tag{12}$$

$$p_{0_k} = 1 - \left(\frac{F_k^c(c_k)}{p_k} \right)^{\frac{1}{c_k}} \tag{13}$$

Formulas (9-13) were derived by solving $p_0(1-p_{0_1})^x = F^c(x)$ for $x = c_1 b$ and $x = c$. If $p_{0_i} >> p_{0_{i-1}}$ the resulting geometric distributions behave sufficiently different, so that the parameters can be chosen independently of each other. To illustrate that we are able to match discrete distributions very accurately with the procedure described above Figure 4 illustrates two discretised pareto distributions and our fit by a mixture of five geometric distributions. (For the pareto distributions we used complemented cdf $F_1^c(x) = (1+x)^{-0.5}$ and $F_1^c(x) = (1+x)^{-0.3}$.) The curves appear almost identical. The quality of the geometric fit depends on the proper choice of the parameters $b, k, c_{1,...,k}$.

As our initial goal was to replace the state pairs $(g_i, w_{i,j})$ of the fragmented $BMAP$ in order to achieve different fragment distributions, we sketch the alternated structure in the left half of Figure 5. For k geometric distributions we now have states $(c_i, g_{i_{1,..,k}}, w_{i,j})$. State c_i serves as a branching state to states $g_{i_{1,...,k}}$ with $\lambda_c >> \lambda_f$. The probability that the $BMAP$ enters state g_{i_j} is given by formulas (11) and (12). The corresponding rates are given by $\lambda_{i_j}^g = (1 - p_{0_j})\lambda_f$ and $\lambda_{i_j}^w = p_{0_j}\lambda_f$. We are thus able to model the fragmentation process within the $BMAP$ class for the case when the given fragment distribution is heavy-tailed and can be matched by our fitting algorithm given above. The increase in the number of states of the transformed $BMAP$ is linear with order $O(k_{max})$.

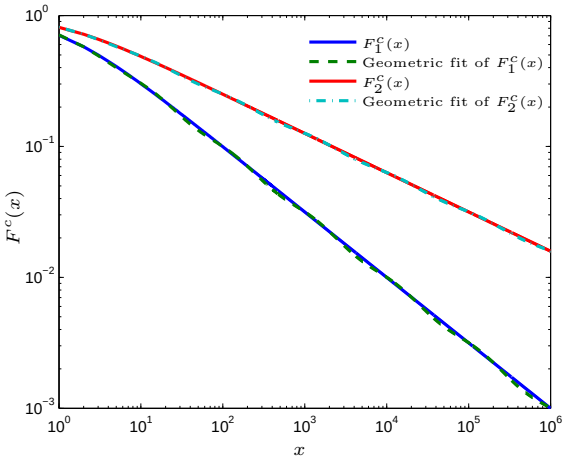


Fig. 4. Approximation of discretised Pareto distributions by mixtures of 5 Geometric distributions

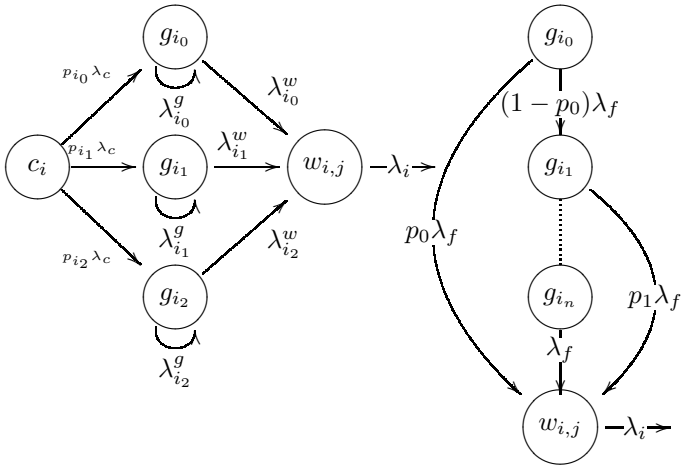


Fig. 5. Structure of transformed state set for heavy-tailed distributed (left) and normally distributed (right) fragments

B.2 Normally Distributed Traffic

Besides heavy-tailed distributions there are a number of applications which produce approximately normally distributed packet lengths, especially video streaming [WZHB02]. In that case for the approximation of the fragment distribution we use the binomial distribution [2], which corresponds to a chain of fragment generating states with branches (see right half of Figure 5).
For given mean and variance of the fragment distribution the parameters p and n are given by Equations (14) and (15).

$$p = 1 - \frac{Var[L^F]}{E[L^F]} \tag{14}$$

$$n = \left\lceil \frac{E[L^F]^2}{E[L^F] - Var[L^F]} \right\rceil \tag{15}$$

The rates are given by $p_i \cdot \lambda_F$ respectively $(1-p_i) \cdot \lambda_F$, where p_i is recursively defined by

$$p_i = \frac{f(i;n,p)}{\prod_{j=0}^{i-1}(1-p_j)}, \quad 0 < i < n \tag{16}$$

[2]The density of the binomial distribution is given by $f(k;n,p) = \binom{n}{k} p^k \cdot (1-p)^{n-k}$

$$p_0 = f(0;n,p) \tag{17}$$

A transition from state g_{i_j} to state $g_{i_{j+1}}$ with $j < n$ always results in a batch arrival of maximum size M, whereas the batch size given a transition from g_{i_j} to $w_{i,j}$ is determined by $p_i^{TF}(k,j)$.

C. Distribution of transformed sojourn time

Because we do not model the transformation process as a timeless one, we additionally give the transformed distributions of sojourn times in waiting states $w_{i,j}$. We use sojourn time instead of rates because in general the time the process should spend in state $w_{i,j}$ follows no exponential distribution anymore. Keeping the term rate would thus be misleading. Nonetheless we will retain markov property by approximating the transformed sojourn time distribution by an exponential one.
In the following we concentrate on the sojourn time of waiting states $w_{i,j}$. Furthermore we assume that fragmentation time is constant, i.e. follows a δ-Distribution. For that case the transformed sojourn time in state w_j is given by Theorem 1 as a mixture of exponentials.

Theorem 1: Given a transition from state g_i to state $w_{i,j}$ the transformed sojourn time in state $w_{i,j}$ has density

$$f_{i,j}^T(x) = \sum_{k=0}^{\infty} p_k \cdot \lambda_j e^{-\lambda_j(x+k\tau_f)}. \tag{18}$$

Here, p_k denotes the probability of having k fragments per packet in state i and τ_f the constant fragmentation time.

Proof The time for fragmentation is distributed like

$$f_F(x) = \sum_{k=0}^{\infty} p_k \delta(x - k\tau_f)$$

which gives the difference distribution

$$\begin{aligned} f_{T-F}(x) &= \int_{-\infty}^{\infty} \lambda_j e^{-\lambda_j(\tau+x)} \cdot f_F(\tau) d\tau \\ &= \int_{-\infty}^{\infty} \lambda_j e^{-\lambda_j(\tau+x)} \cdot \sum_{k=0}^{\infty} p_k \delta(\tau - k\tau_f) d\tau \\ &= \lambda_j \sum_{k=0}^{\infty} p_k e^{-\lambda_j(x+k\tau_f)} \end{aligned}$$

The last equality follows from $\int_{-\infty}^{\infty} g(x+z)\delta(x+c)dx = g(z-c)$. □

To integrate the transformed sojourn time into our model there are two possibilities. First, we can change the leaving rate from $w_{i,j}$ in order to match the transformed sojourn time in mean. Second, we could use a fitting algorithm to match the sojourn by a fixed number of exponentials and use the result to replace $w_{i,j}$

by a compound waiting state. For both cases we have to assume that $f_{T-F}(x)$ has strictly positive support, i.e. fragmentation is done until the next state change with probability 1. As this is clearly not the case, we give the probability $P(T-F<0)$ in Theorem 2, which can be thought of as the error induced by neglecting negative support of $f_{T-F}(x)$.

Theorem 2: The probability $P(T-F<0)$ that fragmentation is still in progress at the next arrival instant is given by

$$P(T-F<0) = 1 - \sum_{k=0}^{\infty} p_k e^{-\lambda_j k \tau_f} \qquad (19)$$

Proof

$$\begin{aligned} P(T-F\geq 0) &= \int_0^{\infty} \lambda_j \sum_{k=0}^{\infty} p_k e^{-\lambda_j (x+k\tau_f)} dx \\ &= \sum_{k=0}^{\infty} \int_0^{\infty} \lambda_j p_k e^{-\lambda_j (x+k\tau_f)} dx \\ &= \sum_{k=0}^{\infty} p_k e^{-\lambda_j k\tau_f} \int_0^{\infty} \lambda_j e^{-\lambda_j x} dx \\ &= \sum_{k=0}^{\infty} p_k e^{-\lambda_j k\tau_f} \\ P(T-F<0) &= 1 - \sum_{k=0}^{\infty} p_k e^{-\lambda_j k\tau_f} \end{aligned}$$

□

D. Sliding Window

In this section we give a $BMAP$-Transformation for sliding window mechanisms. For that purpose we specify the structure of the states of the $BMAP$ describing secondary load. As mentioned in section II-B the interarrival time between two fragments depends on the state of the window mechanism.
This fact is modeled by two distinct states within the state set replacing a single state of the BMAP describing primary load (see Figure 6). State g_{i_0} represents the case when less than W packets haven't been acknowleged yet, which means that interarrival time equals τ_f. Otherwise we have an interarrival time of $\tau_w = RTT - W \cdot \tau_f$ (State g_{i_1}), with RTT, cf. section II-B. State c_i serves as a branching state to model the number of fragments per packet. The probability for a transition to state g_{i_j} when in state c_i is given by:

$$p_{i_0} = \frac{1}{E[L_i^F]+1}, \quad p_{i_1} = \frac{1-p_{i_0}}{W+1}, \quad p_{i_2} = p_{i_1} \cdot W$$

Concerning transition rates we match the transformed distribution of sojourn time T_i^F in state i in its mean which is given by (see Equation (7)):

$$f_i^T(x) = \sum_{k=0}^{N} p_k \cdot \lambda_j e^{-\lambda_j \left(x + \lfloor \frac{k-1}{W} \cdot RTT \rfloor + mod(k,W)\right)}. \quad (20)$$

Transition rate from state $w_{i,j}$ has to be increased because of the amount of time it takes to send the fragments resulting from the last packet arrival. The transformed transition rate is given by $\lambda_i^w = \frac{\lambda_i}{1-\lambda_i \cdot E[T_i^F]}$.
We do not model waiting time as thinking time, which could be independent of the duration of the transmission process. We rather give transformations for arrival processes with arrival instants known a-priori (cf. voice/ video communications). Due to the markovian nature of our model there's clearly no direct relation between the duration of the last fragment burst and the sojourn time in state $w_{i,j}$. We will show however that the analytically transformed model reflects the properties of the Sliding Window Mechanism very well. Moreover, if the fragment distribution is heavy-tailed, we replace the generating state by several states in parallel. If each of that states is in turn replaced by the structure sketched in Figure 6, we achieve a higher degree of correlation between sojourn time in state $w_{i,j}$ and fragmentation time.
To exemplify the properties of Sliding Window Transformation we sketch the transformation of $BMAP_2$ consisting of one load generating state g and one waiting state w. The size of batch arrivals in state g is geometrically distributed with mean 50000. Fragmentation time τ_f was set to $1ms$ with a maximum fragment size of 1000. State w is left with rate 1. Preceding experiments showed that higher values of RTT or lower window sizes W led to more overdispersed distributions of the transformed sojourn time. (That is for the coefficient of variation $c_v = \frac{E[X]}{STD[X]}$ holds $c_v > 1$.) This led to false estimates of packet intervarrival times and consequently to a strong deviation of the total number of fragments sent. In order to capture the overdispersion we choose to model $w_{i,j}$ by 2 states in parallel resulting in a hyperexponential distributed sojourn time.
As can be seen in the upper part of Figure 7, with this correction we achieve very close fits concerning total load with relative difference below 3% in all our experiments. Here the number of packets generated by simulative transformation is shown on the left, whereas the number of packets generated by means of the transformed BMAP is shown on the right. Moreover, the lower part clearly shows that our model captures the main characteristics of time behaviour. As an example, we show empirical interarrival time distribution for $W = 10$ and $RTT = 20ms$. Both expected peaks at τ_f and $RTT - W \cdot \tau_f$ are well approximated by our model, while the actual packet interarrival time (seen at primary load level) shows a good fit, too.

IV. Conclusions

In this paper a number of methods for modeling load transformations were proposed. We concentrated on fragmentation processes within computer networks which is one of most important transformation procedures encountered in networks. After having described the general methods we gave transformations for both request lengths and interarrival times for fragmentation with delay and sliding window mechanisms. As

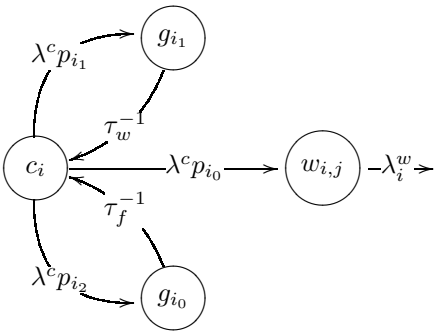


Fig. 6. Structure resulting from Sliding Window Transformation of state i

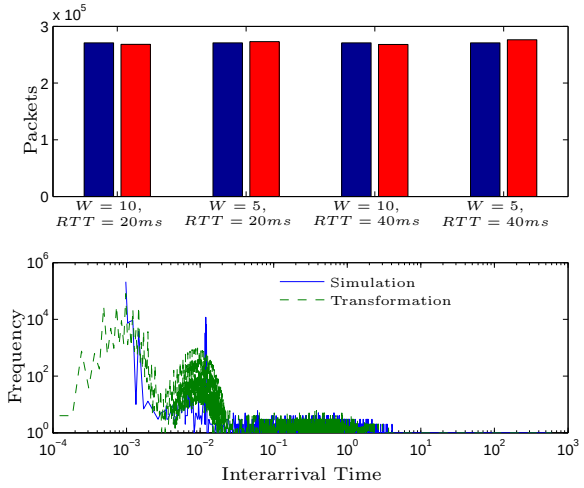


Fig. 7. Evaluation of Sliding Window Transformation

concrete sequences of arrivals lack analytic tractability and generality, we continued by presenting analogous transformation procedures for *Batch Markovian Arrival Processes*, which are at the same time analytically tractable and highly versatile. We evaluated the transformations presented in this paper and showed that the transformed models capture the main characteristics of the transformed loads very well.

We continue to extend the work presented here. Besides further validation with measured data gathered in real networks, we are currently deriving analytic descriptions for other transformation procedures. Moreover, we are extending the sliding window model in order to capture dynamic window sizes and non-constant round-trip-times. We hope that the concept of analytical load transformation presented here provides a valuable tool for the realistic modeling of today's complex network loads.

References

[Bai99] G. Bai, Load measurements and modeling for distributed multimedia applications in high-speed networks, Ph.D. thesis, University of Hamburg, 1999.

[BC92] C. Blondia and O. Casals, Performance analysis of statistical multiplexing of VBR sources, IEEE INFOCOM '92 (Vol. 2) (Los Alamitos, CA, USA), IEEE Computer Society Press, 1992, 828–838.

[CW06] Andrzej Chydzinski and Ryszard Winiarczyk, Blocking Probability in a BMAP Queue, ISCC '06: Proc. 11th IEEE Symp. on Computers and Communications (Washington, DC, USA), IEEE Computer Society, 2006, 547–553.

[FC01] Huei-Wen Ferng and Jin-Fu Chang, Departure Processes of BMAP/G/1 Queues, Queueing Syst. Theory Appl. **39** (2001), no. 2-3, 109–135.

[FW97] Anja Feldmann and Ward Whitt, Fitting Mixtures of Exponentials to Long-Tail Distributions to Analyze Network Performance Models, INFOCOM (3), 1997, 1096–1104.

[HHS98] Rainer Händel, Manfred N. Huber, and Stefan Schröder, ATM networks: concepts, protocols, applications, 3rd ed., Addison-Wesley, 1998.

[Hof01] Jens Hofmann, The BMAP/G/1 Queue with Level-Dependent Arrivals - An Overview, Telecommunication Systems **16** (2001), no. 3-4, 347–359.

[HSB04] Tom Hofkens, Kathleen Spaey, and Chris Blondia, Transient Analysis of the D-BMAP/G/1 Queue with an Application to the Dimensioning of a Playout Buffer for VBR Video, NETWORKING, 2004, 1338–1343.

[KLL03] Alexander Klemm, Christoph Lindemann, and Marco Lohmann, Modeling IP traffic using the batch Markovian arrival process, Perform. Eval. **54** (2003), no. 2, 149–173.

[LB06] Ho Woo Lee and Jung Woo Baek, Threshold Workload Control in the BMAP/G/1 Queue, International Conference on the Quantitative Evaluation of Systems (QEST), IEEE, 2006, 353–364.

[Luc91] D. M. Lucantoni, New results for the single server queue with a batch Markovian arrival process, Stoch. Mod. **7** (1991), 1–46.

[Luc93] D. M. Lucantoni, The BMAP/G/1 queue: a tutorial, Models and Techniques for Performance Evaluation of Computer and Communication Systems (L. Doniatiello and R. Nelson, eds.), Springer-Verlag, New York, 1993, 330–358.

[MPE] MPEG-4 and H.263 Video Traces for Network Performance Evaluation, http://www-tkn.ee.tu-berlin.de/research/trace/trace.html, Last Access: December 2006.

[Neu79] M. F. Neuts, A versatile Markovian point process, J. Appl. Prob. **16** (1979), 764–779.

[PF95] Vern Paxson and Sally Floyd, Wide area traffic: the failure of Poisson modeling, IEEE/ACM Transactions on Networking **3** (1995), no. 3, 226–244.

[Wol99] Bernd Wolfinger, Characterization of Mixed Traffic Load in Service-Integrated Networks, Systems Science Journal **25** (1999), no. 2, 65–86.

[WZHB02] Bernd E. Wolfinger, Martin Zaddach, Klaus D. Heidtmann, and Guangwei Bai, Analytical modeling of primary and secondary load as induced by video applications using UDP/IP, Computer Communications **25** (2002), no. 11-12, 1094–1102.

[Zad01] M. Zaddach, Charakterisierung, Modellierung und Transformation von Videoverkehrslasten, Ph.D. thesis, University of Hamburg, 2001 (in German).

PERFORMANCE ANALYSIS OF A HIGH-SPEED ULTRA-WIDEBAND WPAN MAC

Sergey D. Andreev, Andrey M. Turlikov and Alexey V. Vinel
Department of Information Systems
State University of Aerospace Instrumentation
Bolshaya Morskaya, Saint-Petersburg, Russia
E-mail: vinel@ieee.org

KEYWORDS
Data networks, random multiple access, saturation throughput, MAC, UWB.

ABSTRACT

The current paper addresses the problem of the throughput-delay performance of a contemporary WPAN MAC standard. A brief overview of the standard functionality is first presented that allows a system model derivation. Two different acknowledgement policies are described under which the channel operation is considered. Two possible input traffic models are also considered one of them being saturation conditions under which the performance analysis is done that is further verified by means of the simulation. The obtained results show the system behavior as the number of channel users increase and allow the tuning of the protocol parameters to improve the performance.

INTRODUCTION

With the advent of new technologies the wireless data networks attract more and more attention in the modern world. As a consequence the interest for the analysis of such networks grows steadily. This paper is aimed at the performance analysis of a newly introduced wireless personal area network (WPAN) standard. This standard (ECMA 2005) considers the ultra-wideband physical layer and offers unrivaled data rates which will ensure its applicability. As noticed in (Vishnevsky et al. 2006b) the attention to this standard, especially to its Medium Access Control (MAC) sublayer, is underpaid which leads to limiting its capabilities as 'fine tuning' is yet to be done.

A brief overview of the standard is given in (Vishnevsky et al. 2006a). The current paper extends the understanding of the MAC features and tracts their relations down to the IEEE 802.11 and IEEE 802.16 standards. It is structured as follows. The "MAC SIMPLIFIED DESCRIPTION" section provides a review of the standard-defined mechanisms for the channel access. It is followed by the "PROBLEM STATEMENT" section where the traditional performance metrics are discussed and the necessary questions are posed.

"ANALYTICAL RESULTS" provide more insight into the MAC performance analysis mainly following the approach of (Bianchi 2000) for IEEE 802.11 and of (Vinel et al. 2005) for IEEE 802.16, considering regenerative system behavior, which was noticed in (Vishnevsky and Lyakhov 2002). "NUMERICAL RESULTS" section addresses the verification of the obtained results by means of the simulation. It shows that the information about the number of users in the network, which is available at the MAC layer, can be used to increase the overall system performance. "CONCLUSION" summarizes the main contributions of the paper.

MAC SIMPLIFIED DESCRIPTION

The current paper considers the MAC sublayer of the OSI Reference Model for the high-speed personal ultra-wideband (UWB) wireless networks (ECMA 2005) (referred to as Standard below). The Standard is flexible and supports numerous features such as user mobility, power management, advanced security, range measurement and many more. Therefore only the basic functionality that is relevant to the point of the current paper will be addressed below.

The channel operation in time may be considered as a sequence of adjacent *superframes* which start times (BPSTs) are known to all the channel users. Structurally, each superframe consists of the *beacon period* (BP) immediately followed by the *data period* (see Figure 1). In the BP only *beacons* are transmitted by users with beacon being a type of broadcast frame for the managerial purposes. The beacon includes the necessary service data including the user's id thus enabling all the neighbors to know exactly the total number of users in the system.

Figure 1: Channel Operation Snapshot

In the data period no beacons are transmitted. Instead, three frame types, namely *data* (referred to as *packet* below), *control* and *command* frames may be sent by a user. Alike beacons, control and command frames serve managerial purposes. All the superframes have the same size (65 ms approximately) which means that the more users send beacons in the BP, the fewer data frames may be transmitted in the data period. Should a new user wish to join an existing group of users (beacon group) it first announces its intention in any of the two beacon slots (*signaling slots*) left empty by the other users. The Standard has a robust mechanism ensuring all the users have the same view of the channel operation, i.e. the same BP length and the number of participating users since a so-called hidden terminals problem may occur.

Consider now the data period in more detail. The Standard defines two basic mechanisms of channel access, namely, Distributed Reservation Protocol (*DRP*) and Prioritized Contention Access (*PCA*). The former is a reservation technique enabling users to schedule the channel time for further use through negotiation. The latter is a randomized access scheme which is a generalization of the popular CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) algorithm. Since several reservation types are introduced by the Standard each having complex functional rules it is clear that the proper DRP operation require a sophisticated resource allocation policy to be developed for every channel user type. Therefore one can expect that in practice PCA operation will be more popular than that of DRP. For this reason the primary focus of the current paper will be set on the PCA channel access scheme.

The Standard offers a good variety of the acknowledgement (*ACK*) policies, i.e. different sets of rules according to which the successful receipt of a transmitted packet is verified. The most traditional is Imm-ACK policy, when the acknowledging frame (which is a control frame) is sent back to the sender immediately (as soon as possible) following the successful receipt of a single data frame. This policy is sensible to set when the transmitting data is delay-sensitive and should be delivered with maximum reliability. If the sender receives no ACK frame it retransmits the same packet until in the end its receipt is verified or the limit of retransmission tries is reached when the frame is discarded. The next policy, No-ACK, requires no acknowledgement to be sent. Thus the sender treats each frame that is sent successfully as successfully received. No-ACK is chosen when the data is the most delay-sensitive but tolerate packet drops due to the channel collisions or the background noise. The last possible policy is B-ACK when a sender transmits a block of packets and upon a special request the receiver responds with an ACK vector to acknowledge those packets that are received successfully. The sender may retransmit the packets that dropped.

PROBLEM STATEMENT

Traditionally, in the multiple access theory and its applications, two main metrics are used for the performance evaluation of a MAC protocol. The terms and definitions of these metrics may differ, depending on the problem features and practical issues. In particular, the first metric indicates the efficiency of the channel resources usage and is further referred to as *throughput* being the ratio of channel time spent to transmit the packet payload information. By contrast, the *delay* is the random variable, which characterizes the efficiency of a protocol from the point of view of an arbitrary user. The delay can be measured for any packet in the system as the time interval from the moment the packet arrives at the user queue till it is successfully transmitted. The computation of the throughput and the mean delay values for a typical UWB network scenario is done in the rest of the paper. The following practical questions are to be answered during the analysis:

- What are the values for the mean delay and the throughput when the protocol operates in some typical scenario?
- What is the maximal number of users the protocol can support, provided reasonable values of mean delay and throughput are observed?
- How can the system performance be improved by means of tuning the MAC protocol parameters?

The answers to these questions are found by means of both analytical techniques and interactive simulation.

SYSTEM MODEL

The total of n *users* share the same broadcast channel. As it is baseless to predict how would n change in practice it is considered that the total number of users is constant, which implies no user arrives to or departs from the system. The channel conditions are *ideal*, i.e. there is no background noise (noiseless channel) and all the users can receive each other's transmissions ensuring no hidden terminal is present. Therefore, the three possible situations, namely, successful, collision or empty channel, are distinguished by all the users. The duration of each situation depends solely on the acknowledgement policy considered. Note that the B-ACK policy is parameterized since an implementer must define a number of successive packet transmissions which are verified by a single acknowledgement vector. To avoid implementation issues only the *Imm-ACK* and *No-ACK* policies were chosen to illustrate the system behavior.

For simplicity reasons the time axis is slotted into equal simulation slots. A simulation slot time duration is set to a user clock resolution value as defined in the Standard. As this value is rather small ($1\,\mu s$) such a synchronization can be used for the simulation of the transmissions in the data period.

Two different models of *input traffic* are considered. For the first one the packet arrivals to the system represent Bernoulli process with the constant overall *intensity* $\lambda = \xi n$ packets per superframe, where ξ is the probability that a user generates a packet in a simulation slot. The second traffic model is that of the *saturation conditions* ensuring a user always has a packet that is prepared for transmission. All the packets have the same size of L bytes in total including the necessary headers and trailers. The arrived packet is queued by the user until the moment it is transmitted successfully. The queues are unbounded so that no packet drops would be possible due to overflow.

The PCA protocol defined in the Standard, as mentioned above, is a generalization of the renowned CSMA/CA channel access scheme for the four increasing priorities (categories) of the input traffic, which are Background, Best Effort, Video and Voice respectively. It is based on the truncated binary exponential backoff (*BEB*) conflict resolution protocol. If the channel in *not* sensed busy during the defined Arbitration Inter-Frame Space (*AIFS*) interval the user transmits a pending packet immediately. Otherwise, the user monitors the channel until it is *not* sensed busy and performs a so-called backoff by setting the *backoff counter* value as described below. If collision is sensed (no immediate acknowledgement returned within the SIFS time after the packet transmission) a user also delays the further retransmission for some future time by setting the new backoff counter.

The value of the backoff counter is sampled uniformly in the range $[0, W-1]$, where W is the current value of the *contention window*. The backoff counter value is decreased by a unity after the channel is not sensed busy for AIFS and afterwards every time the empty slot is detected and remains unchanged ('frozen') otherwise (in case of collision or success). When the backoff counter reaches zero a user transmits. In the initialization stage and every time the transmission is successful the user sets its W value to the predefined constant $W_{\min}$. In case of collision the contention window value is doubled until it reaches the upper bound of $W_{\max} = 2^m W_{\min}$ to stop growing (the value of m is referred to as the *maximal backoff stage*). Note that if No-ACK policy is used then the W value is never increased and remains equal to $W_{\min}$ throughout the operation.

One important innovation that extends the classical formulation of the BEB algorithm is the transmission opportunity (*TXOP*) concept. A TXOP is the amount of time for which a user 'captures' the channel and within which it transmits its packets with an interval of SIFS only and without performing the backoff. More specifically, a user transmits pending packets, if any, until all of them are transmitted successfully, collision is sensed, or it reaches a given TXOP limit (Θ). In every outcome a user backoffs according to the BEB rules. Clearly, in the saturation conditions users always transmit maximum packets until TXOP limit provided no collision is sensed. Each traffic priority is defined by the specific values of AIFS, $W_{\min}$, $W_{\max}$ and TXOP limit (see Table 1).

Table 1: System parameters

Parameter name	Parameter variable	Parameter value
Simulation slot time	mClockResolution	1 μs
Traffic priority	mPriority	AC_BK (Background)
Data rate	mRate	53.3 Mbps
Total number of users	n	*Variable*
Total superframe intensity	Λ	*Variable* packets per superframe
Total simulation slot intensity	λ	Λ / T_{SF} packets per simulation slot
User simulation slot intensity	ξ	λ / N packets per simulation slot
Packet length	L	1000 bytes
Superframe duration	mSuperframeLength, T_{SF}	65536 μs
Beacon slot duration	mBeaconSlotLength	85 μs
Beacon period (BP) duration	T_{BP}	$(N+2) \cdot$ mBeaconSlotLength μs
Empty slot duration	pSlotTime	9 μs
SIFS duration	pSIFS	10 μs
AIFS duration	AIFS	7 · pSlotTime + pSIFS
Minimum contention window value	mCWMin, $W_{\min}$	15
Maximum contention window value	mCWMax, $W_{\max}$	1023

Parameter name	Parameter variable	Parameter value
Packet duration	$T_{packet}(L)$	$\left\lceil (42+6\cdot\left\lceil (8\cdot L+38)/100 \right\rceil)\cdot 0.3125 \right\rceil$ μs
Payload duration	$T_{payload}(L)$	$6\cdot\left\lceil (8\cdot L+38)/100 \right\rceil\cdot 0.3125$ μs
Acknowledgment duration	T_{ack}	14 μs
Transmission opportunity limit	mTXOPLimit, Θ	512 μs

ANALYTICAL RESULTS

Due to the complexity of the system defined by the Standard, the analytical results are obtained only for the asymptotic case, when overall traffic intensity is very small ($\lambda \to 0$) and for the saturation case model.
In saturation conditions an approach similar to that of (Binachi 2000) for IEEE 802.11 can be applied. Notice that the main differences in the system model formulated above from the model in (Bianchi 2000) are beacon periods, TXOP limits and No-ACK policy. Therefore, the following assumptions are introduced of which the last two repeat those of Bianchi:

- Only the data period is first considered. The existence of the beacon period will be taken into account in the further analysis.
- The time axis is slotted into non-equal slots. All the users know the slots borders. The duration of a slot depends on the situation in the channel (empty, success or collision). Asynchronous transmissions that occur during the data period are regarded as the synchronous ones.
- The probability that a user starts sending in a slot depends neither on the previous history nor on the behavior of the other users and is denoted as π .

In the framework of the above assumptions, the operation of an arbitrary user is considered. Denoting p as the conditional collision probability of a user making an attempt to transmit in some slot, it is easy to obtain:

$$p = 1-(1-\pi)^{n-1}. \qquad (1)$$

Observing the user states in the beginning of each slot, a sequence of states can be represented by a two-dimensional Markov-chain:

$$\{s(t),\ c(t)\}, \qquad (2)$$

where $s(t)$ is the ratio $W/W_{\min}$ for the user at the beginning of a slot starting at the moment t and $c(t)$ – the value of the user backoff counter at the moment t .
The approach for the computation of π is based on the observation, that the process (2) is a renewal one. Indeed, one can show that according to the binary exponential backoff rules, the moments of user successful transmission are the renewal points.

Consider a process of a packet transmission by a user. Let $\overline{N}$ be the mean number of the packet transmission attempts and $\overline{K}$ be the mean number of slots the user defers the transmission for during this process. Then, the probability π is computed as follows:

$$\pi = \frac{\overline{N}}{\overline{N}+\overline{K}}. \qquad (3)$$

It is easy to see, that the number of the packet transmission attempts is distributed geometrically, thus:

$$\overline{N} = \sum_{i=1}^{\infty} i(1-p)p^{i-1} = \frac{1}{1-p}. \qquad (4)$$

Let $\overline{K}_i$ be the mean number of slots the user has been deferring its transmission for, conditioning that exactly i attempts were made to successfully transmit the packet, then

$$\overline{K} = \sum_{i=1}^{\infty} \overline{K}_i (1-p)p^{i-1}. \qquad (5)$$

One can show that the following equations hold:

$$\overline{K}_i = 2^{i-1}W - \frac{W+i}{2}, \text{ for } 1 \le i \le m+1, \qquad (6)$$

$$\overline{K}_i = 2^m W \frac{i-m+1}{2} - \frac{W+i}{2}, \text{ for } i > m+1. \qquad (7)$$

Substituting (4) – (7) into (3) and after some algebraic simplifications it can be obtained, that:

$$\pi = \frac{2(1-2p)}{(1-2p)(W+1)+pW(1-(2p)^m)}. \qquad (8)$$

The equations (1) and (8) represent the system of two non-linear equalities with two unknowns, π and p , which can be solved numerically to compute π for Imm-ACK. For No-ACK policy the contention window is never increased, thus $m = 0$, what leads to a simple expression for the slot transmission probability:

$$\pi = \frac{2}{W+1}. \qquad (9)$$

Following the above way to compute π the durations of slots in the system for Imm-ACK and No-ACK polices can be summarized in Table 2.

Table 2: Expressions for the Slots Durations

Situation in the channel	Slot duration for Imm-ACK	Slot duration for No-ACK
Success	$T_s^{(1)} = T_{packet}(L) + pSIFS + T_{ack} + AIFS$	$T_s^{(2)} = \Theta + AIFS$
Empty	$T_e = pSlotTime$	
Collision	$T_c^{(1)} = T_{packet}(L) + AIFS$	$T_c^{(2)} = \Theta + AIFS$

Finally, the throughput S can be computed using the following formula:

$$S = \frac{E[data\ period\ length]}{T_{SF}} \cdot \frac{E[payload\ per\ slot]}{E[slot\ duration]}, \qquad (10)$$

which leads to

$$S = (1 - \frac{T_{BP}}{T_{SF}}) \cdot \frac{E[payload\ per\ slot]}{T_e(1-\pi)^n + T_s n\pi(1-\pi)^{n-1} + T_c(1 - n(1-\pi)^{n-1} - (1-\pi)^n)}, \qquad (11)$$

where the values for the slots durations are taken from Table 2. The expressions

$$E[payload\ per\ slot] = \left\lfloor \frac{\Theta}{T_{packet}(L) + pSIFS} \right\rfloor \cdot \cdot T_{payload}(L) n\pi(1-\pi)^{n-1} \qquad (12)$$

for Imm-ACK and

$$E[payload\ per\ slot] = \left\lfloor \frac{\Theta}{T_{packet}(L) + 2pSIFS + T_{ack}} \right\rfloor \cdot \cdot T_{payload}(L) n\pi(1-\pi)^{n-1} \qquad (13)$$

for No-ACK, finish the derivation of the throughput.
For $\lambda \to 0$ the mean delay for the packet transmission D_0 can be computed observing a packet arrival at the empty system, which leads to

$$D_0 = (1 - \frac{T_{BP}}{T_{SF}}) T_s^{(1)} + \frac{T_{BP}}{T_{SF}} (T_{BP}/2 + T_s^{(1)}) \qquad (14)$$

for Imm-ACK policy, and

$$D_0 = (1 - \frac{T_{BP}}{T_{SF}})(AIFS + T_{packet}(L)) + + \frac{T_{BP}}{T_{SF}} (T_{BP}/2 + AIFS + T_{packet}(L)) \qquad (15)$$

for No-ACK policy.
The first terms of the equations (14) and (15) correspond to the case, when a new packet arrives during the data period, while the second terms do to the case, when the arrival takes place during the BP.

NUMERICAL RESULTS

An accurate simulation program has been developed to validate the obtained results. It uses event-driven simulation with a slotted time axis as discussed in the "SYSTEM MODEL" section. The parameters for the simulation runs are summarized in Table 1.
In Figure 2 the saturation throughput versus the user number is demonstrated. Notice that the analytical results give a good approximation of the system performance. The two acknowledgement policies, namely, Imm-ACK and No-ACK, are compared for the standard-defined parameters. Notice that as the number of users increase the throughput performance of the No-ACK policy degrades dramatically.

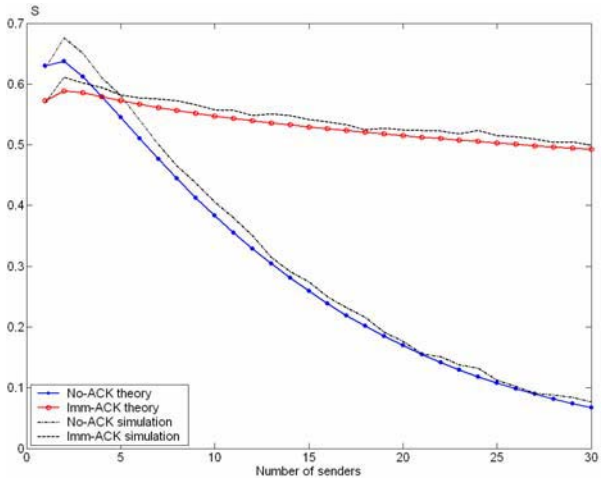


Figure 2: Saturation Throughput Analysis

Low No-ACK throughput is particularly noticeable since the given values are the upper bounds for the real channel throughput because in the presence of the background noise the channel operation suffers further degradation. This effect is basically due to the fact that the users never increase their contention window values. By contrast, the Imm-ACK policy is shown to give higher throughput values for all the user number range considered even despite the ACK overhead.
As mentioned above, there is the information available at the MAC layer about the exact number of users in the system (n). The improvement of the collision resolution protocol on the basis of this information can be done to maximize the saturation throughput. In Imm-ACK case a derivative of (11) should be calculated and set equal to zero. The resulting equality is easy to solve under the assumption that $\pi \ll 1$, which implies $(1-\pi)^n \approx 1 - n\pi + \frac{n \cdot (n-1)}{2} \cdot \pi^2$ and $\pi \approx (n \cdot \sqrt{T_c / 2 \cdot T_e})^{-1}$.
Further through (1) p is obtained, which is used in (8) (or in (9) in No-ACK case) together with π.

The pair (W, m) should next be found which is the solution of the derived equality that maximizes the throughput. Generally, throughput maximization does not lead to the delay minimization. However, as it is illustrated in Figure 3, for the UWB MAC such a choice of parameters reduces the mean delay for the high intensities.

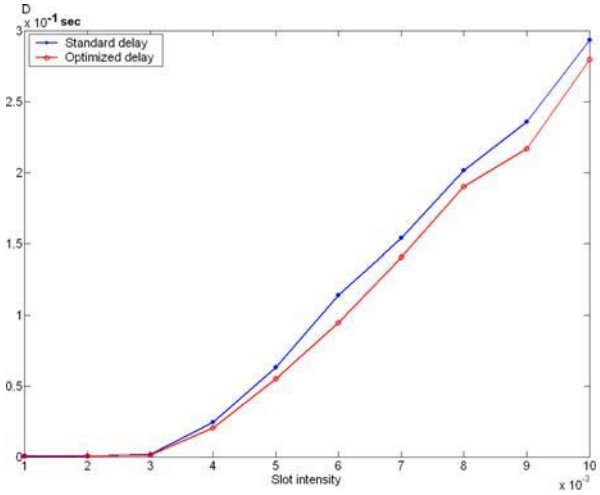


Figure 3: Mean Delay Analysis

CONCLUSION

The main contributions of this paper are twofold. Firstly, the analytical model is developed for the performance analysis of the UWB MAC. Secondly, some interesting numerical results are obtained for the performance metrics of the network and a way to improve the system performance is shown. The further development of the above model is the current research activity of the authors.

REFERENCES

Bianchi G. “Performance Analysis of the IEEE 802.11 Distributed Coordination Function,” IEEE Journal on Selected Areas in Communications, V18, N3, 2000. 535-548.

ECMA-368, “High Rate Ultra Wideband PHY and MAC Standard”, December 2005.

Vinel A., Zhang. Y., Lott M., Turlikov A. “Performance Analysis of the Random Access in IEEE 802.16”, Proc. of the 16th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2005), Berlin, Germany, 2005, 1596–1600.

Vishnevsky V.M. and Lyakhov A.I., “802.11 LANs: Saturation Throughput in the Presence of Noise”, Proc. of 2nd Int. IFIP TC6 Networking Conf. (Networking 2002), Pisa, Italy, May 19-24, 2002. - Lecture Notes in Computer Science, vol.2345, 1008-1019, Springer-Verlag, 2002.

Vishnevsky V.M., Lyakhov A.I., Safonov A.A., Mo S.S., Gelman A.D., "Beaconing in Distributed Control Wireless PAN: Problems and Solutions", Proc. IEEE Consumer Communications & Networking Conference (CCNC 2006), Las Vegas, Nevada, USA, January 8-10, 2006.

Vishnevsky V.M., Lyakhov A.I., Safonov A.A., Mo S.S., Gelman A.D., "Distributed Medium Access Control for High Data Rate Wireless Personal Area Networks", Proc. 10th IEEE International Symposium on Consumer Electronics (ISCE 2006), St. Petersburg, Russia, June 29 - July 1, 2006, 448-452.

AUTHOR BIOGRAPHIES

SERGEY D. ANDREEV was born in Saint-Petersburg, Russia, in 1984. He studied Computer Science at Saint-Petersburg State University of Aerospace Instrumentation, from where he received a Specialist’s degree in 2006. As Ph.D. student at the same university, he continues a research in the field of the random multiple access and is currently interested in wireless telecommunications.

ANDREY M. TURLIKOV is an associate professor of information systems department of Saint-Petersburg State University of Aerospace Instrumentation, where he graduated from in 1980 and received his Ph.D. degree in 1986. Since 1987 he has been involved in teaching work. Dr. Turlikov is the author of more than 60 publications. His research interests include multiple access systems and real-time data transmission protocols.

ALEXEY V. VINEL was born in Saint-Petersburg, Russia in 1983. He has received his bachelor and master degrees in information systems from Saint-Petersburg State University of Aerospace Instrumentation, (where he is a Ph.D. student now), in 2003 and 2005 respectively, both with honors. Since October 2004 till March 2005 Mr. Vinel did an internship at Siemens AG, Munich, Germany. Mr. Vinel’s research interests include random multiple access algorithms analysis and performance evaluation of wireless networks.

PERFORMANCE OF A CLASS OF RETRANSMISSION PROTOCOLS IN CASE OF VARIABLE FEEDBACK DELAYS

Koen De Turck, Stijn De Vuyst and Sabine Wittevrongel
SMACS Research Group, Department of Telecommunications and Information Processing
Ghent University, St-Pietersnieuwstraat 41, 9000 Gent, Belgium
E-mail: {kdeturck,sdv,sw}@telin.UGent.be

ABSTRACT

We consider ARQ retransmission protocols in situations where the length of the feedback delay is modeled by a random process. To keep the analysis tractable, we introduce minor modifications to the standard selective repeat and go-back-N protocols. We study the case where packet errors and feedback delay lengths are independent and identically distributed, as well as the case where both of these system characteristics depend on a Markovian background process with a finite number of states. For both cases we derive stability conditions and closed form expressions for the distribution of the buffer content at the sender side. Numerical examples show a deterioration of the performance under variable feedback delays.

1 INTRODUCTION

Automatic Repeat reQuest (ARQ) is a mechanism to make reliable communication over unreliable channels possible. To this end, the sender divides the data stream into fixed-length packets, adds an error detecting code to each packet, and sends them over the channel to the receiver. The receiver checks the correctness of each packet of which it informs the sender by sending *acknowledgement messages*: a positive acknowledgement (ACK) for each correctly received packet, and otherwise a negative acknowledgement (NAK). The sender retransmits those packets for which a NAK message has been received, thus ensuring a reliable communication. A plethora of different ARQ protocols exists, all of which differ in the exact way that retransmissions and acknowledgements are handled.

Important parameters of the ARQ system with regard to the performance are the *feedback delay*, i.e. the time between the transmission of a packet and the reception of an acknowledgement message for that packet; and also the nature of the error process, which can be modeled by either an independent or a correlated stochastic process. The performance of ARQ has been the subject of numerous studies, some concentrating on the throughput performance (e.g. [1]– [2]), while others also investigate the performance of the buffer at the sender side ([3]– [7]). Some of them have studied the case of correlated errors, which has great practical relevance, e.g. in the context of fading wireless channels.

In this paper we study the throughput *and* the buffer performance of a class of ARQ protocols, with the extra complication that the feedback delay may be a random variable. First, independent and identically distributed (iid) feedback delays are considered, and in later sections we allow for a Markov-type correlation between subsequent feedback delays. Variable feedback delays can occur between a fast-moving sender and/or receiver, in satellite communications, or when transmitting over a network of possibly failing links instead of over just one link.

The salient feature of the class of protocols under study is that at most N packets can be *outstanding*, i.e. waiting for an acknowledgement message, and that the sender combines the acknowledgements of all outstanding packets into a single feedback message. Using joint transmission and acknowledgement can be beneficial in practical protocols, because it is power-saving, as the receiver has fewer messages to send, and it is the transmission of packets that is the dominating power cost. It also avoids the complex situation that packets can overtake each other, i.e. where packet B is transmitted later than packet A, but its acknowledgement arrives earlier at the sender side. This would complicate the practical implementation, and, perhaps even to a higher degree, the theoretical performance study.

For $N = 1$, this trivially reduces to the classic stop-and-wait protocol. For $N > 1$, we will analyse two different variants. For the first one, we impose the further restriction that packets must arrive correctly at the receiver side *in order*, so the receiver requests the retransmission of not only the incorrect packet, but also of every packet with higher sequence number. This variant resembles the go-back-N protocol, except that acknowledgement messages are grouped together. In the following, we will refer to this variant (see figure 1) as simply go-back-N. The other variant is that retransmissions are requested only for the erroneous outstanding packets. This variant (see figure 2) bears similarity with the selective repeat protocol, and in the following we will refer to it as such. We also use another power-saving feature. To avoid frequent transmissions of incom-

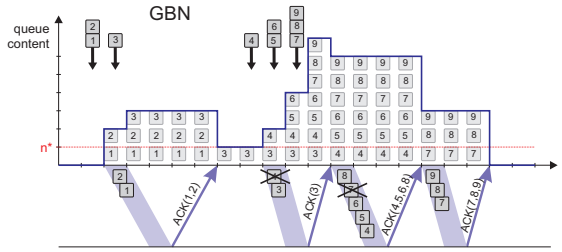


Figure 1: Evolution of the queue content for the go-back-N variant.

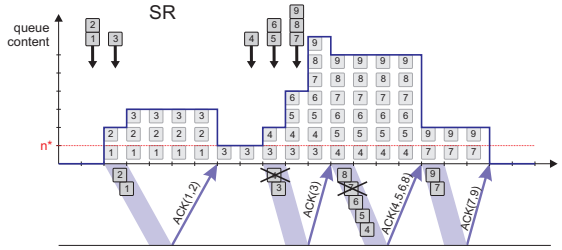


Figure 2: Evolution of the queue content for the selective repeat variant. Note that packet 8 need not be retransmitted here.

plete groups, i.e. with *less* than N packets, we assume that the sender only transmits a group if more than n^* packets are available in the buffer.

The rest of this paper is organised as follows. In section 2, we present the analysis for the case of independent feedback delays. The analysis is extended in section 3 to the case of correlated feedback delays. Numerical examples are discussed in section 4. The paper is concluded in section 5.

2 CASE A: FIXED ERROR PROBABILITY, INDEPENDENT FEEDBACK DELAYS

2.1 MODEL DESCRIPTION

We consider the buffer in the sender part of an ARQ system. The buffer is assumed to be of infinite capacity. We divide the time axis into intervals of the same length, called slots. Packets of fixed length arrive at the buffer according to an independent and identically distributed (iid) stochastic process. We introduce the random variable a, denoting the number of packets that arrive during an arbitrary time slot.

The investigated protocols send the packets per *window* of maximum N, which means that no more than N packets can wait for an acknowledgement at the same time. When a group of packets is transmitted, the sender temporarily stops sending until new acknowledgements are received. The receiver sends a combined acknowledgement for the entire window. This message contains an ACK or NAK for each individual packet in the window. A detailed description of the system operation is as follows:

1. During each slot, a random number of a new packets arrive at the buffer. We assume that these random numbers are iid. Delayed access is assumed, i.e. arriving packets do not enter the buffer until the end of their arrival slot.

2. The sender is either idle until an acknowledgement message arrives, or it is ready to send new packets. Slot boundaries in which the sender is in the latter condition are called *send attempts*. The sender will only start sending when the buffer content exceeds a certain threshold of n^* packets, otherwise the sender waits until this condition is fulfilled.

3. After a successful send attempt, i.e. a send attempt where the sender has transmitted a group of packets, the sender becomes idle for an additional $f-1$ slots, Here, f is the random variable denoting the (variable) feedback delay. The number of packets that arrive during an arbitrary feedback period is denoted by the random variable α.

4. When the acknowledgement message arrives, r packets will need to be resent, while the rest can leave the buffer. The distribution of this random variable depends on the probability p that a packet is received incorrectly, the actual number of sent packets, and on the protocol. The system operation for both go-back-N and selective repeat variants is illustrated in figure 1 and figure 2 respectively.

2.2 BUFFER CONTENT AT SEND ATTEMPTS

We now study the buffer content at send attempts. Let the random variable v_k denote the buffer content at the beginning of the kth send attempt. Following system equation must hold:

$$v_{k+1} = \begin{cases} v_k - N + r + \alpha, & \text{when } v_k \geq N, \\ r + \alpha & \text{when } n^* < v_k < N, \\ v_k + a & \text{when } v_k \leq n^*. \end{cases} \quad (1)$$

Let $A(z), F(z), \alpha(z)$ denote the probability generating functions of a, f and α respectively. We also introduce $R_n(z)$, the probability generating function of r, given that n packets were sent. Clearly, the function $R_n(z)$ will depend on the considered protocol. It will be derived for both go-back-N and selective repeat variants in subsection 2.3. Furthermore, let $v(i)$ and $V(z)$ be respectively the probability mass function and the probability generating function of v_k during stochastic equilibrium. Using (1), we can derive generating function $V(z)$ as follows:

$$\begin{aligned} V(z) &= \lim_{k\to\infty} \mathsf{E}[z^{v_k}] \\ &= \lim_{k\to\infty} \Big\{ \mathsf{E}[z^{v_k - N + r + \alpha} \{v_k \geq N\}] \end{aligned}$$

$$+\sum_{i=n^*+1}^{N-1}\mathsf{E}[z^{r+\alpha}\{v_k=i\}]+\sum_{i=0}^{n^*}\mathsf{E}[z^{v_k+a}\{v_k=i\}]\Big\}$$
$$=\left(V(z)-\sum_{i=0}^{N-1}v(i)z^i\right)z^{-N}R_N(z)\alpha(z)$$
$$+\sum_{i=n^*+1}^{N-1}v(i)R_i(z)\alpha(z)+\sum_{i=0}^{n^*}v(i)z^iA(z), \quad (2)$$

where $\mathsf{E}[G\{H\}]=\Pr[H]\mathsf{E}[G|H]$. Hence, we find the following expression for $V(z)$:

$$V(z)=\frac{1}{z^N-R_N(z)\alpha(z)}\sum_{i=0}^{N-1}v(i)\big(z^Ng_i(z) - z^iR_N(z)\alpha(z)\big), \quad (3)$$

where $g_i(z)$ is defined as:

$$g_i(z)=\begin{cases} R_i(z)\alpha(z), & n^*+1\le i\le N-1,\\ z^iA(z), & 0\le i\le n^*.\end{cases} \quad (4)$$

Note that $\alpha(z)=F(A(z))$ since α is a sum of f iid random variables with common probability generating function $A(z)$.

2.3 DERIVATION OF $R_n(z)$

Let $r(i,n)$ denote the probability that i out of n packets need to be resent. Obviously, $r(i,n)=0$ when $i>n$. For a go-back-N protocol, when a transmission error occurs, the corrupt packet *and* all following packets have to be resent, so we get:

$$r(0,n)=(1-p)^n, \text{ and } r(i,n)=(1-p)^{n-i}p, \quad (5)$$

where p is the packet error probability. For selective repeat protocols, only the corrupt packets have to be resent, so we get following binomial probabilities:

$$r(i,n)=C_n^ip^i(1-p)^{n-i}, \quad (6)$$

where C_n^i are binomial coefficients. The corresponding probability generating functions $R_n(z)\doteq\sum_{i=0}^n r(i,n)z^i$ can now easily be derived. It follows that for go-back-N:

$$R_n(z)=(1-p)^n+pz\frac{(1-p)^n-z^n}{1-p-z}, \quad (7)$$

and for selective repeat:

$$R_n(z)=(1-p+pz)^n. \quad (8)$$

2.4 BUFFER CONTENT AT RANDOM SLOTS

We want to know the distribution of the buffer content on an arbitrary slot boundary, with probability generating function $U(z)$, and probability mass function $u(n)$.

We first derive the distribution of the buffer content at successful and unsuccessful send attempts, i.e. send attempts at which the buffer content respectively does and does not exceed the threshold n^*. The probability $v^*(n)$ that there are n packets in the buffer at an unsuccessful send attempt is equal to

$$v^*(n)=\frac{v(n)}{\sum_{k=0}^{n^*}v(k)}. \quad (9)$$

The probability generating function $V_s(z)$ of the buffer content at successful send attempts is equal to

$$V_s(z)=\frac{V(z)-\sum_{k=0}^{n^*}v(k)z^k}{1-\sum_{k=0}^{n^*}v(k)}. \quad (10)$$

We distinguish between the case that the buffer content during a random slot is not larger the threshold, and the case that it exceeds the threshold.

When the buffer content does not exceed the threshold n^*, it is clear that this must be at an unsuccessful send attempt. The probability $u^*(n)$ of having a buffer content of n *provided* that $n\le n^*$ during a random slot is equal to $v^*(n)$.

Next, we derive the probability generating function $U_s(z)$ of the buffer content during a random slot provided that it exceeds the threshold. Such random slot must occur during a feedback delay following a successful send attempt. Hence, the buffer content of a random slot that exceeds n^* is equal to the buffer content at the previous successful send attempt *plus* the number ℓ of customers that arrive up to a random slot of a period of length f with distribution $F(z)$. The random variable ℓ has generating function $L(z)$, which we can derive using a formula from Takagi [8]:

$$L(z)=\frac{\mathsf{E}_f[\sum_{j=0}^{f-1}A^j(z)]}{\mathsf{E}[f]}=\frac{F(A(z))-1}{E[f](A(z)-1)} \quad (11)$$

Hence, the probability generating function $U_s(z)$ is given by:

$$U_s(z)=V_s(z)L(z). \quad (12)$$

The probability generating function $U(z)$ of the buffer content during a random slot is a weighted sum of $U_s(z)$ and $\sum_{k=0}^{n^*}u^*(k)z^k$, as we must account for the fact that every succesful send attempt on average corresponds to $\mathsf{E}[f]$ random slots, while an unsuccessful send attempt corresponds to only one random slot. Hence, we find for $U(z)$:

$$U(z)=\frac{\Pr[v\le n^*]\sum_{k=0}^{n^*}u^*(k)z^k+E[f]\Pr[v>n^*]U_s(z)}{\Pr[v\le n^*]+\mathsf{E}[f]\Pr[v>n^*]}$$
$$=\frac{\sum_{j=0}^{n^*}v(j)z^j+\left(V(z)-\sum_{j=0}^{n^*}v(j)z^j\right)L(z)}{\Pr[v\le n^*]+\mathsf{E}[f]\Pr[v>n^*]}. \quad (13)$$

2.5 THROUGHPUT AND STABILITY

The throughput is defined as the average number of packets that gets correctly transmitted per slot, provided that there are always enough new packets available. When this condition is fulfilled, the sender can always transmit groups of N packets, and it takes a random number f of slots until the sender can start the transmission of the next group. Of each group, there will be on average $N - \mathsf{E}[r] = N - R'_N(1)$ successful transmissions, so that the throughput η equals:

$$\eta = \frac{N - R'_N(1)}{F'(1)}. \tag{14}$$

In (14) we use primes to denote derivatives. Note that when $F'(1) < N$, the throughput can in effect be larger than 1, which runs counter to the normal convention that 1 slot equals 1 transmission time. When the feedback delay would have a constant duration of, say f slots, than the throughput η would be $\frac{N - R'_N(1)}{f}$, as would also be found when taking $F(z) = z^f$.

Stability is ensured when there are on average less arriving packets than the throughput. Hence, the system is stable when following condition is fulfilled:

$$\mathsf{E}[a] < \eta \Longleftrightarrow A'(1)F'(1) < N - R'_N(1). \tag{15}$$

3 CASE B: BACKGROUND STATES

3.1 MODEL DESCRIPTION

In this section, we extend the analysis by allowing the packet error probability and the feedback delay to depend on a Markovian background state. Suppose there are a finite number M of such background states. The evolution of the feedback delay and background state is defined by the following probabilities:

$$f(i,j,k) \doteq \Pr[\text{next state} = j, \text{feedback delay} = k \\ |\text{current state} = i]. \tag{16}$$

It will prove useful to collect these probabilities into a *probability generating matrix* $\mathbf{F}(z)$. This is an $M \times M$ matrix of partial generating functions, with the entry on row i and column j equal to:

$$[\mathbf{F}(z)]_{ij} = \sum_{k=0}^{\infty} f(i,j,k) z^k. \tag{17}$$

The background state can also change when there are not enough packets to initiate a transmission. In that case, transitions occur according to transition matrix $\mathbf{q}$, with entries defined as $[\mathbf{q}]_{ij} = \Pr[\text{next state } = j | \text{current state} = i]$.

A final change compared to the uncorrelated case is that the packet error probability depends on the background state, so we introduce the error probabilities $p(i)$. We introduce generating matrices $\mathbf{R}_n(z)$ which are diagonal and the entry on row i, column i is equal to $R_n(z)$ with $p(i)$ substituted for p.

3.2 BUFFER CONTENT AT SEND ATTEMPTS

We now derive the generating vector function $\mathbf{V}(z)$ of the buffer content at send attempts. The definition is as follows:

$$[\mathbf{V}(z)]_k \doteq \sum_{n=0}^{\infty} z^n \Pr[\text{buffer content} = n, \text{state} = k] \\ = \sum_{n=0}^{\infty} z^n [\mathbf{v_n}]_k. \tag{18}$$

After an analogous derivation we find that:

$$\mathbf{V}(z) = \sum_{i=0}^{N-1} \mathbf{v}(i) \left(z^N \mathbf{f}_i(z) - z^i \mathbf{R}_N(z) \boldsymbol{\alpha}(z) \right) \\ \times (z^N I - \mathbf{R}_N(z)\boldsymbol{\alpha}(z))^{-1}, \tag{19}$$

where $\mathbf{f}_i(z)$ is defined as:

$$\mathbf{f}_i(z) = \begin{cases} \mathbf{R}_i(z)\boldsymbol{\alpha}(z), & n^* + 1 \le i \le N-1, \\ z^i A(z) I, & 0 \le i \le n^*, \end{cases} \tag{20}$$

where I denotes the identity matrix of appropriate dimensions, and $\boldsymbol{\alpha}(z)$ is equal to $\mathbf{F}(A(z))$. Formula (19) contains MN unknowns. These can be found by cancelling out the zeros of $(z^N I - \mathbf{R}_N(z)\boldsymbol{\alpha}(z))$ inside the unit circle. It can be shown that under stability conditions, there are exactly $MN - 1$ such zeros [9]. Together with the normalization condition, these equations suffice to determine all the unknowns.

3.3 BUFFER CONTENT AT RANDOM SLOTS

The reasoning of subsection 2.4 can be repeated here, with some slight adaptations.

Because the duration of a feedback delay depends on the background state at a send attempt, we must take these background states into account and start from the joint distribution of the buffer content at send attempts and the background states. The key idea in the following is that when we filter out only those send attempts that have the same background state, say j, we can apply the same reasoning as in subsection 2.4, and thus, for every j, get a relationship between $v(j,k) \doteq [\mathbf{v_j}]_k$ and $u(j,k)$, defined as the probability that during a random slot, the buffer content is j and the background state at the most recent send attempt was k. The distribution of the duration of a feedback delay that starts in background state j is given by the generating function $F_{j*}(z)$:

$$F_{j*}(z) = \sum_{k=1}^{M} [\mathbf{F}(z)]_{jk}. \tag{21}$$

Analogous to (11), we define also $L_{j*}(z)$, which is equal to

$$L(z) = \frac{F_{j*}(A(z)) - 1}{A(z) - 1}. \quad (22)$$

Then we find that the partial generating function $U_j(z)$ of the buffer content, given that the background state during the most recent send attempt is j, equals:

$$U_j(z) = \frac{\sum_{k=0}^{n^*} v(j,k)z^k + \left(V_j(z) - \sum_{k=0}^{n^*} v(j,k)z^k \right) L_{j*}(z)}{\Pr[v \le n^*, j] + \mathsf{E}[f] \Pr[v > n^*, j]}. \quad (23)$$

This joint distribution is of no great practical value on its own, so we define $U(z)$, the generating function of the buffer content at random time instants, as:

$$U(z) = \sum_{j=1}^{M} U_j(z). \quad (24)$$

3.4 THROUGHPUT AND STABILITY

The derivation of the throughput is slightly more involved than in the uncorrelated case. The throughput equals the average number of successfully transmitted packets divided by the average feedback delay. As both of these features depend on the background state, we must first determine the distribution of the background state at send attempts, $\boldsymbol{\pi}$, which is a row vector with M entries, and where $[\boldsymbol{\pi}]_j$ is the probability that the background state is j at send attempts (under heavy traffic).

Hence, the distribution of background states at send attempts $\boldsymbol{\pi}$ is a stationary solution of the transition matrix $\mathbf{F}(1)$:

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{F}(1), \text{ and } \boldsymbol{\pi}\mathbf{1} = \mathbf{1}, \quad (25)$$

where $\mathbf{1}$ is a column vector of appropriate size. In other words, $\boldsymbol{\pi}$ the normalized left eigenvector of $\mathbf{F}(1)$ corresponding to eigenvalue 1.

The average number of successfully transmitted packets at send attempts equals $N - \boldsymbol{\pi}\mathbf{R}'_N(1)\mathbf{1}$, while the average feedback delay at send attempts equals $\boldsymbol{\pi}\mathbf{F}'(1)\mathbf{1}$, hence the throughput η equals

$$\eta = \frac{N - \boldsymbol{\pi}\mathbf{R}'_N(1)\mathbf{1}}{\boldsymbol{\pi}\mathbf{F}'(1)\mathbf{1}}. \quad (26)$$

The condition for stability leads in this case to the following expression:

$$\mathsf{E}[a] < \eta \iff A'(1)\boldsymbol{\pi}\mathbf{F}'(1)\mathbf{1} < N - \boldsymbol{\pi}\mathbf{R}'_N(1)\mathbf{1}. \quad (27)$$

An important subclass of the model is obtained when only the error probabilities depend on the background state, and not the duration of feedback delays. The matrix generating function $\mathbf{F}(z)$ can in that case be written as:

$$\mathbf{F}(z) = F(\mathbf{q}z), \quad (28)$$

for some generating function $F(z)$, and where $\mathbf{q}$ is the transition matrix of the background state. The reader can easily verify that in this case, $\boldsymbol{\pi}\mathbf{F}'(1)\mathbf{1} = F'(1)$.

4 NUMERICAL EXAMPLES

In this section we will show the impact of variable feedback delays on different performance characteristics. We first consider some examples with uncorrelated feedback delays. Figure 3 shows the mean delay versus the mean feedback delay. The mean delay is found via the mean buffer content using Little's Law. The batch sizes of the arrival process are considered geometric, and the load is fixed at 0.8. The error probability p is 0.2. We see that for geometric feedback delays, the packet delay is much higher than for constant feedback delays, and that the selective repeat variant outperforms the go-back-N variant. Figure 4 shows the throughput versus the mean feedback delay. Note that the exact distribution of the feedback delay does not matter, as the throughput only depends on the mean feedback delay. We see that the throughput of the go-back-N variant is considerably worse than that of the selective repeat variant.

For our examples of the case with background states, we will use the simple but powerful Gilbert-Elliott channel model [10]. This channel model has two background states, dubbed 'GOOD', which has a low packet error probability $p(1)$, and 'BAD', which has a higher error probability $p(2)$. Transitions occur from slot to slot. The transition matrix $\mathbf{q}$ of the Markov chain has 4 entries, which can be conveniently written in terms of the following two parameters, which have a clear physical meaning: σ is the stationary probability of being in the BAD state, and K is a measure for how much longer the average BAD period is compared to the uncorrelated case. The background state does not affect the distribution of the feedback delays. Figure 5 shows that the error correlation does not change the point of instability, as the curves corresponding to the same protocol tend to the same asymptotic. Nevertheless the mean buffer content in case of the highly correlated channel is consistently higher than in the lightly correlated case.

5 CONCLUSION

We studied the performance of ARQ when the feedback delay is a random variable, for a fairly broad class of protocols. Numerical examples show that with a variable feedback delay, the buffer performance deteriorates, in some cases to a very high degree.

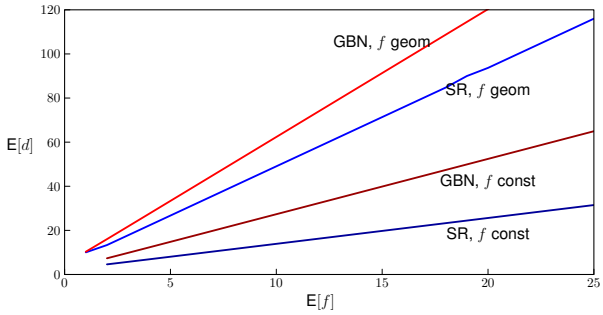


Figure 3: Mean delay versus mean feedback delay of the go-back-N and selective repeat variants, and both geometric and deterministic feedback delays, in case of $N = 5, n^* = 1$ and geometrically distributed arrivals.

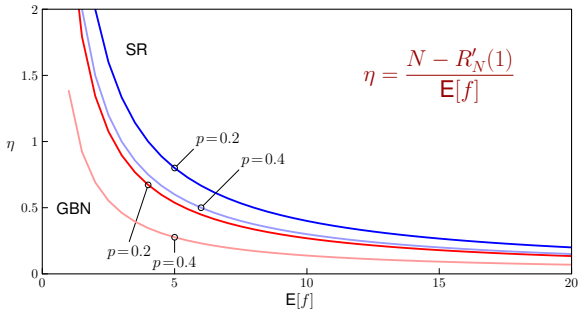


Figure 4: Throughput of the go-back-N and selective repeat variants versus mean feedback delay, for error probabilities $p = 0.2$ and 0.4, group size $N = 5$.

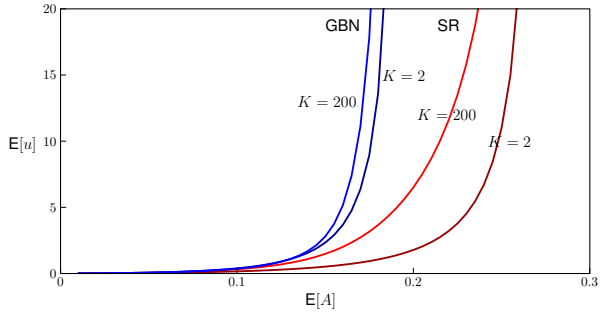


Figure 5: Mean buffer content of the go-back-N and selective repeat variants over a lightly ($K = 2$) and heavily ($K = 200$) correlated Gilbert-Elliott channel for varying arrival rates, $N = 2, p(1) = 0.1; p(2) = 0.5, \sigma = 0.5$ and geometric feedback delays with $\mathsf{E}[f] = 4$. Arrivals occur in geometric batch sizes.

REFERENCES

[1] Bhunia, C.T.: ARQ - Review and modifications. IETE Technical Review **18** (2001) 381–401

[2] Bruneel, H., Moeneclaey, M.: On the throughput performance of some continuous ARQ strategies with repeated transmissions, IEEE Transactions on Communications, 1986, vol. COM-34, no. 3, pp. 244-249.

[3] Towsley, D., Wolf, J.K.: On the statistical analysis of queue lengths and waiting times for statistical multiplexers with ARQ retransmission schemes. IEEE Transactions on Communications **25** (1979) 693–703

[4] Konheim, A.G.: A queueing analysis of two ARQ protocols. IEEE Transactions on Communications **28** (1980) 1004–1014

[5] Towsley, D.: A statistical analysis of ARQ protocols operating in a non-independent error environment. IEEE Transactions on Communications **27** (1981) 971–981

[6] De Vuyst, S., Wittevrongel, S., Bruneel, H.: Delay analysis of the Stop-and-Wait ARQ scheme under correlated errors. Proceedings of HET-NETs 2004, Performance Modelling and Evaluation of Heterogenous Networks (26-28 July 2004, Ilkley, West Yorkshire, UK), p. 21/1–21/11

[7] De Turck, K., Wittevrongel, S.: Delay analysis of the go-back-N ARQ protocol over a time-varying channel, Proceedings of the Second European Performance Evaluation Workshop, EPEW 2005 (Versailles, 1-3 September 2005), Lecture Notes in Computer Science.

[8] Takagi, H.: *Queueing analysis, a foundation of performance evaluation, Vol. 3: Discrete-time systems*, Elsevier Science Publishers BV, Amsterdam, The Netherlands, 1993.

[9] Gail, H. R., Hantler, S. L., Taylor, B. A.: Spectral analysis of $M/G/1$ and $G/M/1$ type Markov chains. Adv. Appl. Prob. **28** (1996) 114–165

[10] Gilbert, E.N.: Capacity of a burst noise channel. The Bell System Technical Journal **39** (1960) 1253–1265

SESSION 3

Queueing Systems

AN INTRODUCTION TO CLASSICAL CYCLIC POLLING MODEL

Zsolt Saffer
Department of Telecommunications
Technical University of Budapest
1521 Budapest Hungary
safferzs@hit.bme.hu

ABSTRACT

An alternative analysis of the classical $M/G/1$ cyclic order polling model is presented. We introduce the new Markov regenerative process framework, which provides a way to perform the analysis of the polling model on a unified, i.e. service discipline independent way. We derive the decomposition property and the general expressions of the stationary number of customers as well as the stationary waiting time. We show a unified way to reduce the analysis for finding the stationary number of customers at the polling epochs and demonstrate it on several known service disciplines. The tutorial character of the approach makes it appropriate to be used for an introduction.

KEYWORDS:

polling model, service discipline, decomposition, zero and nonzero-switchover-times model.

INTRODUCTION

In this paper we study the classical polling model, where the server attends the queues in a cyclic order. For a summary on such systems see (Takagi 1986). An early work of (Kuehn 1979) has shown essential service discipline independent formulas of the stationary mean cycle time, station service time and intervisit time. A basic property of such systems is an $M/G/1$ decomposition property. It means, that the stationary number of the customer who arrives to queue i can be decomposed as the sum of two independent random variables.

The $M/G/1$ decomposition property in a more general setting was presented first by (Fuhrmann and Cooper 1985). They have proved it for the $M/G/1$ queue with generalized vacation. In the literature on polling models, traditionally the nonzero-switchover-times and the zero-switchover-times models are considered separately. Consequently considerable research effort has been taken to relate the zero- and nonzero-switchover-times models to each other, see e.g. (Srinivasan et al. 1995). Later (Borst and Boxma 1997) has unified the derivation of the stochastic decomposition for both the nonzero- and the zero-switchover-times models.

The principial goal of this paper is to present a general introduction to the classical cyclic polling model, that unifies the above mentioned results.

The contribution of this paper is two-fold. The first contribution is the alternative analysis of the classical $M/G/1$ cyclic order polling model. We provide a unified way to reduce the analysis for finding the stationary number of customers at the polling epochs. The second contribution is the introduction of the Markov regenerative process (MRP) framework. This provides a way to perform the analysis of the polling model on a unified, i.e. service discipline independent way.

The rest of this paper is organized as follows. In the next section we introduce the classical cyclic polling model. Section III gives a brief description of the MRP framework. The analysis of the polling model follows in section IV. Section V supplements the results. In section VI we demonstrate on some waiting time formulas, how the results can be used to derive several known expressions in a simple and unified way. Our conclusion is given in section VII.

THE CLASSICAL CYCLIC POLLING MODEL

Consider a continuous-time asymmetric polling model. A single server attends a sequence of N stations in cyclic order. Each station has an infinite buffer queue, which is served when the server attends that queue. The arrival process of the customers is Poisson at each station. λ_i denotes the stationary arrival rate at station i. The customer who arrives to station i is called i-customer. The customer service times have general distributions, but no bulk departure is allowed. B_i, b_i and $b_i^{(2)}$ denote the service time at station i and the first two moments of it, respectively. The switchover times have general distributions. R_i denotes the switchover time following the service of station i. On the classical cyclic polling model we impose the following conditions:

C.1 LAA condition. The "lack of anticipation" (LAA) assumption holds for the arrival process at each station. It ensures to have the "Poisson arrivals see times averages" (PASTA) property (Wolff 1982).

C.2 Mutual independency condition. The arrival processes, the service times and the switchover times are mutually independent.

C.3 Mixed-discipline system condition. Each station can use different service disciplines.

Definition 1: Cycle time: The cycle time of a given station is defined as the time ellapsed between two consecutive server visits to that station. The (polling) cycle time of station i is called i-polling cycle. C_i and c_i denote the i-polling cycle, and its mean, respectively.

Definition 2: Polling epoch, departure epoch: The arrival time of the server to a given station is called polling epoch. Similarly the time when the server finishes the service at the given station and departs is called departure epoch. The polling epoch of station i is called i-polling epoch. Similarly i-departure epoch is

the name of departure epoch of station i. F_i, f_i, $f_i^{(2f)}$ denotes the number of customers at station i, seen at the i-polling epoch, and the first two factorial moments of it, respectively. Analogously M_i, m_i, $m_i^{(2f)}$ denotes the number of customers at station i, seen at the i-departure epoch, and the first two factorial moments of it, respectively,

Definition 3: Service discipline: The service discipline gives a condition on the beginning and the end of the service at the given station. It does not depend on the history of the system and it is work-conserving as well as nonpreemptive.

The most commonly known disciplines are exhaustive, gated, non-exhaustive, semi-exhaustive, binomial-gated, binomial-exhaustive, limited-N and nonpreemtive limited-T (for more details see (Takagi 1986)).

To obtain the expression of the LST of the stationary waiting time still supplementary conditions are necessary on the model.

SC.1 Independence condition. The arrival processes, the service processes, the switchover times and the service disciplines of the model are mutually independent.

SC.2 FIFO condition. The queueing discipline is the First-In-First-Out (FIFO) order at each queue.

Now the properties needed to obtain the expression of the LST of the stationary waiting time can be defined as follows:

W.1 A new arriving i-customers do not affect the time in the system of the previously arrived i-customers, i.e. their waiting and service time. This holds due to the conditions **C.1**, **SC.1** and **SC.2**.

W.2 The waiting time of a tagged customer and the service time of it are independent. This follows from the condition **SC.1**.

As usual the server utilization at station i is denoted by ρ_i, and ρ stands for the total server utilization. The following simple relations hold: $\rho_i = \lambda_i b_i$, and $\rho = \sum_{i=1}^{N} \rho_i$. The Laplace-Stieljes transform (LST) of a nonnegative random variable A is denoted as $A^*(s)$, and for the discrete random variable A its probability-generating function (PGF) is denoted by $\widehat{A}(z)$.

MARKOV REGENERATIVE PROCESS FRAMEWORK

Observe that in our polling model the number of customers at every station, seen at any time can be interpreted as a multivariate $Z(t)$ Markov regenerative process. There exists an underlying $\{(Y(m), U(m)), m \geqq 0\}$ Markov renewal sequence with multidimensional countable state space Ω, where $U(0) = 0$ by definition. Here $Y(m)$ is the number of customers at every stations at the m-th i-polling epoch, while $U(m)$ is the m-th i-polling epoch, i.e. $Z\left(U(m)^+\right) = Y(m)$. The intervals between two consecutive regenerative points correspond to the i-polling cycles. $C(m)$ denotes the m-th i-polling cycle, for every $m > 0$. We call these i-polling cycles simply cycles. Given the state of $Z(t)$ at the regenerative point the stochastic behavior of the system repeats itself independent of the ellapsed polling cycles.

During the analysis of polling models the subject of interest are often quantities taking their values only in some embedded time points. This is the case e.g. when we are interested in the number of customers at each station only at the arrival times of i-customers. They are a kind of embedded processes.

Consider an $\{X(m,l), m \geqq 1, l \geqq 1\}$ stochastic process embedded in the $Z(t)$ Markov regenerative process with state space Ω, where m ($m \geqq 1$) is the index of the polling cycle, and l ($l \geqq 1$) is the index of the embedded time points inside the given polling cycle. We define the following quantities:

$Y \stackrel{def}{=} lim_{m \longrightarrow \infty} Y(m)$,

$C \stackrel{def}{=} lim_{m \longrightarrow \infty} C(m)$,

H_m - the number of the embedded time points inside the m-th polling cycle of $Z(t)$,

$H \stackrel{def}{=} lim_{m \longrightarrow \infty} H_m$,

$X(l) \stackrel{def}{=} lim_{m \longrightarrow \infty} X(m,l)$, in other words it is the value of $X(m,l)$ at the l-th embedded time points ($l \geqq 1$) inside of a random cycle,

$X \stackrel{def}{=} lim_{M \longrightarrow \infty} \frac{\sum_{m=1}^{M} \sum_{l=1}^{H_m} X(m,l)}{\sum_{m=1}^{M} H_m}$, in other words it is the value of $X(m,l)$ at random occurrence inside of a random cycle,

$X_i(l)$- the i-th element of the vector $X(l)$, and finally

X_i - the i-th element of the vector X.

Theorem 1: Assume, that (i) the Y limiting distribution exists, (ii) $E[C] < \infty$, and (iii) $E[H] < \infty$. Then the X_i limiting distribution exists, and the following relation holds:

$$E\left[z^{X_i}\right] = \frac{E\left[\sum_{l=1}^{H} z^{X_i(l)}\right]}{E[H]}. \qquad (1)$$

Proof. Due to (i), the assumption (ii) is equivalent with $E\{C \mid Y = j\} < \infty$ for every $j \in \Omega$. According to theorem 9.30 of (Kulkarni 1995) these ensure the existence of the limiting distribution of $Z(t)$. Due to the markov regenerative property of $Z(t)$ it follows, that a stationary version of $Z(t)$ also exists. In this version of the process the stochastic behavior of each polling cycle is the same, consequently H, for every $l \geqq 1$ the $X(l)$ and X limiting distributions exist, as well as all are stationary distributions. For the limiting distribution of X we can write:

$$P\{X = k\} = \frac{E\left[\sum_{l=1}^{H} P\{X(l) = k\}\right]}{E[H]}.$$

The assumption (iii) ensures, that the distribution is not degenerate. This expression is also valid for X_i, for the PGF of X_i from which we directly get the statement of the theorem.□

Remark 1: The statement of the theorem 1 suggests thinking in stationary random cycles. The PGF (or distribution) of X_i is the average of the "partial" PGFs (or distributions) of $X_i(l)$ over a random cycle. The number of occurrences of $X_i(l)$ in the random cycle equals with the limiting distribution of the number of embedded time points inside of a cycle.

In a stable polling system (i) and (ii) hold. Therefore the stationary distributions of all the necessary measures exist. From now on we assume that the polling system is stable.

ANALYSIS

We introduce several notations for the analysis (they are illustrated on figure 1).

G_i, g_i - the number of customers served at station i during an i-polling cycle, and its mean,

A_i - the number of customers arriving to station i during an i-polling cycle,

$T_i(n)$ - the service completion time of the n-th i-customer during an i-polling cycle ($n = 0, \ldots, G_i$, where $T_i(0)$ is the i-polling epoch by definition),

$A_i(n)$ - the number of customers arriving to station i during the service time of the n-th i-customer ($n = 1, \ldots, G_i + 1$, where $A_i(G_i + 1)$ is the number of customers arriving to station i during the intervisit time of the station i),

$T_i(n,k)$ - the arriving time of the k-th i-customer during the service time of the n-th i-customer ($n = 1, \ldots, G_i + 1$, where $T_i(G_i + 1, k)$ is the arriving time of the k-th i-customer during the intervisit time of the i-polling cycle, $k = 1, \ldots, A_i(n)$).

Theorem 2: (Expression of $\widehat{Q}_i(z)$.) Consider the stable classical nonzero-switchover-times cyclic polling model satisfying conditions **C.1** - **C.3.** For this model the stationary PGF of the number of i-customers at a random point in time is

$$\widehat{Q}_i(z) = \frac{(1-\rho_i)(1-z)B_i^*(\lambda_i - \lambda_i z)}{B_i^*(\lambda_i - \lambda_i z) - z} \cdot \frac{\widehat{M}_i(z) - \widehat{F}_i(z)}{(f_i - m_i)(1-z)}. \quad (2)$$

Proof. Let Q_i, Q_i^d, and Q_i^a denote the number of customers at station i seen at any time, seen by a departing i-customer, and seen by an arriving i-customer, respectively. It is well known, that in our model the stationary Q_i^a and Q_i^d are the same (see e.g. chapter 5. in (Kleinrock 1975)). Additionally we have PASTA (model condition **C.1**), so we can summarize:

$$\widehat{Q}_i(z) = \widehat{Q}_i^a(z) = \widehat{Q}_i^d(z). \quad (3)$$

In the stable model $E[G_i] < \infty$ and $E[A_i] < \infty$ hold, so the relationship (1) can be applied to express $\widehat{Q}_i^d(z)$ and $\widehat{Q}_i^a(z)$. We get:

$$\widehat{Q}_i^d(z) = \frac{E\left[\sum_{n=1}^{G_i} z^{Q_i(T_i(n))}\right]}{E[G_i]}, \quad (4)$$

$$\widehat{Q}_i^a(z) = \frac{E\left[\sum_{n=1}^{G_i+1}\sum_{k=1}^{A_i(n)} z^{Q_i(T_i(n,k))}\right]}{E\left[\sum_{n=1}^{G_i+1} A_i(n)\right]}. \quad (5)$$

We decompose the nominator of (5) as follows:

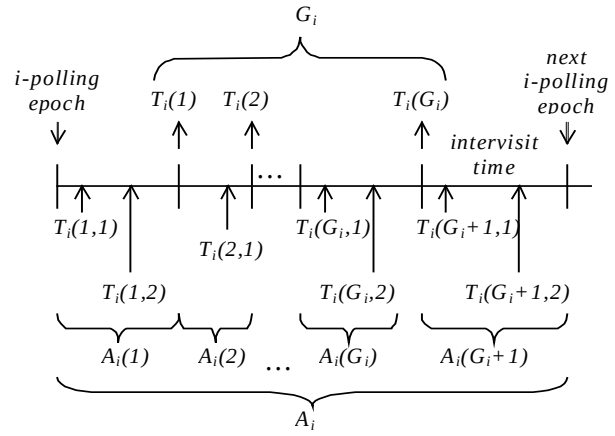


Fig. 1. Notations

$$E\left[\sum_{n=1}^{G_i+1}\sum_{k=1}^{A_i(n)} z^{Q_i(T_i(n,k))}\right] = E\left[\sum_{n=1}^{G_i}\sum_{k=1}^{A_i(n)} z^{Q_i(T_i(n,k))}\right] + E\left[\sum_{k=1}^{A_i(G_i+1)} z^{Q_i(T_i(G_i+1,k))}\right]. \quad (6)$$

During the service of the n-th i-customer the k-th arriving i-customer sees, on one hand, the i-customers left at the service completion time of the $(n-1)$-th i-customer and, on the other hand, $(k-1)$ previously arrived i-customers. Note, that the service starting time of the first i-customer is $T_i(0)$ due to work-conservation property of service disciplines. So the first term of the right side of (6) is:

$$E\left[\sum_{n=1}^{G_i}\sum_{k=1}^{A_i(n)} z^{Q_i(T_i(n,k))}\right] = E\left[\sum_{n=1}^{G_i} z^{Q_i(T_i(n-1))}\sum_{k=1}^{A_i(n)} z^{k-1}\right] = E\left[\sum_{n=1}^{G_i} z^{Q_i(T_i(n-1))}\frac{1-z^{A_i(n)}}{1-z}\right]. \quad (7)$$

Due to Poisson arrivals and general independent service times the terms $Q_i(T_i(n-1))$ and $A_i(n)$ are independent. The arrival process and customer service time are mutual independent (condition **C.2**), so $E\left[z^{A_i(n)}\right] = B_i^*(\lambda_i - \lambda_i z)$ for $n = 1...G_i$. Using these, and (4) it follows:

$$E\left[\sum_{n=1}^{G_i} z^{Q_i(T_i(n-1))}\frac{1-z^{A_i(n)}}{1-z}\right] = \frac{1-B_i^*(\lambda_i-\lambda_i z)}{1-z}. \quad (8)$$

$$E\left[\sum_{n=1}^{G_i} z^{Q_i(T_i(n))} + z^{Q_i(T_i(0))} - z^{Q_i(T_i(G_i))}\right]$$
$$= \frac{1-B_i^*(\lambda_i-\lambda_i z)}{1-z}\left(g_i\widehat{Q}_i^d(z) + \widehat{F}_i(z) - \widehat{M}_i(z)\right).$$

We consider now the second term of the right side of (6). Using the same argument as in (7), and noting that $A_i(G_i+1)$ can depend on $Q_i(T_i(G_i))$, we have:

$$E\left[\sum_{k=1}^{A_i(G_i+1)} z^{Q_i(T_i(G_i+1,k))}\right] \quad (9)$$
$$= E\left[z^{Q_i(T_i(G_i))}\sum_{k=1}^{A_i(G_i+1)} z^{k-1}\right]$$
$$= \frac{1}{1-z}E\left[z^{Q_i(T_i(G_i))} - z^{Q_i(T_i(0))}\right]$$
$$= \frac{\widehat{M}_i(z) - \widehat{F}_i(z)}{1-z}.$$

The stationary mean number of served and arriving i-customers in a polling cycle are equal:

$$g_i = E[G_i] = E[A_i] = E\left[\sum_{n=1}^{G_i+1} A_i(n)\right]. \quad (10)$$

Substituting (8) and (9) into (6) and then into (5), as well as using (10) and (3) we get:

$$\widehat{Q}_i(z) = \frac{(1-B_i^*(\lambda_i-\lambda_i z))}{g_i(1-z)} \quad (11)$$
$$\cdot\left(g_i\widehat{Q}_i(z) + \widehat{F}_i(z) - \widehat{M}_i(z)\right)$$
$$+\frac{\widehat{M}_i(z) - \widehat{F}_i(z)}{g_i(1-z)}.$$

Solving (11) we have:

$$\widehat{Q}_i(z) = \frac{1}{g_i}\frac{B_i^*(\lambda_i-\lambda_i z)}{B_i^*(\lambda_i-\lambda_i z) - z}\left(\widehat{M}_i(z) - \widehat{F}_i(z)\right). \quad (12)$$

We need the following relationship, where s_i denotes the stationary mean service time of the station i:

$$(1-\rho_i)g_i = g_i - \lambda_i b_i g_i = g_i - \lambda_i s_i = f_i - m_i. \quad (13)$$

Now multiplying both the nominator and the denominator of (12) by $(1-\rho_i)(1-z)$, and using (13) we get the statement of the theorem.□

Theorem 3: (Decomposition of Q_i.) Consider the stable classical nonzero-switchover-times cyclic polling model satisfying conditions **C.1** - **C.3.** For this model the stationary number of i-customers in the system can be decomposed as the sum of two independent random variables, as follows:

$$Q_i = Q_i^r + Q_i^{aI}. \quad (14)$$

Proof. Let Q_i^{aI} denotes the number of customers at station i, seen by an i-customer arriving in the intervisit time of station i. Additionally Q_i^r denotes the number of customers in the corresponding M/G/1 queue of station i (having the same arrival rate and customer service time, as station i of the polling system). The first term on the right side of (2) is the Pollaczek-Khinchin formula for Q_i^r (chapter 5. in (Kleinrock 1975)). To interpret the right term we write the PGF of Q_i^{aI} by applying (1):

$$\widehat{Q}_i^{aI}(z) = \frac{E\left[\sum_{k=1}^{A_i(G_i+1)} z^{Q_i(T_i(G_i+1,k))}\right]}{E[A_i(G_i+1)]}. \quad (15)$$

The denominator is the mean number of i-customers arriving in the intervisit time. It is the same as $(f_i - m_i)$. Together with (9) we get:

$$\widehat{Q}_i^{aI}(z) = \frac{\widehat{M}_i(z) - \widehat{F}_i(z)}{(f_i-m_i)(1-z)}. \quad (16)$$

Now it can be seen, that the second term on the right side of (2) equals with (15). So we have:

$$\widehat{Q}_i(z) = \widehat{Q}_i^r(z)\,\widehat{Q}_i^{aI}(z). \quad (17)$$

□

Corollary 1: For the stable classical nonzero-switchover-times cyclic polling model satisfying conditions **C.1** - **C.3**, the mean stationary number of i-customers:

$$E[Q_i] = \rho_i + \frac{\lambda_i^2 b_i^{(2)}}{2(1-\rho_i)} + \frac{f_i^{(2f)} - m_i^{(2f)}}{2(f_i-m_i)}. \quad (18)$$

Proof. It can be derived from (2).□

Theorem 4: (Expression of $W_i^*(s)$.) Consider the stable classical nonzero-switchover-times cyclic polling model satisfying conditions **C.1** - **C.3** and supplementary conditions **SC.1** - **SC.2.** For this model the LST of the stationary waiting time of the customers at station i is

$$W_i^*(s) = \frac{s(1-\rho_i)}{s-\lambda_i+\lambda_i B_i^*(s)} \quad (19)$$
$$\cdot\frac{\widehat{M}_i\left(1-\frac{s}{\lambda_i}\right) - \widehat{F}_i\left(1-\frac{s}{\lambda_i}\right)}{\frac{s}{\lambda_i}(f_i-m_i)}.$$

Proof. We argue, that due to FIFO queueing discipline the number of i-customers left in the system at service completion of a tagged i-customers is equal with the number of i-customers arrived during the sojourn time of that i-customer in the system. Due to the mutual independency condition **C.2,** and properties **W.1** and **W.2** for $\widehat{Q}_i^d(z)$ we get:

$$\widehat{Q}_i^d(z) = W_i^*(\lambda_i-\lambda_i z)\,B_i^*(\lambda_i-\lambda_i z). \quad (20)$$

Substituting $s = \lambda_i - \lambda_i z$ and rearranging (20) gives the basic relationship for $W_i^*(s)$:

$$W_i^*(s) = \frac{\widehat{Q}_i^d\left(1-\frac{s}{\lambda_i}\right)}{B_i^*(s)}.$$

Taking into account (3) and by substituting (2) we get the statement of the theorem.□

Theorem 5: (Decomposition of W_i.) Consider the stable classical nonzero-switchover-times cyclic polling model satisfying conditions **C.1** - **C.3** and supplementary conditions **SC.1** - **SC.2.** For this model the LST of the stationary waiting time of the customers at station i can be decomposed as the sum of two independent random variables, one being the stationary waiting time in the corresponding standard M/G/1 queue, and the other relates to the stationary number of the i-customers seen by the i-customers arriving in the intervisit time of the station i.

Proof. Let W_i^r denote the waiting time in the corresponding M/G/1 queue of station i. The left term on the right side of (19) is the Pollaczek-Khinchin formula for the waiting time of the customers in the standard M/G/1 system (Kleinrock 1975), i.e. $W_i^{r*}(s)$. Using this and (16):

$$W_i^*(s) = W_i^{r*}(s)\,\widehat{Q}_i^{aI}\left(1 - \frac{s}{\lambda_i}\right). \qquad (21)$$

□

Corollary 2: For the stable classical nonzero-switchover-times cyclic polling model satisfying conditions **C.1** - **C.3** and supplementary conditions **SC.1** - **SC.2**, the mean stationary waiting time is

$$E[W_i] = \frac{\lambda_i b_i^{(2)}}{2(1-\rho_i)} + \frac{f_i^{(2f)} - m_i^{(2f)}}{2\lambda_i (f_i - m_i)}\ . \qquad (22)$$

Proof. It can be derived from (19).□

SUPPLEMENTS

Corollary 3: Consider the stable classical nonzero-switchover-times polling model satisfying conditions **C.1** - **C.3** and the supplementary conditions **SC.1** - **SC.2**. If the

$$\widehat{F}_i(z) \Rightarrow \widehat{M}_i(z)$$

relation can be expressed explicitly, then the analysis reduces to the one of finding the stationary number of customers at the i-polling epochs.

Proof. Having $\widehat{M}_i(z)$ as an expression of $\widehat{F}_i(z)$, substituting it and it's derivatives in the (2), (18), (19) and (22) results, that only $\widehat{F}_i(z)$ and it's derivatives remains unknown. After solving the system for $\widehat{F}_i(z)$ or it's derivatives, we can substitute them in that equalities to get $\widehat{Q}_i(z)$, $E[Q_i]$, $W_i^*(s)$ and $E[W_i]$.□

Theorem 6: Consider the stable classical zero-switchover-times polling model satisfying conditions **C.1** - **C.3** and optionally the supplementary conditions **SC.1** - **SC.2**. Then all the previous results, i.e. the theorems 2, 3, 4 and 5 and the corollaries 1, 2, and 3 are also valid for the zero-switchover-times polling model .

Proof. Let Ω^1 denote the nonzero-switchover-times polling model. The polling model, where all the input parameters are the same except the switchover times is referred as corresponding zero-switchover-times polling model. This is denoted by Ω^0. Let us introduce a modified zero-switchover-times polling model, say Ω^{0+},

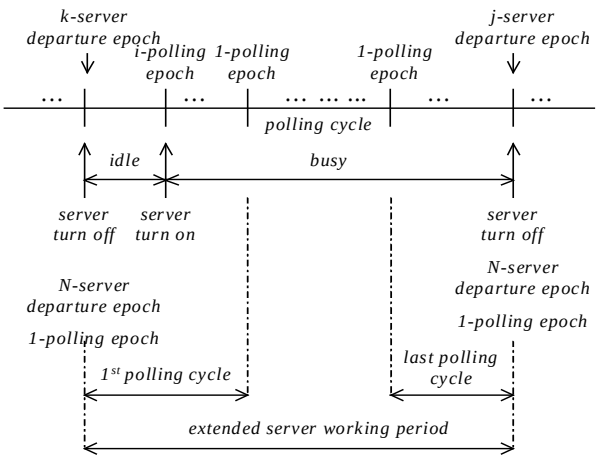


Fig. 2. Modified zero-switchover-times model

where the server working period is virtually extended before the server turn on and after the server turn off (figure 2). This is done on such a way, that the idle period is assigned with the first polling cycle of the next server working period. Additionally some server visits are inserted, if necessary. So the server starts with visiting the first queue (before the idle period), and finishes by visiting the N-th queue (again before the next idle period). Due to these extensions the extended server working period composes a number of complete polling cycles. Any change in the number of customers in the Ω^{0+} model has a corresponding event at the same time in the Ω^0 model, because during the extended visits and the idle time period no arrival and no service occurs. Therefore in any time between the server turn on and next turn off the number of customers in the system are also the same in both the Ω^0 and Ω^{0+} models. It is true for $\widehat{Q}_i^d(z)$, $\widehat{Q}_i^{aI}(z)$ and $\widehat{Q}_i^a(z)$, therefore also for $\widehat{Q}_i(z)$ due to PASTA.

Unfortunately the distributions of the random variables F_i and M_i in the Ω^{0+} model differs from their counterpairs in the Ω^0 model. This is exactly because of the extended server visits, i.e. there can be more number of occurences of i-polling and i-departure epochs in the extended server working period. It follows, that in these epochs the number of occurences of no customer present is also different in the Ω^0 and Ω^{0+} models. However this difference in the Ω^0 and Ω^{0+} models affects $\widehat{F}_i(z)$, $\widehat{M}_i(z)$, f_i, and m_i on the same way. Hence due to the substraction and division this difference also vanishes in $\frac{\widehat{M}_i(z)-\widehat{F}_i(z)}{(f_i-m_i)}$. Therefore the meaning of this term of the right side of (2) in the Ω^{0+} model is the same as that one in the the Ω^0 model.□

Corollary 4: For nonzero-switchover-times model the formula (19) can be further simplified as follows:

$$W_i^*(s) = \frac{1}{c_i}\,\frac{\widehat{M}_i\left(1-\frac{s}{\lambda_i}\right) - \widehat{F}_i\left(1-\frac{s}{\lambda_i}\right)}{s - \lambda_i + \lambda_i B_i^*(s)}. \qquad (23)$$

Proof. The statement follows by applying the equilibrium relation: $\frac{g_i}{\lambda_i} = c_i$, and (13).

WAITING TIME EXAMPLES

In the next we demonstrate the use of corollary 3.

Example *1:* Exhaustive service discipline

For exhaustive service discipline $\widehat{M}_i(z) = 1$, and $m_i = 0$. Substituting these values in (19) results:

$$W_i^*(s) = \frac{s(1-\rho_i)}{s-\lambda_i+\lambda_i B_i^*(s)} \frac{1-\widehat{F}_i\left(1-\frac{s}{\lambda_i}\right)}{\left(\frac{s}{\lambda_i}\right) f_i}. \quad (24)$$

This equation also can be found as equation 3 in (Srinivasan et al. 1995) for the zero-switchover-times model. For the nonzero-switchover-times model applying from (23) we get:

$$W_i^*(s) = \frac{1}{c_i} \frac{1-\widehat{F}_i\left(1-\frac{s}{\lambda_i}\right)}{s-\lambda_i+\lambda_i B_i^*(s)}. \quad (25)$$

This equation also can be found as equation 4.32, p. 80, in (Takagi 1986).

Example *2:* Gated service discipline

Due to the definition of the service strategy the service time of the station i is $S_i^*(s) = \widehat{F}_i(B_i^*(s))$. Additionally M_i equals with the number of i-customers arriving during S_i. In other words:

$$\widehat{M}_i(z) = \widehat{F}_i(B_i^*(\lambda_i - \lambda_i z)). \quad (26)$$

Similarly $m_i = \rho_i f_i$. Substituting these values in (19), for the zero-switchover-times model we get:

$$W_i^*(s) = \frac{s(1-\rho_i)}{s-\lambda_i+\lambda_i B_i^*(s)} \frac{\widehat{F}_i(B_i^*(s)) - \widehat{F}_i\left(1-\frac{s}{\lambda_i}\right)}{\left(\frac{s}{\lambda_i}\right)(1-\rho_i) f_i}. \quad (27)$$

For the nonzero-switchover-times model using (26) in (23) gives:

$$W_i^*(s) = \frac{1}{c_i} \frac{\widehat{F}_i(B_i^*(s)) - \widehat{F}_i\left(1-\frac{s}{\lambda_i}\right)}{s-\lambda_i+\lambda_i B_i^*(s)}. \quad (28)$$

This equation also can be found as equation 5.45, p. 110, in (Takagi 1986).

Example *3:* Non-exhaustive service discipline

At the i-departure epoch, the server just completed one service with probability $P(F_i \geqq 1)$, or no service with probability $1 - P(F_i \geqq 1)$, because the discipline has only these two possibilities. That implies, that during an i-polling cycle $g_i = P(F_i \geqq 1)$. Conditioning $\widehat{F}_i(z)$ on $F_i \geqq 1$, and denoting it by F_i^+, that is $P\left(F_i^+ = k\right) = \frac{P(F_i=k)}{P\left(F_i \geqq 1\right)}$ $(k = 1, 2, ...)$, it follows that $\widehat{F}_i^+(z) = \frac{\widehat{F}_i(z)-(1-g_i)}{g_i}$.

If $F_i \geqq 1$, then the PGF of the arriving i-customers during the service time of the one i-customer served is $B_i^*(\lambda_i - \lambda_i z)$. This is independent from $\widehat{F}_i^+(z)$, so after some straightforward manipulation we get:

$$\widehat{M}_i(z) = \frac{B_i^*(\lambda_i - \lambda_i z)}{z}\left(\widehat{F}_i(z) - (1-g_i)\right) + (1-g_i). \quad (29)$$

Similarly for m_i we get: $m_i = g_i f_i - g_i(1-\rho_i)$. Substituting these values in (19) gives:

$$W_i^*(s) = \frac{s(1-\rho_i)}{s-\lambda_i+\lambda_i B_i^*(s)} \cdot \frac{\frac{B_i^*(s)-\left(1-\frac{s}{\lambda_i}\right)}{1-\frac{s}{\lambda_i}}\left(\widehat{F}_i\left(1-\frac{s}{\lambda_i}\right) - (1-g_i)\right)}{\left(\frac{s}{\lambda_i}\right)((1-g_i) f_i - g_i(1-\rho_i))}. \quad (30)$$

CONCLUSION

We have presented an alternative analysis of the classical $M/G/1$ cyclic polling system. The analysis was based on a new Markov regenerative framework, which enabled to carry out the analysis on unified, service discipline independent way. The MRP framework provides a general way of thinking in stationary random cycles and of relating stationary measures.

We have obtained the general expressions for the stationary number of customers (expression (2) and (18)), and for the stationary waiting time (expressions (19), (22) and (23)). These results except (23) are valid for both the nonzero- and the zero-switchover-times models. The expressions can be used to simplify the analysis of the relating polling models (corollary 3). We have demonstrated this methodology on several known service disciplines in the previous chapter.

References

[1] Borst S. C. and O. J. Boxma. 1997. "Polling models with and without switchover times." *Operations Research* 45, 536-543.

[2] Fuhrmann S. W. and R. B. Cooper. 1985. "Stochastic Decompositions in the M/G/1 Queue with Generalized Vacations." *Operations Research* 33, 1117-1129.

[3] Kleinrock L. 1975. *Queuing Systems. Vol I: Theory.* John Wiley.

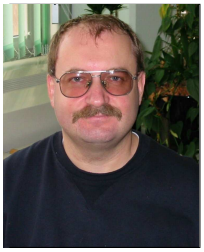
[4] Kuehn, P.J. 1979. "Multiqueue Systems with Nonexhaustive Cyclic Service." *The Bell System Technical Journal* 58, 671-698.

[5] Kulkarni, V. G. 1995. *Modeling and Analysis of Stochastic Systems.* Chapman & Hall.

[6] Srinivasan, M. M.; S.-C. Niu and R. B. Cooper. 1995. "Relating Polling Models with Zero and Nonzero Schwitchover Times." *Queueing Systems* 19, 149-168.

[7] Takagi H. 1986. *Analysis of Polling Systems.* MIT Press.

[8] Wolff R. W. 1982. "Poisson Arrivals See Times Averages." *Operations Research* 30, 223-231.



ZSOLT SAFFER was born in Keszthely, Hungary. He studied electrical engineering on Technical University (TU) of Budapest and obtained Msc. degree in 1989. He worked some years at Alcatel Austria before in 1997 he joined to Philips Austria. Now he also takes part in PhD. course at the Telecommunication Department of TU Budapest.

MULTI-SERVER LOSS QUEUEING SYSTEM OPERATING IN RANDOM ENVIRONMENT

Chesoong Kim
Department of Industrial Engineering
Sangji University
Wonju, Kangwon, 220-702, Korea
E-mail: dowoo@sangji.ac.kr

A.N. Dudin
V.I. Klimenok
V.V. Khramova
Department of Applied Mathematics and Computer Science
Belarusian State University
4, Nezavisimosti Ave., Minsk-30, 220030, Belarus
E-mail: dudin@bsu.by

Sang Cheon Lee
Division of Industrial & Systems Engineering
ERI Gyeong Sang National University Jinju, 660-701, Korea
E-mail: sclee@gsnu.ac.kr

We consider the $BMAP/PH/N/0$ queueing system operating in finite state Markovian random environment. Disciplines of customers admission are partial admission and complete rejection. The stationary distribution of the number of customers in the system is calculated. Loss probability and other performance measures are derived. Illustrative numerical examples are presented. They show effect of admission strategy and quality of the system approximation by simpler models.

KEYWORDS

$BMAP/PH/N/0$ queueing model, stationary state distribution, loss probability.

INTRODUCTION

Well-known Erlang's loss model $M/M/N/0$ was a good mathematical model for classic telephone networks. This is surprising that formula for a loss probability in the $M/M/N/0$ queue serves for needs of the practical engineering until now. However, the flows in the modern telecommunication networks have lost the nice properties of their predecessors in the old classic networks. In opposite to the stationary Poisson input (stationary ordinary input with no aftereffect), the modern real life flows are non-stationary, group and correlated. The $BMAP$ (Batch Markovian Arrival Process) is one of the appropriate models of such inputs. The $BMAP$ was introduced as a versatile Markovian point process ($VMPP$) by M.F. Neuts in the 1970th. The original development of $VMPP$ contained extensive notations; however these notations were simplified greatly in (Lucantoni 1991) and ever since this process bears the name $BMAP$. The Erlang loss model for the case of the $BMAP$ input was investigated in (Klimenok 1999). The extension of the $BMAP/M/N/0$ model to the case of PH (Phase-type) service time distribution was considered in (Klimenok et al. 2005). In practical systems, the input and service processes are not absolutely stable, they are influenced by external factors, e.g., the different level of the noise in the transmission channel, hardware degradation and recovering, change of the distance by a mobile user from the base station, etc. In mathematical modeling of such systems the influence of external factors can be taken into account by using the queues operating in random environment. The $BMAP/M/N/0$ model operating in Markov random environment was investigated in (Kim et al. 2006). Here we investigate the extension of this model to the case of PH service time distribution.

MATHEMATICAL MODEL

We consider an N - server queueing system. The behavior of the system depends on the state of the stochastic process (random environment) r_t, $t \geq 0$, which is assumed to be an irreducible continuous time Markov chain with the state space $\{1, \dots, R\}$, $R \geq 2$, and the infinitesimal generator Q. The input flow into the system is the following modification of the $BMAP$. In this input flow, the batch arrivals are directed by the process ν_t, $t \geq 0$ (the directing process) with the state space $\{0, 1, \dots, W\}$. Under the fixed state r of the random environment, this process behaves as an irreducible continuous time Markov chain. Transitions of the chain ν_t, $t \geq 0$, which are accompanied by arrival of k-size batch, are described by the matrices $D_k^{(r)}$, $k \geq 0$, $r = \overline{1, R}$, with the generating function $D^{(r)}(z) = \sum_{k=0}^{\infty} D_k{}^{(r)} z^k$. The matrix $D^{(r)}(1)$ is the irreducible generator for all $r = \overline{1, R}$. Under the fixed state r of the random environment, the average intensity $\lambda^{(r)}$ (fundamental rate) of the $BMAP$ is defined as $\lambda^{(r)} = \boldsymbol{\theta}^{(r)}(D^{(r)}(z))'|_{z=1}\mathbf{e}$, and the intensity $\lambda_b^{(r)}$ of batch arrivals is defined as $\lambda_b^{(r)} = \boldsymbol{\theta}^{(r)}(-D_0^{(r)})\mathbf{e}$. Here $\boldsymbol{\theta}^{(r)}$ is the solution to the equations $\boldsymbol{\theta}^{(r)} D^{(r)}(1) = \mathbf{0}$, $\boldsymbol{\theta}^{(r)}\mathbf{e} = 1$, $\mathbf{e}$ is a column vector of appropriate size consisting of 1's. The vari-

ation coefficient $c_{var}^{(r)}$ of intervals between batch arrivals is given by $(c_{var}^{(r)})^2 = 2\lambda_b^{(r)}\boldsymbol{\theta}^{(r)}(-D_0^{(r)})^{-1}\mathbf{e} - 1$ while the correlation coefficient $c_{cor}^{(r)}$ of intervals between successive batch arrivals is calculated as $c_{cor}^{(r)} = (\lambda_b^{(r)}\boldsymbol{\theta}^{(r)}(-D_0^{(r)})^{-1}(D^{(r)}(1) - D_0^{(r)})(-D_0^{(r)})^{-1}\mathbf{e} - 1)/$ $/(c_{var}^{(r)})^2$. The state of the process ν_t, $t \geq 0$, is not changed at the epochs of the process r_t, $t \geq 0$, transitions.

The service process is defined by the modification of the phase (PH)-type service time distribution. Service time is interpreted as the time until the irreducible continuous time Markov chain m_t, $t \geq 0$, with the state space $\{1, \ldots, M+1\}$ reaches the absorbing state $M+1$. Under the fixed value r of the random environment, transitions of the chain m_t, $t \geq 0$, within the state space $\{1, \ldots, M\}$ are defined by the irreducible sub-generator $S^{(r)}$ while the intensities of transition into the absorbing state are defined by the vector $\boldsymbol{S_0}^{(r)} = -S^{(r)}\mathbf{e}$. At the service beginning epoch, the state of the process m_t, $t \geq 0$, is chosen according to the probabilistic row vector $\boldsymbol{\beta}^{(r)}$, $r = \overline{1,R}$. It is assumed that the state of the process m_t, $t \geq 0$, is not changed at the epochs of the process r_t, $t \geq 0$, transitions. Just the exponentially distributed sojourn time of the process m_t, $t \geq 0$, in the current state is re-started with a new intensity defined by the sub-generator corresponding to the new state of the random environment r_t, $t \geq 0$. The system under consideration has finite state space. Due to possibility of group arrivals, it can occur that there are free servers in the system at arrival epoch, however the number of these servers is less than the number of customers in a group. In such situation the acceptance of customers to the system is realized according to the partial admission (PA) discipline (only a part of the group corresponding to the number of free servers is allowed to enter the system while the rest of the group is lost) or the complete rejection (CR) discipline (a whole group leaves the system if the number of free servers is less than the number of customers in the group). Our aim is to calculate the stationary state distribution and main performance measures of the described queueing model.

STATIONARY STATE DISTRIBUTION

It is easy to see that operation of the considered queueing system is described in terms of the irreducible continuous-time Markov chain:

$$\xi_t = \{n_t, r_t, \nu_t, m_t^{(1)}, \ldots, m_t^{(n_t)}\},\ t \geq 0,$$

where n_t is the number of customers in the system (the number of busy servers), r_t is the state of random environment, $r_t = \overline{1,R}$, ν_t is the state of the $BMAP$ directing process and $m_t^{(n)}$ is the state of PH service process on the nth server, $n_t = \overline{0,N}, \nu_t = \overline{0,W}, m_t^{(n)} = \overline{1,M}, n = \overline{1,N}$, at epoch t, $t \geq 0$. Enumerate the states of the chain ξ_t, $t \geq 0$, in the lexicographic order and form the row vectors $\mathbf{p}_n$, $n = \overline{0,N}$, of probabilities corresponding to the state n of the first component of the process ξ_t, $t \geq 0$. Denote also $\mathbf{p} = (\mathbf{p}_0, .., \mathbf{p}_N)$. Vector $\mathbf{p}$ satisfies the system of linear algebraic equations of the form:

$$\mathbf{p}A = \mathbf{0},\ \mathbf{p}\mathbf{e} = 1, \tag{1}$$

where A is the infinitesimal generator of the Markov chain ξ_t, $t \geq 0$.

Let

- $\mathbf{e}_n$ be a column vector of size n, consisting of 1's;
- I (O) be an identity (zero) matrix of appropriate dimension (when needed we will identify the dimension of this matrix with a suffix);
- $diag\{a_l,\ l = \overline{1,L}\}$ be a diagonal matrix with diagonal entries a_l;
- $\otimes$ and $\oplus$ be the symbols of Kronecker product and sum of matrices;
- $\Omega^{\otimes l} = \underbrace{\Omega \otimes \ldots \otimes \Omega}_{l},\ l \geq 1,\ \Omega^{\otimes 0} = 1;$
- $\Omega^{\oplus l} = \sum_{m=0}^{l-1} I_{n^m} \otimes \Omega \otimes I_{n^{l-m-1}},\ l \geq 1;$
- $D(z) = \sum_{k=0}^{\infty} diag\{D_k^{(r)}, r = \overline{1,R}\}z^k;$
- $\bar{W} = W + 1;$
- $\mathcal{D}_k^{(n)} = diag\{D_k^{(r)} \otimes I_{M^n},\ r = \overline{1,R}\},\ k \geq 0,\ n = \overline{0,N};$
- $\mathcal{B}_l^{(n)} = diag\{I_{\bar{W}} \otimes I_{M^n} \otimes \left(\boldsymbol{\beta}^{(r)}\right)^{\otimes l},\ r = \overline{1,R}\},\ n = \overline{1,N};$
- $\mathcal{Q}^{(n)} = Q \otimes I_{\bar{W}} \otimes I_{M^n},\ n = \overline{0,N};$
- $\mathcal{S}^{(n)} = diag\{I_{\bar{W}} \otimes \left(S^{(r)}\right)^{\oplus n},\ r = \overline{1,R}\},\ n = \overline{1,N};$
- $\mathcal{S}_0^{(n)} = diag\{I_{\bar{W}} \otimes \left(\boldsymbol{S_0}^{(r)}\right)^{\oplus n}, r = \overline{1,R}\},\ n = \overline{1,N};$
- $\mathcal{C}^{(n)} = \mathcal{Q}^{(n)} + \mathcal{D}_0^{(n)} + \mathcal{S}^{(n)},\ n = \overline{0,N-1}.$

Lemma 1. Infinitesimal generator A of the Markov chain $\xi_t, t \geq 0$, has the following block structure:

$$A = (A_{n,n'})_{n,n'=\overline{0,N}} =$$

$$= \begin{pmatrix} \mathcal{C}^{(0)} & \mathcal{D}_1^{(0)}\mathcal{B}_1^{(0)} & \cdots & \mathcal{D}_{N-1}^{(0)}\mathcal{B}_{N-1}^{(0)} & \bar{\mathcal{D}}_N^{(0)}\mathcal{B}_N^{(0)} \\ \mathcal{S}_0^{(1)} & \mathcal{C}^{(1)} & \cdots & \mathcal{D}_{N-2}^{(1)}\mathcal{B}_{N-2}^{(1)} & \bar{\mathcal{D}}_{N-1}^{(1)}\mathcal{B}_{N-1}^{(1)} \\ O & \mathcal{S}_0^{(2)} & \cdots & \mathcal{D}_{N-3}^{(2)}\mathcal{B}_{N-3}^{(2)} & \bar{\mathcal{D}}_{N-2}^{(2)}\mathcal{B}_{N-2}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ O & O & \cdots & \mathcal{C}^{(N-1)} & \bar{\mathcal{D}}_1^{(N-1)}\mathcal{B}_1^{(N-1)} \\ O & O & \cdots & \mathcal{S}_0^{(N)} & \bar{\mathcal{C}}^{(N)} \end{pmatrix},$$

where

(i) in case of PA discipline

$$\bar{\mathcal{D}}_{N-n}^{(n)} = \sum_{k=0}^{\infty} \mathcal{D}_{N+k-n}^{(n)},\ n = \overline{0,N-1};$$

(ii) in case of CR discipline

$$\bar{\mathcal{D}}_{N-n}^{(n)} = \mathcal{D}_N^{(n)},\ n = \overline{0,N-1};$$

and

$$\bar{\mathcal{C}}^{(N)} = \mathcal{Q}^{(N)} + \sum_{k=0}^{\infty} \mathcal{D}_k^{(N)} + \mathcal{S}^{(N)}$$

for both disciplines.

To solve system (1), we use the effective stable procedure (Klimenok et al. 2005) based on the the special structure of the matrix A (it is upper block Hessenberg) and probabilistic meaning of unknown vector $\mathbf{p}$.

PERFORMANCE MEASURES

Having the vector $\mathbf{p}$ been calculated, we are able to calculate the main performance measure of the considered model. It is the probability P_{loss} that an arbitrary customer will be lost (loss probability).

Theorem 1. Loss probability P_{loss} is calculated as follows

(i) in case of PA discipline

$$P_{loss} = 1 - \lambda^{-1} \sum_{i=0}^{N-1} \mathbf{p}_i \sum_{k=0}^{N-i} (k+i-N) \mathcal{D}_k^{(i)} \mathbf{e},$$

(ii) in case of CR discipline

$$P_{loss} = 1 - \lambda^{-1} \sum_{i=0}^{N-1} \mathbf{p}_i \sum_{k=0}^{N-i} k \mathcal{D}_k^{(i)} \mathbf{e},$$

where $\lambda = \mathbf{x} D(z)|'_{z=1} \mathbf{e}$ and $\mathbf{x}$ is an invariant vector of the matrix $Q \otimes I_{\bar{W}} + D(1)$.

A number of other stationary performance measures of the considered model are calculated as follows:

- The probability to see n busy servers

$$p_n = \mathbf{p}_n \mathbf{e},\ n = \overline{0, N};$$

- The mean number of busy servers

$$N_{busy} = \sum_{n=1}^{N} n \mathbf{p}_n \mathbf{e};$$

- The mean number N_{idle} of idle servers

$$N_{idle} = N - N_{busy};$$

- The vector $\hat{\mathbf{p}}_n$ whose $(\bar{W}(r-1)+\nu+1)$-th entry is the joint probability to see n busy servers, the random environment in the state r and the process ν_t in the state ν

$$\hat{\mathbf{p}}_n = \mathbf{p}_n (I_{R\bar{W}} \otimes \mathbf{e}_{M^n}),\ n = \overline{0, N};$$

- The vector $\mathbf{p}^{(a)}(n)$ whose $(\bar{W}(r-1)+\nu+1)$-th entry is the joint probability that an arbitrary arriving call sees n busy servers and the random environment in the state r and the state of the process ν_t becomes ν after arrival epoch

$$\mathbf{p}^{(a)}(n) = \lambda^{-1} \hat{\mathbf{p}}_n D(z)'|_{z=1},\ n = \overline{0, N},$$

- The probability $p^{(a)}(n)$ that an arbitrary arriving call sees n busy servers

$$p^{(a)}(n) = \mathbf{p}^{(a)}(n) \mathbf{e},\ n = \overline{0, N};$$

- The vector ${\mathbf{p}_b}^{(a)}(n)$ whose $(\bar{W}(r-1)+\nu+1)$-th entry is the joint probability that an arbitrary arriving batch sees n busy servers and the random environment in the state r and the state of the process ν_t becomes ν after arrival epoch

$${\mathbf{p}_b}^{(a)}(n) = {\lambda_b}^{-1} \hat{\mathbf{p}}_n (D(1) - D(0)),\ n = \overline{0, N},$$

where $\lambda_b = \mathbf{x}(D(1) - D(0))\mathbf{e}$;

- The probability ${p_b}^{(a)}(n)$ that an arbitrary arriving batch sees n busy servers

$${p_b}^{(a)}(n) = {\mathbf{p}_b}^{(a)}(n) \mathbf{e},\ n = \overline{0, N}.$$

NUMERICAL EXAMPLES

Let us present the results of two experiments. We consider the $BMAP/PH/N/0$ system operating in two-states RE. It means that the system has $R = 2$ operation modes. We assume that the number of servers N is equal to 5. Let the generator of the random environment be as follows $Q = \begin{pmatrix} -5 & 5 \\ 15 & -15 \end{pmatrix}$.

The goal of the first experiment is to analyze the influence of the customers admission discipline on the value of the loss probability. In the second experiment we compare the loss probability in the original system in RE and more simple exponential queueing systems, which can be considered as "engineering" analogs of original system.

The first experiment.

Consider the MAP_1 and MAP_2 which are characterized by the matrices

$${D_0}^{(1)} = \begin{pmatrix} -15.7327 & 0.6062 & 0.5924 \\ 0.5178 & -2.2897 & 0.4679 \\ 0.5971 & 0.5653 & -1.9597 \end{pmatrix},$$

$$D^{(1)} = \begin{pmatrix} 14.1502 & 0.3021 & 0.0818 \\ 0.1071 & 1.0323 & 0.1646 \\ 0.0858 & 0.1979 & 0.5136 \end{pmatrix},$$

$${D_0}^{(2)} = \begin{pmatrix} -25.5398 & 0.3933 & 0.3612 \\ 0.1452 & -2.2322 & 0.2 \\ 0.2959 & 0.3875 & -1.7526 \end{pmatrix},$$

$$D^{(2)} = \begin{pmatrix} 24.2421 & 0.4669 & 0.0763 \\ 0.0341 & 1.6669 & 0.186 \\ 0.0091 & 0.2555 & 0.8046 \end{pmatrix}.$$

Basing on these MAPs we construct $BMAP_1$ and $BMAP_2$ which describe the input flow under the first and the second operation mode correspondingly. We assume that $BMAP_r$ is defined by the matrices $D_k^{(r)}, k = \overline{1, 10}, r = \overline{1, 2}$. To build these matrices, we follow such a way. The matrix $D_0^{(r)}$ is

the same as one in the correspondent MAP and the rest of the matrices are calculated by $D_k^{(r)} = D^{(r)} q^{k-1}(1-q)/(1-q^{10})$, $k = \overline{1,10}$, $r = \overline{1,2}$, where $q = 0.8$.

$BMAP_1$ has fundamental rate $\lambda^{(1)} = 18.98$, the variation coefficient $c_{var}^{(1)} = 2$ and the correlation coefficient $c_{cor}^{(1)} = 0.2$. The PH_1 service is described by the vector $\boldsymbol{\beta}^{(1)} = (1, 0)$ and the matrix $S^{(1)} = \begin{pmatrix} -20 & 20 \\ 0 & -20 \end{pmatrix}$. It means that the service time has Erlangian distribution of order 2 with mean rate $\mu^{(1)} = 10$ and the squared correlation coefficient $(c_{var}^{(1)})^2 = 0.5$.

For $BMAP_2$, $\lambda^{(2)} = 18.98$, $c_{var}^{(2)} = 2$, $c_{cor}^{(2)} = 0.3$. The PH_2 service is described by the parameters $\boldsymbol{\beta}^{(2)} = (0.2, 0.8)$, $S^{(2)} = \begin{pmatrix} -30 & 5 \\ 5 & -40 \end{pmatrix}$. The mean rate of service $\mu^{(2)} = 31.76$ and the squared correlation coefficient $(c_{var}^{(2)})^2 = 1.07$.

Figure 1 shows the dependence of the loss probability P_{loss} on the value $\rho = \lambda/N\mu$ in cases of partial admission (PA) and complete rejection (CR) disciplines. Here μ is defined as $\mu = \mathbf{q} diag\{\mu^{(r)}, r = \overline{1,R}\}\mathbf{e}$, where $\mathbf{q}$ is an invariant vector of the matrix Q. Note that the value ρ can be considered as an estimate of the system load. Looking at figure 1 we can see that the loss probability P_{loss} depends on the customers admission discipline and the discipline CR gives higher value of P_{loss} than the discipline PA.

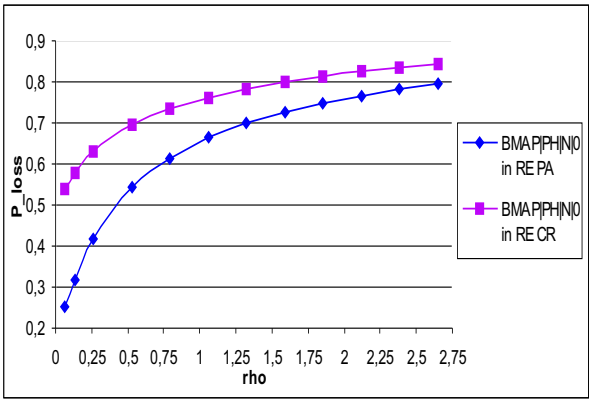


Figure 1. Loss probability for disciplines of partial admission and complete rejection

The second experiment.

In this experiment, we compare the loss probability in the original system in RE and more simple queueing systems which can be considered as exponential analogs of the original system.

In the first part of the experiment we deal with ordinary input flows.

The first mode is described as follows. The MAP_3 input is defined by the matrices

$$D_0{}^{(1)} = \begin{pmatrix} -7.8663 & 0.3031 & 0.2962 \\ 0.2589 & -1.1448 & 0.2339 \\ 0.2985 & 0.2826 & -0.9798 \end{pmatrix},$$

$$D^{(1)} = \begin{pmatrix} 7.0751 & 0.151 & 0.0409 \\ 0.0535 & 0.5161 & 0.0824 \\ 0.0429 & 0.099 & 0.2568 \end{pmatrix}.$$

For this MAP, $\lambda^{(3)} = 2.5$, $c_{var}^{(3)} = 2$, $c_{cor}^{(3)} = 0.2$. The PH_3 service is described by the matrices $\boldsymbol{\beta}^{(3)} = (1, 0)$, $S^{(3)} = \begin{pmatrix} -1 & 1 \\ 0 & -1 \end{pmatrix}$. The mean rate of service is $\mu^{(3)} = 0.5$, the squared variation coefficient $(c_{var}^{(3)})^2 = 0.5$.

The parameters defining the second mode are as follows.

$$D_0{}^{(2)} = \begin{pmatrix} -12.7699 & 0.1967 & 0.1806 \\ 0.0726 & -1.1161 & 0.1 \\ 0.1479 & 0.1937 & -0.8763 \end{pmatrix},$$

$$D^{(2)} = \begin{pmatrix} 12.1211 & 0.2334 & 0.0381 \\ 0.0171 & 0.8334 & 0.093 \\ 0.0045 & 0.1277 & 0.4025 \end{pmatrix}.$$

For this MAP, $\lambda^{(4)} = 2.5$, $c_{var}^{(4)} = 2$, $c_{cor}^{(4)} = 0.3$. The PH_4 service is described by the parameters $\boldsymbol{\beta}^{(4)} = (0.2, 0.8)$, $S^{(4)} = \begin{pmatrix} -24 & 5 \\ 5 & -35 \end{pmatrix}$. The mean rate of service is $\mu^{(4)} = 26.12$, the squared variation coefficient $(c_{var}^{(4)})^2 = 1.11$.

The first type analog is the system $M/M/N/0$ in the RE. It differs from the original system by assumption that input flows are stationary Poisson ones whose intensities are equal to fundamental rate of corresponding MAPs in the original system. The second type analog is the Erlang system $M/M/N/0$ whose parameters are obtained by means of averaging the corresponding parameters of just described system $M/M/N/0$ in the RE according to the stationary distribution of the RE.

Figure 2 shows the dependence of the loss probability on the value $\rho = \lambda/N\mu$ in the original system - $MAP/PH/N/0$ in RE, in the first type analog - $M/M/N/0$ system in RE and the second type analog - $M/M/N/0$ system. Note that in case $M/M/N/0$ system in RE and $M/M/N/0$ system the value ρ is the exact system load.

In the second part of the current experiment we investigate the behavior of the loss probability in original system in RE and its exponential analogs in case of group input. The first type analog is the system in RE with group Poisson input and exponentially distributed service time. The second type analog is the system with group Poisson input and exponentially distributed service time.

Here we consider two disciplines of admission: partial admission and complete rejection.

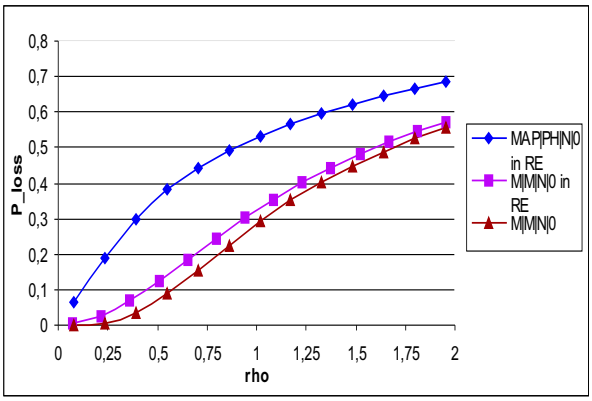


Figure 2. Loss probability in original system and exponential queueing systems

Basing on MAP_3 and MAP_4 we construct $BMAP_3$ and $BMAP_4$. The matrix $D_0^{(r)}$ is the same as one in the correspondent MAP and the matrices D_k are defined as $D_k^{(r)} = D^{(r)}q^{k-1}(1-q)/(1-q^6)$, $k = \overline{1,6}$, $r = \overline{1,2}$, where $q = 0.8$.

The first operation mode of original system in RE is described by $BMAP_3$. This $BMAP$ has $\lambda^{(3)} = 7.17$, $c_{var}^{(3)} = 2$, $c_{cor}^{(3)} = 0.2$. The PH_3 service is described by the parameters $\boldsymbol{\beta}^{(3)}$, $S^{(3)}$.

The second operation mode is described by $BMAP_4$ having $\lambda^{(4)} = 7.17$, $c_{var}^{(4)} = 2$, $c_{cor}^{(4)} = 0.3$. The PH_4 service is described by the parameters $\boldsymbol{\beta}^{(4)}$, $S^{(4)}$. Figure 3 corresponds to partial admission discipline. It shows the dependence of the loss probability on the value ρ in the original system - $BMAP/PH/N/0$ in RE, in the first type analog: $M^X/M/N/0$ in RE and the second type analog: $M^X/M/N/0$.

Figure 4 illustrates the analogous dependence in case of complete rejection discipline.

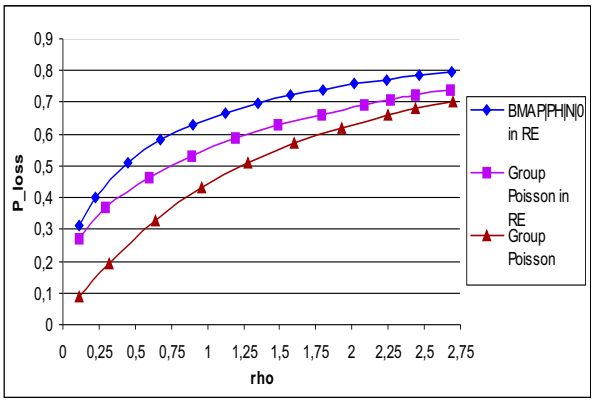


Figure 3. Loss probability in original system and exponential queueing systems for partial admission discipline

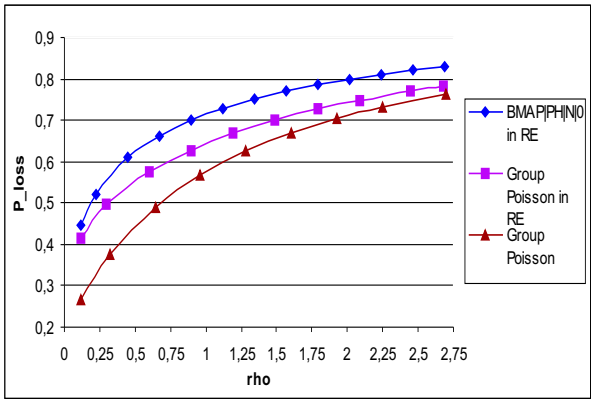


Figure 4. Loss probability in original system and exponential queueing systems for complete rejection discipline

ACKNOWLEDGMENTS

This work was supported by grant No. R01-2006-000-10668-0 from the Basic Research Program of Korea Science & Engineering Foundation.

REFERENCES

Lucantoni D. New results on the single server queue with a batch Markovian arrival process //Comm. Stat. - Stochastic Models, 1991. V. 7. P. 1-46.

Klimenok V. I. Characteristics calculation for multi-server queue with losses and bursty traffic // Automatic Control and Computer Sciences, 1999. V. 12. P. 393-415.

Klimenok V. I., Orlovsky D. S., Dudin A. N. Lack of invariant property of the Erlang loss model in case of MAP input // Queueing Systems, 2005. V. 49. P. 187-213.

Kim C. S., Klimenok V. I., Khramova V. V. BMAP/M/N/0 queueing model operating in random environment // 3rd International conference Information Systems and Technologies, Minsk, November 1-3, 2006. V. 1. P. 188-193.

MAP/(PH, PH)/c RETRIAL QUEUE WITH SELF-GENERATION OF PRIORITIES AND NON - PREEMPTIVE SERVICE

A.Krishnamoorthy and **S.Babu**
Department of Mathematics, Cochin University of
Science and Technology, Cochin-682 022, India.
E-mail: {ak,babu.s}@cusat.ac.in

ABSTRACT

Customers join the c server system according to a Markovian arrival process. If any of the servers is free, such a customer enters for service immediately. If all servers are busy the arriving customer enters an orbit of infinite capacity. Each customer in the orbit, tries independently of each other, to access the server at a constant rate θ. Customer in the orbit independently of others generate into priority with inter occurrence time exponentially distributed with parameter γ. A priority generated customer is immediately taken for service if any of the server is free. Else it waits in a waiting space (specially for priority generated customers) of capacity c, if this waiting space is not full at that instant, for priority generated customers. If this waiting space is full the present priority generated customer leaves the system for ever. The service discipline is non-preemptive priority. The service times of ordinary and priority generated customers follow PH-distribution. We provide a numerical procedure to compute the optimal number of servers to be employed to minimize the loss of customers. It is proved that the system is always stable. Several performance measures are evaluated.

KEYWORDS: Retrial queues, Self-generation of Priorities, Non-Preemptive service, Markovian Arrival process, PH-distribution, Matrix Analytic method, Server optimization.

Subject classification: Primary $60K25$; Secondary $68M20$, $90B22$.

1 INTRODUCTION

In the retrial queuing system, customers (or calls or objects) arriving into a busy system join a group of blocked customers called orbit and try for service after a random amount of time. Such phenomenon occurs in communication and computer networks, aircraft landing, machine repair and several other fields.

Quite a large number of probabilistic models possessing a variety of properties have been discussed in the literature on priority queues (see books by Gross and Harris [3], Jaiswal[4]. All these treat priority queues with exogenous priority rules which means that the decision of selecting the next unit for service may depend only on the knowledge of the priority class to which the unit belongs. However in many applications this discipline may not be an accurate modelling approach. This is especially the case in several real life situations such as aircraft landing, communication related problems. At the time of arrival a customer does not assume (i.e., not assigned) any priority; however those in the orbit generate priority, resulting in the need for urgent attention. For example, an aircraft in queue for landing may develop problems such as running out of fuel and so has to be given the next chance to land. We shall call such customers as priority generated customers (see Krishnamoorthy, Viswanath and Deepak[5], Gomez Corral, Krishnamoorthy and Viswanath[2]).

In this paper we discuss a multi-server queuing process in which customers in the orbit generate into priority. The capacity of the system is assumed to be infinite. The service discipline is non-preemptive and so a priority generated customer waits until one server becomes free, provided all servers are busy at the priority generation epoch. Thus we assume that a waiting space of capacity c (as many as the number of servers) is provided exclusively for the priority generated customers. Hence at any given time there can be at most $2c$ priority generated customers in the system. Any customer in the orbit generating into priority at an epoch when all servers are busy and c priority generated customers are already in the wait, will have to leave the system in search of urgent service elsewhere. Customers arrive to the system according to

Markovian Arrival Process (MAP). Service times of 'ordinary' (i.e., customers in the orbit and primary customers) and 'priority generated' customers follow distinct phase type (PH) distributions.

This paper is arranged as follows. In section 2 the problem is mathematically formulated and analyzed. In section 3 we prove that the system under study is always stable. We construct a dominating process to arrive at a truncation level. From there we proceed to obtain the long run system state distribution. These are done in section 4. In section 5 we provide a number of system performance measures of interest. Finally, in section 6 we investigate the optimal value of c numerically.

2 MATHEMATICAL MODELLING

Here we consider a service system with c servers, to which customers arrive according to a Markovian arrival process with representation (D_0, D_1). An arriving customer enters service immediately if at least one server is free; on the other hand it enters an orbit of infinite capacity if all servers are busy. Each customer in the orbit tries independently of each other to access the server at a constant rate θ (i.e., if there are k customers in the orbit, the rate of retrial is $k\theta$). Each Customer in the orbit independently of others generate into priority with inter occurrence time exponentially distributed with parameter γ. A priority generated customer is immediately taken for service if at least one of the servers is free. Else it waits in a waiting space (specially for priority generated customers) of capacity c if this waiting space is not full at that instant with priority generated customers. If this waiting space is full the priority generated customer will leave the system for ever. A customer in service (priority generated or otherwise) will be completely served before the priority generated customer is taken for service (non-preemptive service).

The service time of ordinary and priority generated customers follow PH-distribution with representation (α,T) and (β,S) respectively. Write $T_0 = -T\underline{e}$ and $S_0 = -S\underline{e}$ where $\underline{e}$ is a column vector of 1's of appropriate order.

We use the following definitions based on Kronecker sum $\oplus$ and Kronecker product $\otimes$.

Definition 1. *For a given square matrix A, define* $A^{\oplus m}$ *as the matrix*
$A^{\oplus m} = A \oplus A \oplus ..... \oplus A$, *m terms, for* $m \geq 1$ *and* $A^{\oplus 0} = 0$, *the scalar.*

Definition 2. *For a column vector B with n entries the matrix*
$B^{\oplus m} = B \otimes I_{n^{m-1}} + I_n \otimes B \otimes I_{n^{m-2}} + ... + I_{n^{m-1}} \otimes B$, *for* $m \geq 1$ *and* $B^{\oplus 0} = 1$, *the scalar.*

Let, $N_1(t)$ = number of customers in the orbit at time t, $N_2(t)$ = number of busy servers at time t, $N_3(t)$ = number of priority generated customers in service at time t, $N_4(t)$ = number of priority generated customers waiting for service at time t, $M(t)$ = phases of Markovian arrival process, $\mathcal{M}_1(t)$ = vector of phases of service process of ordinary customers and $\mathcal{M}_2(t)$ = vector of phases of service process of priority generated customers.

Write $X(t) = (N_1(t), N_2(t), N_3(t), N_4(t), M_(t), \mathcal{M}_1(t), \mathcal{M}_2(t))$; then $\{X(t) : t \geq 0\}$ forms a continuous time Markov chain with state space $S = \bigcup_{k=0}^{\infty} L(k)$, in which the k^{th}level $L(k) = \bigcup_{i=0}^{c} l_k(i)$, where

$$l_k(i) = \begin{cases} l'(k,0,0,0) \text{ , if } i = 0 \\ \bigcup_{j=0}^{i} l''(k,i,j,0) \text{ ,if } 1 \leq i < c \\ \bigcup_{j,j_1=0}^{c} l'''(k,c,j,j_1) \end{cases}$$

The element $l'(k,0,0,0)$ represents $\{(k,0,0,0,\nu) : 1 \leq \nu \leq m_1\}$, which means all servers are idle. $l''(k,i,j,0)$ represents $\{(k,i,j,0,\nu,\mu_1,..,\mu_{i-j},\eta_1,..,\eta_j) : 1 \leq \nu \leq m_1, 1 \leq \mu_1,..,\mu_{i-j} \leq m_2, 1 \leq \eta_1,..,\eta_j \leq m_3\}$, here we consider the service phases only for busy servers. Finally $l'''(k,c,j,j_1)$ represents $\{(k,c,j,j_1,\nu,\mu_1,..,\mu_{c-j},\eta_1,..,\eta_j) : 1 \leq \nu \leq m_1, 1 \leq \mu_1,..,\mu_{c-j} \leq m_2, 1 \leq \eta_1,..,\eta_j \leq m_3\}$.

By partitioning the state space into levels based on the number of customers in the orbit, the generator of the above Markov chain has the block partitioned form

$$Q = \begin{bmatrix} B_0 & A_0 & & & \\ C_1 & B_1 & A_0 & & \\ & C_2 & B_2 & A_0 & \\ & & \ddots & \ddots & \ddots \end{bmatrix}.$$

The description of A_0, B_k and C_k are as follows:

$$A_0 = \begin{bmatrix} 0_{\nu_1 \times \nu_1} & 0_{\nu_1 \times \nu_2} \\ 0_{\nu_2 \times \nu_1} & A \end{bmatrix} \text{ in which}$$

$$\nu_1 = m_1 \sum_{i=0}^{c-1} \sum_{j=0}^{i} m_2^{i-j} m_3^j,$$

$$\nu_2 = (c+1)\, m_1 \sum_{j=0}^{c} m_2^{c-j} m_3^j \text{ and}$$

$$A = \begin{bmatrix} \varphi(0) & 0 & .. & 0 \\ 0 & \varphi(1) & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & \varphi(c) \end{bmatrix} \text{ where}$$

$\varphi(j) = I_{c+1} \otimes D_1 \otimes I_{m_2^{c-j} m_3^j}$, $j = 0, 1, ..., c$;

$$B_k = \begin{bmatrix} B_{1k0} & B_{0k0} & & & & \\ B_{2k1} & B_{1k1} & B_{0k1} & & & \\ & B_{2k2} & B_{1k2} & B_{0k2} & & \\ & & \ddots & \ddots & \ddots & \\ & & & B_{2kc-1} & B_{1kc-1} & B_{0kc-1} \\ & & & & B_{2kc} & B_{1kc} \end{bmatrix},$$

$k = 0, 1, 2, \ldots.$;
for $i = 0, 1, ..., (c-2)$

$$B_{0ki} = \begin{bmatrix} \varphi_i(0) & 0 & 0 & .. & 0 \\ 0 & \varphi_i(1) & 0 & .. & 0 \\ .. & .. & .. & .. & .. \\ 0 & 0 & .. & \varphi_i(i) & 0 \end{bmatrix},$$

where $\varphi_i(j) = D_1 \otimes \alpha \otimes I_{m_2^{i-j} m_3^j}$,
$j = 0, 1, ..., i$,

$$B_{0kc-1} = \begin{bmatrix} V_0 & 0 & .. & 0 & 0 \\ 0 & V_1 & .. & 0 & 0 \\ .. & .. & .. & .. & .. \\ 0 & 0 & .. & V_{c-1} & 0 \end{bmatrix}, \text{ in which}$$

$$V_j = \begin{bmatrix} D_1 \otimes \alpha \otimes I_{m_2^{c-1-j} m_3^j} & 0 & .. & 0 \end{bmatrix},$$

$j = 0, 1, ..., (c-1)$;
for $i = 0, 1, ..., (c-1)$,

$$B_{1ki} = \begin{bmatrix} \psi_k(i,0) & & & \\ & \psi_k(i-1,1) & & \\ & & \ddots & \\ & & & \psi_k(0,i) \end{bmatrix},$$

with $\psi_k(i,j) = D_0 \oplus T^{\oplus i} \oplus S^{\oplus j} - k\theta I_{m_1 m_2^{i-j} m_3^j}$,
j=0,1,...,i;

$$B_{1kc} = \begin{bmatrix} V'_{k0} & V''_0 & & & & \\ & V'_{k1} & V''_1 & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & V'_{kc} & V''_c \end{bmatrix},$$

$$V'_{kj} = \begin{bmatrix} \zeta_k(j) & & & & \\ \zeta'(j) & \zeta_k(j) & & & \\ & \ddots & & \ddots & \\ & & & \zeta'(j) & \zeta_k(j) \end{bmatrix},$$

$\zeta_k(j) = D_0 \oplus T^{\oplus(c-j)} \oplus S^{\oplus j} - k\gamma I_{m_1 m_2^{c-j} m_3^j}$,
$\zeta'(j) = I_{m_1 m_2^{c-j}} \otimes S_0^{\oplus j} \otimes \beta,\ j = 0, 1, ..., c$;

$$V''_j = \begin{bmatrix} 0 & 0 & ... & 0 & 0 \\ \delta_1(j) & 0 & .. & 0 & 0 \\ .. & .. & .. & .. & .. \\ 0 & 0 & .. & \delta_1(j) & 0 \end{bmatrix},$$

$\delta_1(j) = I_{m_1} \otimes T_0^{(c-j)} \otimes I_{m_3^j} \otimes \beta,\ j = 0, 1, ..., c$;
for $i = 1, 2, \ldots, c-1$

$$B_{2ki} = \begin{bmatrix} \delta_2(i,1) & 0 & 0 & .. \\ \delta_3(i,1) & \delta_2(i,2) & 0 & .. \\ 0 & \delta_3(i,2) & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & \delta_2(i,i) \\ 0 & 0 & .. & \delta_3(i,i) \end{bmatrix},$$

$\delta_2(i,j) = I_{m_1} \otimes T_0^{\oplus(i-j+1)} \otimes I_{m_3^{j-1}}$
$\delta_3(i,j) = I_{m_1 m_2^{i-j}} \otimes S_0^{\oplus j},\ j = 1, 2, ..., i$;

$$B_{2kc} = \begin{bmatrix} V'''_0 \\ V'''_1 \\ \vdots \\ V'''_c \end{bmatrix},$$

$$V'''_0 = \begin{bmatrix} \xi_1(0) & 0 & .. & 0 \\ 0 & 0 & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & 0 \end{bmatrix},$$

$$V'''_1 = \begin{bmatrix} \xi_2(1) & \xi_1(1) & 0 & .. & 0 \\ 0 & 0 & 0 & .. & 0 \\ .. & .. & .. & .. & .. \\ 0 & 0 & 0 & .. & 0 \end{bmatrix},$$

$$V'''_2 = \begin{bmatrix} 0 & \xi_2(2) & \xi_1(2) & 0 & .. & 0 \\ 0 & 0 & 0 & 0 & .. & 0 \\ .. & .. & .. & .. & .. & .. \\ 0 & 0 & 0 & 0 & .. & 0 \end{bmatrix},$$

$$\ldots\ldots,\ V'''_c = \begin{bmatrix} 0 & .. & 0 & \xi_2(c) \\ 0 & .. & 0 & 0 \\ .. & .. & .. & .. \\ 0 & .. & 0 & 0 \end{bmatrix},$$

V'''_i are block matrices of order $(c+1) \times c$,
$\xi_1(i) = I_{m_1} \otimes T_0^{\oplus(c-i)} \otimes I_{m_3^i}$ and
$\xi_2(i) = I_{m_1 m_2^{c-i}} \otimes S_0^{\oplus i},\ i = 0, 1, .., c$;

$$C_k = \begin{bmatrix} 0 & W_{k0} & 0 & .. & 0 & 0 \\ 0 & 0 & W_{k1} & .. & 0 & 0 \\ .. & .. & .. & .. & .. & .. \\ 0 & 0 & 0 & .. & W_{kc-2} & 0 \\ 0 & 0 & 0 & .. & 0 & W_{kc-1} \\ 0 & 0 & 0 & .. & 0 & W_{kc} \end{bmatrix},$$

for $k = 0, 1, 2, ...$;
$W_{k0} = \begin{bmatrix} I_{m_1} \otimes (k\theta\alpha) & 0_{m_1 \times m_1 m_3} \end{bmatrix}$,

$$W_{k1} = \begin{bmatrix} I_{m_1 m_2} \otimes (k\theta\alpha) & 0 & 0 \\ 0 & I_{m_1 m_2} \otimes (k\theta\alpha) & 0 \end{bmatrix},$$

where 0 blocks in the third column are of order $m_1 m_2 \times m_1 m_3^2$ and $m_1 m_3 \times m_1 m_3^2$ respectively

$$W_{kc-2} = \begin{bmatrix} \omega_k(2) & 0 & 0 & .. & 0 & 0 \\ 0 & \omega_k(3) & 0 & .. & 0 & 0 \\ .. & .. & .. & .. & .. & .. \\ 0 & 0 & 0 & .. & \omega_k(c) & 0 \end{bmatrix},$$

$\omega_k(j) = I_{m_1 m_2^{c-j} m_3^{j-2}} \otimes (k\theta\alpha),\ j = 2, 3, ..c$

$$W_{kc-1} = \begin{bmatrix} w'_{k0} & 0 & .. & 0 & 0 \\ 0 & w'_{k1} & .. & 0 & 0 \\ .. & .. & .. & .. & .. \\ 0 & 0 & .. & w'_{kc-1} & 0 \end{bmatrix},$$

$$w'_{ki} = \begin{bmatrix} I_{m_1 m_2^{c-i} m_3^{i-1}} \otimes (k\theta\alpha)\ 0 \ldots 0 \end{bmatrix}_{(c+1)\times 1}$$

the 0's in w'_{ki} are zero matrices of order $m_1 m_2^{c-i} m_3^{i-1} \times m_1 m_2^{c-i+1} m_3^{i-1}$

$$W_{kc} = \begin{bmatrix} w''_0 & 0 & .. & 0 \\ 0 & w''_1 & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & w''_c \end{bmatrix}, \text{ where}$$

$$w''_{ki} = \begin{bmatrix} 0 & \tau_k(i) & 0 & 0 & .. & 0 \\ 0 & 0 & \tau_k(i) & 0 & .. & 0 \\ .. & .. & .. & .. & .. & .. \\ 0 & 0 & 0 & 0 & .. & \tau_k(i) \\ 0 & 0 & 0 & 0 & .. & \tau_k(i) \end{bmatrix}.$$

$\tau_k(i) = k\gamma I_{m_1 m_2^{c-i+1} m_3^{i-1}},\ i = 0, 1, ..., c.$

3 SYSTEM STABILITY

Theorem 1. *With $\gamma > 0$, the system under discussion is always stable.*

Proof. Consider the Lyapunov test function defined by $\varphi(s) = k$, where 's' is a state in level k. Then the mean drift y_s is given by

$$y_s = \sum_{p \neq s} [\phi(p) - \phi(s)] q_{sp}$$
$$= \sum_{s'} [\phi(s') - \phi(s)]\, q_{ss'} + \sum_{s''} [\phi(s'') - \phi(s)]\, q_{ss''}$$
$$+ \sum_{s'''} [\phi(s''') - \phi(s)]\, q_{ss'''}$$

where s', s'' and s''' vary over the states belonging to levels $k-1$, k and $k+1$ respectively. Then $\varphi(s) = k,\ \varphi(s') = k-1,\ \varphi(s'') = k$ *and* $\varphi(s''') = k+1$

$$y_s = -\sum_{s'} q_{ss'} + \sum_{s'''} q_{ss'''}$$
$$= \begin{cases} -k\theta + \sum_{s'''} q_{ss'''}, & \text{at least one server is free} \\ -k\gamma + \sum_{s'''} q_{ss'''}, & \text{all servers are busy} \end{cases}$$

Since the number of phases is finite, $\sum_{s'''} q_{ss'''}$ is bounded by some fixed constant for any s in level $k \geq 1$. Hence we can find a positive real number K such that $\sum_{s'''} q_{ss'''} < K$ for all s in level $k \geq 1$. Thus, for any $\varepsilon > 0$,we can find K_1 large enough that $y_s < -\varepsilon$ for any s belonging to level $i \geq K_1$. Hence the theorem follows from Tweedi [7]. □

4 STEADY STATE DISTRIBUTION

Here the process $\{X(t) : t \geq 0\}$ is a positive recurrent LDQBD; let $x = (x_0, x_1, ...)$ be its steady state distribution. x_i satisfies the relationship $x_{k+1} = x_k R_k,\ k \geq 0$, which gives $x_{k+1} = x_0 \prod_{l=0}^{k} R_l$, where the family of matrices $\{R_k : k \geq 0\}$ are the minimal nonnegative solution of the system of equations $A_0 + R_k B_{k+1} + R_k R_{k+1} C_{k+2} = 0$ for $k \geq 0$ and x_0 is the solution of $x_0(B_0 + R_0 C_1) = 0$ subject to

$$x_0 \left(I + \sum_{k=1}^{\infty} \prod_{l=0}^{k-1} R_l \right) \underline{e} = 1. \qquad (1)$$

Before proceeding to the numerical computations we construct a dominating process. Here the process under discussion, $\{X(t), t \geq 0\}$, satisfies the condition that for all $k \geq 1$ and for all i, there exists j such that $(\mathrm{C}_k)_{ij} > 0$ and hence a dominating process $\bar{X}(t)$ (see Bright and Taylor [1]) on the same state space as that of X(t) and with generator

$$\bar{Q} = \begin{bmatrix} B_0 & A_0 & & & \\ 0 & \bar{B}_1 & \bar{A}_0 & & \\ & \bar{C}_2 & \bar{B}_2 & \bar{A}_0 & \\ & & \bar{C}_3 & \bar{B}_3 & \bar{A}_0 \\ & & & \ddots & \ddots & \ddots \end{bmatrix},$$

where, $(\bar{A}_0)_{i,j} = \frac{1}{N}\left((A_0\underline{e})_{\max}\right)$,
$(\bar{C}_k)_{i,j} = \frac{1}{N}\left((C_{k-1}\underline{e})_{\min}\right), k \geq 2$,
$(\bar{B}_k)_{i,j} = (B_k)_{i,j}, i \neq j$ and $k \geq 1$, in which $N = \sum_{i=1}^{c=1} \sum_{j=0}^{i} m_2^{i-j} m_3^j + (c+1) \sum_{j=0}^{c} m_2^{c-j} m_3^j$, $(A_0\underline{e})_{max}$ is the maximum element of the column vector $A_0\underline{e}$ and $(C_{k-1}\underline{e})_{\min}$ is the minimum element of the column vector $C_{k-1}\underline{e}$.

We fix a truncation level K^* from the above method and employ Neuts-Rao [6] procedure in numerical computations. Thus $x_k(K^*),\ 0 \leq k \leq K^*$, is given by $x_k(K^*) = x_0(K^*) \prod_{l=0}^{k-1} R_l$ where $x_0(K^*)$ satisfies $x_0(B_0 + R_0 C_1) = 0$. The components of x above the level K^* are given by $x_{K^*+i} = x_{K^*} \prod_{j=1}^{i} R_{K^*+j}$ and equation (1) becomes $x\underline{e} = x_{K^*+1}(I - R_{K^*})^{-1}\underline{e} + x_0(K^*)\left(I + \sum_{k=1}^{K^*} \prod_{l=0}^{k-1} R_l)\right)\underline{e} = 1$.

5 SYSTEM PERFORMANCE MEASURES

We partition each x_k in the steady state probability vector $x = (x_0, x_1, x_2, ...)$ as $x_k = (y_{k0}, y_{k1}, ..., y_{kc})$ in which $y_{ki} = (y_{ki}(0,0), y_{ki}(1,0), ..., y_{ki}(i,0))$, for $k < c$ and $y_{kc} = (y_{kc}(j_1, j_2) : 0 \leq i, j \leq c)$. Here $y_{ki}(j_1, j_2)$ represents the row vector corresponding to $N_2(t) = i,\ N_3(t) = j_1$ and $N_4(t) = j_2$, respectively.

We concentrate on the following system performance measures.

- Average number E_1 of customers in the orbit $= \sum_{k=0}^{\infty} k x_i \underline{e}$.
- Average number E_2 of successful retrials $= \sum_{k=1}^{\infty} k\theta \left(\sum_{i=0}^{c-1} \sum_{j=0}^{i} y_{ki}(j,0) \right) \underline{e}$.
- Average number E_3 of priority generated customers in the system $= \sum_{k=0}^{\infty} \left(\sum_{i=1}^{c-1} \sum_{j=1}^{i} j y_{ki}(j,0) + \sum_{i=0}^{c} \sum_{j=1}^{c} (i+j)\, y_{kc}(i,j) \right) \underline{e}$.
- Average number E_4 of priority generated cus-

tomers waiting

$$= \sum_{k=0}^{\infty} \left(\sum_{i=0}^{c} \sum_{j=1}^{c} j y_{kc}(i,j) \right) \underline{e}.$$

• Average number E_5 of priority generated customers lost per unit time

$$= \sum_{k=1}^{\infty} k\gamma \left(\sum_{i=0}^{c} y_{kc}(i,c) \right) \underline{e}.$$

• Average number E_6 of idle servers $= \sum_{k=0}^{\infty} \left(\sum_{i=1}^{c-1} (c-i) \sum_{j=0}^{i} y_{ki}(j,0) \right) \underline{e}$. In order to find the optimal value of c numerically we construct a cost function as follows. Let

C_1 = Holding cost per unit of a priority generated customer in service.

C_2 = Holding cost per unit of a priority generated customer waiting for service.

C_3 = Cost per unit of a priority generated customer lost to the system.

C_4 = Cost per unit of an idle server.

•The expected total cost

$\mathrm{TC} = (E_3 - E_4)C_1 + E_4C_2 + E_5C_3 + E_6C_4$.

6 NUMERICAL EXAMPLE

Take $D_0 = \begin{bmatrix} -11.0 & 0.50 \\ 0.25 & -0.75 \end{bmatrix}$ and

$D_1 = \begin{bmatrix} 10.0 & 0.50 \\ 0.25 & 0.25 \end{bmatrix}$.

Here fundamental arrival rate = 3.83333 and correlation = 0.12356

Let $S = \begin{bmatrix} -8.0 & 4.0 \\ 4.0 & -8.0 \end{bmatrix}$, $S_0 = \begin{bmatrix} 4.0 \\ 4.0 \end{bmatrix}$

$T = \begin{bmatrix} -15.0 & 3.0 \\ 3.0 & -15.0 \end{bmatrix}$ and $T_0 = \begin{bmatrix} 12.0 \\ 12.0 \end{bmatrix}$

with $\alpha = \begin{bmatrix} 0.3 & 0.7 \end{bmatrix}$ and $\beta = \begin{bmatrix} 0.4 & 0.6 \end{bmatrix}$.

Further $\gamma = 10$, $\vartheta = 5$, $C_1 = 10$, $C_2 = 10$, $C_3 = 200$, $C_4 = 25$. We then have

Table 1. Number of servers versus expected total cost.

c	1	2	3	4	5
E_1	0.44208	0.13240	0.03042	0.00560	0.00089
E_2	0.48705	0.11774	0.10850	0.02366	0.00401
E_3	0.22213	0.14520	0.02080	0.00217	0.00024
E_4	0.16043	0.06997	0.00483	0.00027	0.00002
E_5	1.70035	0.18007	0.00398	0.00003	0.00000
E_6	0.64099	1.59876	2.66702	3.67911	4.68039
TC	358.3161	77.4350	67.6795	92.0055	117.0122

From the above table we conclude that the optimal number of servers corresponding to the given input parameters is 3.

Acknowledgment: A. Krishnamoorthy acknowledges research support from NBHM (Department of Atomic Energy, Government of India) Grant No: $48/5/2003/R$ & $DII/3169$ and S.Babu thanks the University Grants Commission (Government of India) for supporting his research under the Faculty Improvement Programme.

References

[1] Bright,L. and Taylor,P.G.: Equilibrium distribution for level-dependent quasi-birth-and-death processes. Comm. Stat. Stochastic Models, 11:497- 525, 1995.

[2] Gomez Corral,A., Krishnamoorthy,A. and Viswanath.C.Narayanan: Impact of Self generation of Priorities on Multiserver queues with Finite Capacity. Stochastic Models, 21:427-447, 2005.

[3] Gross,D. and Harris,C.M.: Fundamentals of Queueing Theory, 3rd Ed., JohnWiley and Sons,Inc., New York, 1988.

[4] Jaiswal,N.K.: Priority Queues. Academic Press, New York,1968.

[5] Krishnamoorthy, A., Viswanath. C. Narayanan . and Deepak,T.G.: On a queuing system with self-generation of priorities. Neural Parallel and Scientific Computing, 13, 2005.

[6] Neuts,F.M. & Rao,B.M.: Numerical investigations of a multiserver retrial model. Queuing systems, 7:169-190, 1990.

[7] Tweedi,R.L.: Sufficient condition for regularity, recurrence and ergodicity of Markov processes. Proc. Camb. Phil. Soc.,78,Part I,1975.

MODELING FINITE-SOURCE RETRIAL QUEUEING SYSTEMS WITH UNRELIABLE HETEROGENEOUS SERVERS AND DIFFERENT SERVICE POLICIES USING MOSEL

Patrick Wüchner, Hermann de Meer,
Faculty of Informatics and Mathematics,
University of Passau,
Innstraße 43,
94032 Passau, Germany.
patrick.wuechner@uni-passau.de

Gunter Bolch,
Institute of Computer Science,
University of Erlangen,
Martensstraße 3,
91508 Erlangen, Germany.
bolch@informatik.uni-erlangen.de

János Roszik, János Sztrik,
Faculty of Informatics,
University of Debrecen,
Egyetem tér 1. Po.Box 12,
4010 Debrecen, Hungary.
jsztrik@inf.unideb.hu

KEYWORDS
Performance and Reliability Evaluation, Retrial Queuing System Model, Unreliable Heterogeneous Servers.

ABSTRACT

This paper deals with the performance analysis of multiple server retrial queueing systems with a finite number of homogeneous sources of calls, where the heterogeneous servers are subject to random breakdowns and repairs. The requests are serviced according to Random Selection and Fastest Free Server disciplines.

The novelty of this investigation is the introduction of different service rates and different service policies together with the unreliability of the servers, which has essential influence on the performance of the system, and thus, it plays an important role in practical modeling of computer and communication systems. All random variables involved in the model construction are assumed to be exponentially distributed and independent of each other.

The main steady-state performability measures are derived, and several numerical calculations are carried out by the help of the MOSEL tool (Modeling, Specification and Evaluation Language) under different service disciplines. The numerical results are graphically displayed to illustrate the effect of failure rates on the mean response time and on the overall system utilization.

I. INTRODUCTION

Retrial queueing systems (also known as queueing systems with repeated attempts or with returning customers) are characterized by the following feature: a primary request finding all servers busy on arrival does not wait in a queue, but leaves the service area and after some random time repeats its demand. For the most important results on this type of queues see, for example (Falin and Templeton 1997), (Artalejo 1998), (Artalejo 1999), and (Falin 1999). This feature plays a special role in many computer and communication systems having a significant negative impact on the performance characteristics of the system. For some examples in the field of computer systems and communication networks see (Li and Yang 1995), (Janssens 1997), and (Tran-Gia and Mandjes 1997).

In general, the components of practical computer systems are subject to random breakdowns. This has a heavy influence on the performance measures just as the retrial phenomenon. Thus, if we model computer systems containing unreliable components it is important to take it into account in the model construction. Of course, the breakdown of the servers has the most significant negative impact on the performance of the most frequently used client-server architecture. For modeling this type of systems, both infinite and finite-source retrial queues with server breakdowns were applied—see, for example, (Artalejo 1994), (Aissani and Artalejo 1998), (Wang et al. 2001), and (Almasi et al. 2005). Queueing systems with heterogeneous servers are still an interesting topic. For recent results confer, for example, (Rykov 2001), (Nobel and Tijms 2000). However, for retrial queueing systems with heterogeneous servers we have found only (Pourbabai 1987). To the best knowledge of the authors there is no paper on finite-source retrial queues with heterogeneous servers, not even in purely reliable case.

In this paper, we analyze the finite-source retrial queue with unreliable heterogeneous (asymmetric) servers, that is, the servers have different parameters in service, failure, and repair rates. In the present study, the most important heterogeneous characteristic is the service rate, since we compare two service policies, namely Random Service (RS) and Fastest Free Server (FFS). In the case of RS discipline, the requests are assigned to the idle servers randomly, and in the FFS case, the requests are assigned to the fastest available free server.

The purpose of this paper is to generalize the models of (Falin 1999) and (Almasi et al. 2005). The novelty of this investigation is the introduction of different service rates and different service policies together with the unreliability of the servers.

The main steady-state performability measures are derived, and several numerical calculations are carried out by the help of the MOSEL tool (Begain et al. 2001) under different service disciplines. The numerical results are graphically displayed to illustrate the effect of failure rates on the mean response time and on the overall system utilization.

The organization of the paper is as follows. In the next section, we give the mathematical model description and derive the performance measures. Then, we use the efficient software tool MOSEL to formulate the model

Print 0-9553018-4-X, CD 0-9553018-3-1

and to obtain the performance measures. In Section 3, we present some numerical examples for the models under different service disciplines. The results are graphically displayed using the IGL (Intermediate Graphical Language) interpreter which belongs to MOSEL. By the help of these figures we illustrate the effect of failure rates on the mean response time and on the overall system utilization. Section 4 is devoted to some conclusions.

II. THE $M/\vec{M}/c//K$ RETRIAL QUEUEING MODEL WITH UNRELIABLE SERVERS AND DIFFERENT SERVICE POLICIES

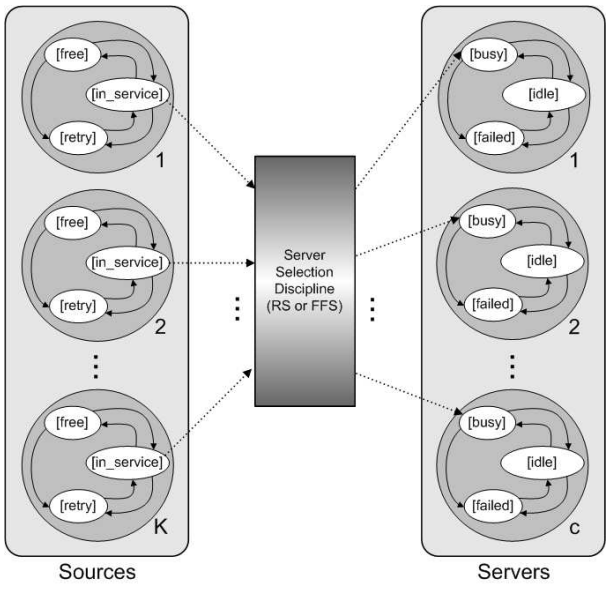


Fig. 1. Finite-source retrial queueing system

Consider a finite-source retrial queueing system with c servers, where the primary calls are generated by K, $c \leq K < \infty$, sources (see Fig. 1). Each server can be in operational (`up`) or non-operational (`failed`) states, and if it is up it can be `idle` or `busy`. Each source can be in three states: generating a primary call (`free`), sending repeated calls (`retry`), and under service (`in_service`) by one of the servers. If a source is free at time t, it can generate a primary call during the interval $(t, t+dt)$ with probability $\lambda dt + o(dt)$. If one of the servers is up and idle at the moment of the arrival of the call then the service of the call starts. In the case of Random Service (RS) discipline, the requests are assigned to the available free servers randomly, but in the Fastest Free Server (FFS) case the availability and idleness of the servers are always examined according to the highest service rates. The service is finished during the interval $(t, t+dt)$ with probability $\mu_k dt + o(dt)$ if the kth server is available.

Server k can fail during the interval $(t, t+dt)$ with probability $\delta_k dt + o(dt)$ if it is idle, and with probability $\gamma_k dt + o(dt)$ if it is busy. If the server fails in busy state, the interrupted request returns to the orbit and, thus, the respective source will retry to get service. If $\delta_k = 0$ and $\gamma_k > 0$, or $\delta_k = \gamma_k > 0$, *active* or *independent breakdowns* can be discussed, respectively. The repairman follows FIFO discipline to fix up the server breakdowns, the repair time of the kth server is exponentially distributed with a finite mean $1/\tau_k$. If all the servers are failed, two different cases can be treated: In the *blocked sources* case, all the operations are stopped except from the repair of the servers. On the other hand, in the *unblocked (intelligent) sources* case, only service is interrupted but all the other operations are continued. If all the servers are busy or failed at the moment of the arrival of a call, the source starts generating a Poisson flow of retrial calls with rate ν until it finds an available free server. After service, the source can generate a new primary call, and the server becomes idle so it can serve a new call. All the times involved in the model construction are assumed to be mutually independent of each other.

The state of the system at time t can be described by the process $X(t) = (\alpha_1(t), \ldots, \alpha_c(t); N(t))$, where $N(t) \leq K$ is the number of retrying sources at time t, and $\alpha_k(t), k = 1, \ldots, c$, denotes the state of the kth server at time t. If there is a customer under service at the kth server, we define $\alpha_k(t) = 1$, if it is operational and idle then $\alpha_k(t) = 0$, otherwise the server is failed and $\alpha_k(t) = -1$.

Because of the exponentiality of the involved random variables and the finite number of sources, this process is a Markov chain with a finite state space. Since the state space of the process $(X(t), t \geq 0)$ is finite, the process is ergodic for all reasonable values of the rates involved in the model construction. From now on, we assume that the system is in the steady-state. Because of the fact that the state space of the describing Markov chain is very large, it is difficult to calculate the system measures in the traditional way of solving the system of steady-state global balance equations. To simplify this procedure we use the software tool MOSEL.

Let us define the stationary probabilities by:

$$P(s_1, \ldots, s_c, j) = \lim_{t \to \infty} P\{\alpha_1(t) = s_1, \ldots, \alpha_c(t) = s_c, N(t) = j\},$$

$$s_1, \ldots, s_c = -1, 0, 1, \;\; j = 0, \ldots, K^*,$$

where K^* is the number of sources in service and given by:

$$K^* = K - \sum_{s_k, s_k = 1} s_k.$$

Furthermore, let us denote by $N(\infty)$ the number of retrying sources, $C(\infty)$ the number of busy servers, $A(\infty)$ the number of available servers at steady-state, and denote by $p_{kj} = P\{C(\infty) = k, N(\infty) = j\}$ the joint steady-state distribution of the number of busy servers and the number of retrying sources.

Once we have obtained the above defined probabilities, the main steady-state system performance measures can be derived as follows:

- *Mean number of retrying sources (calls in orbit):*

$$N = E[N(\infty)] = \sum_{k=0}^{c}\sum_{j=1}^{K} jp_{kj} = \sum_{s_1,\ldots,s_c}\sum_{j=1}^{K^*} jP(s_1,\ldots,s_c,j).$$

- *Utilization of the kth server:*

$$U_k = \sum_{s_1,\ldots,s_c,s_k=1}\sum_{j=0}^{K^*} P(s_1,\ldots,s_c,j), \quad k=1,\ldots,c.$$

- *Mean number of busy servers:*

$$C = E[C(\infty)] = \sum_{k=1}^{c} U_k.$$

- *Mean number of calls staying in the orbit or in service:*

$$M = E[N(\infty) + C(\infty)] = N + C.$$

- *Utilization of the repairman:*

$$U_R = \sum_{\substack{s_1,\ldots,s_c \\ -1\in\{s_1,\ldots,s_c\}}}\sum_{j=0}^{K^*} P(s_1,\ldots,s_c,j).$$

- *Utilization of the sources:*

$$U_S = \begin{cases} \frac{E[K-C(\infty)-N(\infty);A(\infty)>0]}{K} & \text{(blocked case)}, \\ \frac{E[K-C(\infty)-N(\infty)]}{K} & \text{(unblocked case)}. \end{cases}$$

- *Overall utilization of the system:*

$$U_O = C + KU_S + U_R.$$

- *Mean rate of generation of primary calls:*

$$\overline{\lambda} = \begin{cases} \lambda E[K-C(\infty)-N(\infty);A(\infty)>0] & \text{(blocked case)}, \\ \lambda E[K-C(\infty)-N(\infty)] & \text{(unblocked case)}. \end{cases}$$

- *Mean waiting time:* $E[W] = N/\overline{\lambda}$.
- *Mean response time:* $E[T] = M/\overline{\lambda}$.

Validation of Results

The numerical results generated by the MOSEL tool in the reliable case (see model descriptions in Appendix) were validated by the Pascal program given in the book of Falin and Templeton (Falin and Templeton 1997). The service rates are the same for all servers in each case. In Table 1 we can see that the corresponding performance measures with RS and FFS disciplines are very close to the reliable case and to each other with very low failure and very high repair rates. The results are the same up to the 6th decimal digit.

The MOSEL models were tested in unreliable case, too. Since only the unreliable single server case was treated earlier, the results were validated by the $M/M/1//K$ retrial model with server breakdowns which was studied in (Almasi et al. 2005). The numerical calculations given in (Almasi et al. 2005) correspond to the examples of the paper at hand.

III. NUMERICAL EXAMPLES

In this section, we present some numerical results to illustrate graphically the differences between the service disciplines in the mean response time, in the utilization of the servers and in the overall system utilization. In the legends of the figures, the FFS policy is referred to as *ordered*, and the random case where the service rate of the servers is the average of the rates of the heterogeneous cases is referred to as *averaged random*. In all cases we consider independent breakdowns with the same failure and repair rate for all servers, respectively.

The input system parameters of the Figures 2, 3, and 4 are collected in Table 2 where the RS and FFS disciplines are compared.

Table 2: Input system parameters of Figs. 2, 3, and 4

	c	K	λ	$\mu_1,\ldots,\mu_c-\mu_{avg}$	ν	δ,γ	τ
Fig. 2, 3, 4	4	20	1	8, 5, 4, 1 – 4.5	4	x axis	0.2

The system parameters of Figures 5, 6, 7, 8, and 9 are collected in Table 3 where only the FFS discipline is treated.

Table 3: Input system parameters of Figs. 5, 6, 7, 8, and 9

	c	K	λ	μ_1,μ_2	ν	δ,γ	τ
Fig. 5, 6	2	5	0.2	1, 1	1.1	x axis	0.01
Fig. 7, 8, 9	2	5	0.2	1.5, 0.5	1.1	x axis	0.01

In Figures 5 and 7, the effects of the server failure rate on the mean response time are displayed with homogeneous and different servers, respectively. In Figures 6 and 9, we can see the effect of the server failure rate on the overall utilization. In Figure 8, the server utilization is shown in the case of different servers. In each figure, the reliable, the blocked, and unblocked (intelligent) cases are analyzed .

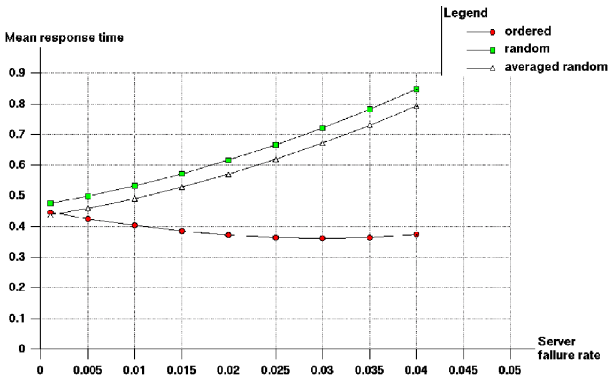


Fig. 2. Mean response time versus server failure rate

Discussion of Results

- In Figure 2, it is shown how the increase of the server failure rate affects the mean response time. The averaged random case has a little better response time than the not averaged random case comparable to the former figures. The surprising decrease in the mean response time in FFS case can be explained by the help of Figure 3.

Table 1: Validations in the reliable case

	Pascal (Falin and Templeton 1997)	RS	FFS
Number of servers:	4	4	4
Number of sources:	20	20	20
Request generation rate:	0.1	0.1	0.1
Service rate:	1	1	1
Retrial rate:	1.2	1.2	1.2
Server failure rate:	–	1e-25	1e-25
Server repair rate:	–	1e+25	1e+25
Mean waiting time:	0.1064954794	0.1064959317	0.1064959929
Mean number of busy servers:	1.8007480431	1.8007485102	1.8007485548
Mean number of call-repeating sources:	0.1917715262	0.1917717923	0.1917718470

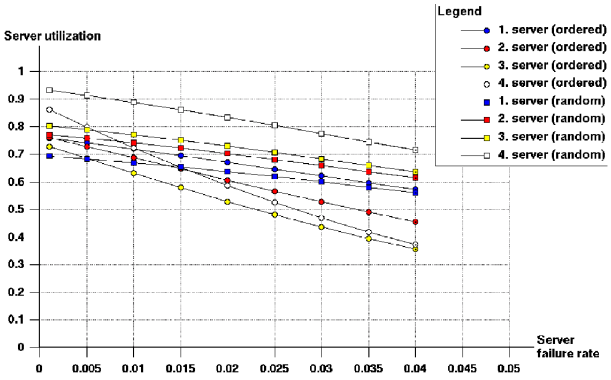


Fig. 3. Server utilization versus server failure rate

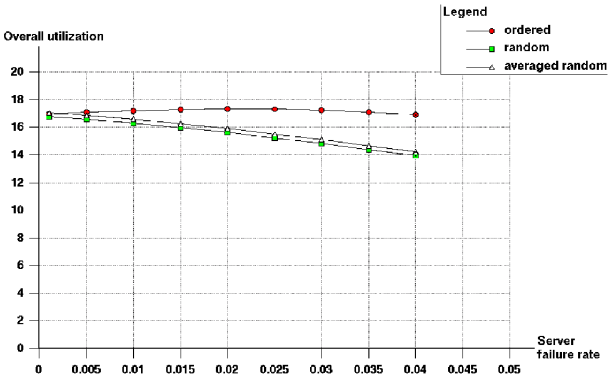


Fig. 4. Overall utilization versus server failure rate

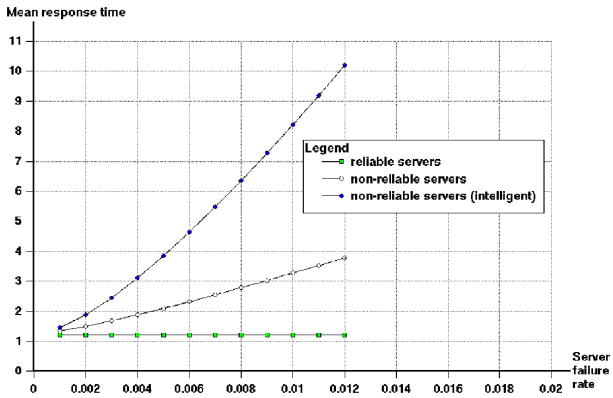


Fig. 5. $E[T]$ versus server failure rate

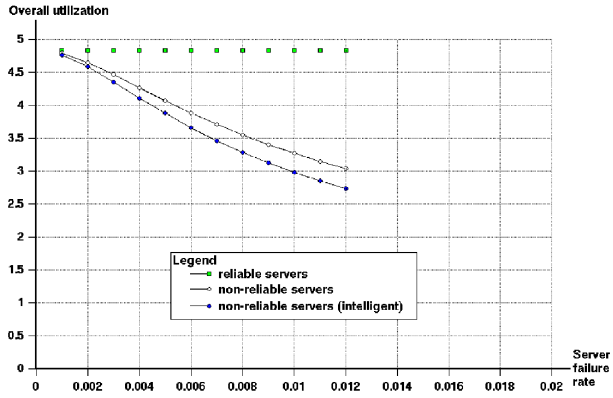


Fig. 6. U_O versus server failure rate

• In Figure 3, we can see the server utilization versus the server failure rate with the same parameter setup as in Figure 2. In RS case, the slowest server has the highest utilization and the fastest has the lowest, since it services the request much faster and the requests are assigned to the available and free servers with the same probability. In the beginning of the ordered (FFS) case, the slowest server has the highest utilization too, but as it fails more often, its service is interrupted more often and looses from its utilization much faster than the faster servers, since it gets requests to serve only if all the other servers busy or failed.

• In Figure 4, the overall utilization is displayed versus the server failure rate. Like the mean response time in Figure 2, the overall utilization is getting better for a while in the FFS case as the server failure rate increases.

• In Figures 5 and 7, it can be observed that the increase of the server failure rate can have a heavy impact on the mean response time, and as it increases the difference between the two unreliable model increases significantly.

• In Figures 6 and 9, it is shown that the overall utilization can be very low if the server failure rate increases and the repair rate is not high enough.

IV. CONCLUSIONS

In this paper, the performance of finite-source retrial queueing systems with homogeneous sources and unreliable heterogeneous (asymmetric) servers is studied. The novelty of the investigation is the introduction of different service rates and different service policies with the unreliability of the servers. The MOSEL software package was used to formulate the model and to calculate the steady-state system performance measures which were graphically displayed to show the differences between the service disciplines in the mean response time, in the utiliza-

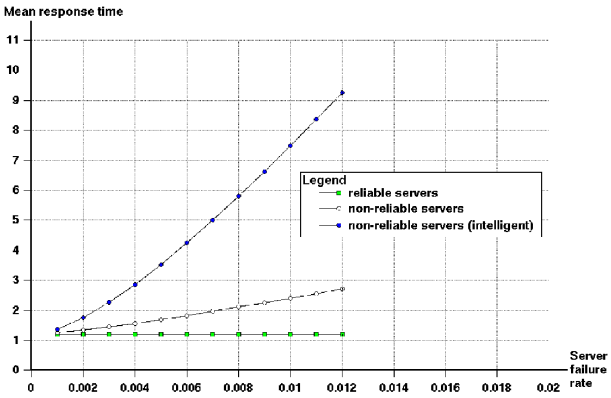


Fig. 7. $E[T]$ versus server failure rate

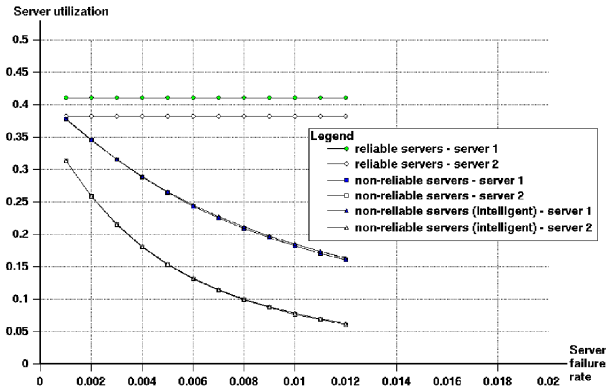


Fig. 8. Server utilization versus server failure rate

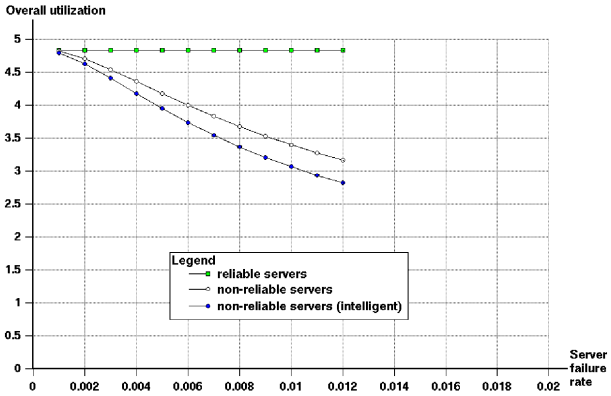


Fig. 9. U_O versus server failure rate

tion of the servers and in the overall system utilization versus server failure rate. It was demonstrated that the FFS discipline has more favorable system measures, as it was expected, and in the case of RS policy, it is more worthy to apply homogeneous servers with the average service rates of the heterogeneous case, at least with these setup of the parameters.

In future, we will extend our model to have a closer look on the effect of non-exponentially distributed service and retrial times. For this, we will employ the successor of MOSEL, called MOSEL-2 (Wuechner et al. 2006; Bolch et al. 2006), which is capable of specifying and evaluating non-Markovian discrete-event system models.

V. ACKNOWLEDGMENTS

This research is partially supported by the German-Hungarian Intergovernmental Scientific Cooperation, HAS-DFG, 436 UNG 113/180/0-1, by the Hungarian Scientific Research Fund, OTKA K60698/2006, by the Network of Excellence EuroFGI – IST 028022, and by EPSRC, GR/S69009/01.

REFERENCES

Aissani, A. and J. Artalejo (1998) "On the single server retrial queue subject to breakdowns. ", *Queueing Systems 30*, 309–321.

Almasi, B., J. Roszik, and J. Sztrik (2005) "Homogeneous finite-source retrial queues with server subject to breakdowns and repairs. ", *Mathematical and Computer Modelling 42*, 673–682.

Artalejo, J. (1994) "New results in retrial queueing systems with breakdown of the servers. ", *Statistica Neerlandica 48*, 23–36.

Artalejo, J. R. (1998) "Retrial queues with a finite number of sources. ", *J. Korean Math. Soc. 35*, 503–525.

Artalejo, J. R. (1999) "Accessible bibliography on retrial queues. ", *Mathematical and Computer Modelling 30*, 1–6.

Begain, K., G. Bolch, and H. Herold (2001). *Practical Performance Modeling – Application of the MOSEL Language*. Kluwer Academic Publishers.

Bolch, G., S. Greiner, H. de Meer, and K. Trivedi (2006). *Queueing Networks and Markov Chains* (2 ed.). New York: John Wiley & Sons.

Falin, G. and J. Templeton (1997). *Retrial Queues*. Chapman & Hall.

Falin, G. I. (1999) "A multiserver retrial queue with a finite number of sources of primary calls. ", *Mathematical and Computer Modelling 30*, 33–49.

Janssens, G. K. (1997) "The quasi-random input queueing system with repeated attempts as a model for collision-avoidance star local area network. ", *IEEE Transactions on Communications 45*, 360–364.

Li, H. and T. Yang (1995) "A single server retrial queue with server vacations and a finite number of input sources. ", *European Journal of Operational Research 85*, 149–160.

Nobel, R. D. and H. C. Tijms (2000) "Optimal control of a queueing system with heterogeneous servers and setup costs. ", *IEEE Trans. Autom. Control 45*, 780–794.

Pourbabai, B. (1987) "Markovian queueing systems with retrials and heterogeneous servers. ", *Computers and Mathematics with Applications 13*, 917–923.

Rykov, V. (2001) "Monotone control of queueing systems with heterogeneous servers. ", *Queueing Systems 37*, 391–403.

Tran-Gia, P. and M. Mandjes (1997) "Modeling of customer retrial phenomenon in cellular mobile networks. ", *IEEE Journal of Selected Areas in Communications 15*, 1406–1414.

Wang, J., J. Cao, and Q. Li (2001) "Reliability analysis of the retrial queue with server breakdowns and repairs. ", *Queueing Systems 38*, 363–380.

Wuechner, P., H. De Meer, J. Barner, and G. Bolch (2006) "A brief introduction to MOSEL-2. ", In R. German and A. Heindl (Eds.), *Proc. of MMB 2006 Conference*, pp. 473–476. GI/ITG/MMB, University of Erlangen: VDE Verlag.

AUTHOR BIOGRAPHIES

PATRICK WÜCHNER received his computer science Diploma in 2004 from the University of Erlangen-Nuernberg, Germany. Since then, he is research fellow and PhD student at the University of Passau, Germany. He is interested in research on mathematical performance and reliability modeling of computer systems, communication networks, and self-organizing systems.

HERMANN DE MEER is currently appointed as Full Professor at the University of Passau, Germany, and as Honorary Professor at University College London, UK. He is director of the Institute of IT-Security and Secu-

rity Law (ISL) at the University of Passau, Germany. He had been an Assistant Professor at Hamburg University, Germany, a Visiting Professor at Columbia University in New York City, USA, and a Reader at University College London, UK. His research interests include peer-to-peer systems, quality of service, Internet protocols, home networking, IT security, and mobile computing.

GUNTER BOLCH is Acad. Director the Department of Computer Science 4 (Operating Systems) at the University of Erlangen-Nuernberg, Germany. He studied telecommunication at the Technical Universities of Karlsruhe and Berlin and was Assistant Professor at the Department of Process Control at the University of Karlsruhe. After receiving his Ph.D. in 1973, he went to the University of Erlangen where from 1982 until his retirement in 2006 he was head of the Performance Modeling and Process Control research group at the Department of Computer Science 4. He was researching and lecturing in the area of performance modeling, process control, and operating systems.

JÁNOS ROSZIK received his MSc Degree in Computer Science in 2003 at the University of Debrecen, Hungary. He is currently a PhD student at the Department of Informatics Systems and Networks of the same university. His primary research interests are performance analysis of retrial queues and their application in modeling of telecommunication systems.

JÁNOS SZTRIK is a Full Professor at the Faculty of Informatics, University of Debrecen, Hungary. He obtained the Candidate of Mathematical Sciences Degree in Probability Theory and Mathematical Statistics in 1989 from the Kiev State University, Kiev, USSR, Habilitation from University of Debrecen in 1999, Doctor of the Hungarian Academy of Sciences, Budapest, 2002. His research interests are in the field of production systems modeling and analysis, queueing theory, reliability theory, and computer science.

APPENDIX

A. The MOSEL Model for the Random Service Policy

```
// ============================== Constant definitions ====
#define NT 20
#define NS 4
// ====================== Variables (input parameters) ====
VAR double prgen;
VAR double prretr;
<1..NS> VAR double prrun#;
<1..NS> VAR double cpubreak_idle#;
<1..NS> VAR double cpubreak_busy#;
<1..NS> VAR double cpurepair#;
// ================================== Node definitions ====
enum cpu_states {cpu_busy, cpu_idle, cpu_failed};
NODE busy_terminals[NT] = NT;
NODE retrying_terminals[NT] = 0;
NODE waiting_terminals[NS] = 0;
<1..NS> NODE cpu#[cpu_states] = cpu_idle;
        NODE freecpus[NS] = NS;
        NODE failedcpus[NS] = 0;
<1..NS> NODE sr#[NS] = 0;
// ======================================== Transitions ====
FROM cpu1[cpu_idle], busy_terminals, freecpus
        TO cpu1[cpu_busy], waiting_terminals
        W prgen*busy_terminals/freecpus;
FROM cpu2[cpu_idle], busy_terminals, freecpus
        TO cpu2[cpu_busy], waiting_terminals
        W prgen*busy_terminals/freecpus;
FROM cpu3[cpu_idle], busy_terminals, freecpus
        TO cpu3[cpu_busy], waiting_terminals
        W prgen*busy_terminals/freecpus;
FROM cpu4[cpu_idle], busy_terminals, freecpus
        TO cpu4[cpu_busy], waiting_terminals
        W prgen*busy_terminals/freecpus;
FROM busy_terminals TO retrying_terminals
        IF freecpus==0 W prgen*busy_terminals;
FROM cpu1[cpu_idle], retrying_terminals, freecpus
        TO cpu1[cpu_busy], waiting_terminals
        W prretr*retrying_terminals/freecpus;
FROM cpu2[cpu_idle], retrying_terminals, freecpus
        TO cpu2[cpu_busy], waiting_terminals
        W prretr*retrying_terminals/freecpus;
FROM cpu3[cpu_idle], retrying_terminals, freecpus
        TO cpu3[cpu_busy], waiting_terminals
        W prretr*retrying_terminals/freecpus;
FROM cpu4[cpu_idle], retrying_terminals, freecpus
        TO cpu4[cpu_busy], waiting_terminals
        W prretr*retrying_terminals/freecpus;
<1..NS><NS> FROM cpu<#1>[cpu_busy], waiting_terminals{
  TO cpu<#1>[cpu_idle], busy_terminals, freecpus W prrun<#1>;
  TO cpu<#1>[cpu_failed],retrying_terminals, failedcpus,sr<#2>(<#1>)
  W cpubreak_busy<#1>; }
<1..NS><NS> FROM cpu<#1>[cpu_idle], freecpus
  TO cpu<#1>[cpu_failed], failedcpus, sr<#2>(<#1>)
  W cpubreak_idle<#1>;
<1..NS> IF sr1==# FROM sr1(#), cpu#[cpu_failed], failedcpus
           TO cpu#[cpu_idle], freecpus W cpurepair#;
<2..NS> IF sr<#-1>==0 FROM sr#(sr#) TO sr<#-1>(sr#);
// ========================================== Results ====
<1..NS> RESULT>> if(cpu#==cpu_busy) cpuutil# += PROB;
<1..NS> RESULT>> if(cpu#==cpu_busy) busycpus += PROB;
<1..NS> RESULT>> if(cpu#==cpu_idle OR cpu#==cpu_busy) goodcpus+=PROB;
<1..NS> RESULT>> if(cpu#==cpu_failed) nfailedcpus += PROB;
RESULT if(busy_terminals>0) busyterm += PROB*busy_terminals);
RESULT>> termutil = busyterm / NT;
RESULT>> if(retrying_terminals>0) retravg += (PROB*retrying_terminals);
RESULT>> if(failedcpus>0) repairutil += PROB;
RESULT if(waiting_terminals>0) waitall += (PROB*waiting_terminals);
RESULT>> resptime = (retravg + waitall) / NT / (prgen * termutil);
RESULT>> overallutil = busycpus + termutil*NT + repairutil;
```

B. The MOSEL Model for the Fastest Free Server Policy

```
// ============================== Constant definitions ====
#define NT 20
#define NS 4
// ====================== Variables (input parameters) ====
VAR double prgen;
VAR double prretr;
<1..NS> VAR double prrun#;
<1..NS> VAR double cpubreak_idle#;
<1..NS> VAR double cpubreak_busy#;
<1..NS> VAR double cpurepair#;
// ================================== Node definitions ====
enum cpu_states {cpu_busy, cpu_idle, cpu_failed};
NODE busy_terminals[NT] = NT;
NODE retrying_terminals[NT] = 0;
NODE waiting_terminals[NS] = 0;
<1..NS> NODE cpu#[cpu_states] = cpu_idle;
        NODE freecpus[NS] = NS;
        NODE failedcpus[NS] = 0;
<1..NS> NODE sr#[NS] = 0;
// ======================================== Transitions ====
FROM cpu1[cpu_idle], busy_terminals, freecpus
        TO cpu1[cpu_busy], waiting_terminals
        W prgen*busy_terminals;
FROM cpu2[cpu_idle], busy_terminals, freecpus
        TO cpu2[cpu_busy], waiting_terminals
        IF cpu1==cpu_busy W prgen*busy_terminals;
FROM cpu3[cpu_idle], busy_terminals, freecpus
        TO cpu3[cpu_busy], waiting_terminals
        IF cpu1==cpu_busy AND cpu2==cpu_busy
        W prgen*busy_terminals;
FROM cpu4[cpu_idle], busy_terminals, freecpus
        TO cpu4[cpu_busy], waiting_terminals
        IF cpu1==cpu_busy AND cpu2==cpu_busy AND cpu3==cpu_busy
        W prgen*busy_terminals;
FROM busy_terminals TO retrying_terminals
        IF freecpus==0 W prgen*busy_terminals;
FROM cpu1[cpu_idle], retrying_terminals, freecpus
        TO cpu1[cpu_busy], waiting_terminals
        W prretr*retrying_terminals;
FROM cpu2[cpu_idle], retrying_terminals, freecpus
        TO cpu2[cpu_busy], waiting_terminals
        IF cpu1==cpu_busy W prretr*retrying_terminals;
FROM cpu3[cpu_idle], retrying_terminals, freecpus
        TO cpu3[cpu_busy], waiting_terminals
        IF cpu1==cpu_busy AND cpu2==cpu_busy
        W prretr*retrying_terminals;
FROM cpu4[cpu_idle], retrying_terminals, freecpus
        TO cpu4[cpu_busy], waiting_terminals
        IF cpu1==cpu_busy AND cpu2==cpu_busy AND cpu3==cpu_busy
        W prretr*retrying_terminals;
<1..NS><NS> FROM cpu<#1>[cpu_busy], waiting_terminals{
  TO cpu<#1>[cpu_idle], busy_terminals, freecpus W prrun<#1>;
  TO cpu<#1>[cpu_failed],retrying_terminals,failedcpus, sr<#2>(<#1>)
  W cpubreak_busy<#1>; }
<1..NS><NS> FROM cpu<#1>[cpu_idle], freecpus
  TO cpu<#1>[cpu_failed], failedcpus, sr<#2>(<#1>) W cpubreak_idle<#1>;
<1..NS> IF sr1==# FROM sr1(#), cpu#[cpu_failed], failedcpus
           TO cpu#[cpu_idle], freecpus W cpurepair#;
<2..NS> IF sr<#-1>==0 FROM sr#(sr#) TO sr<#-1>(sr#);
// ========================================== Results ====
<1..NS> RESULT>> if(cpu#==cpu_busy) cpuutil# += PROB;
<1..NS> RESULT>> if(cpu#==cpu_busy) busycpus += PROB;
<1..NS> RESULT>> if(cpu#==cpu_idle OR cpu#==cpu_busy) goodcpus+=PROB;
<1..NS> RESULT>> if(cpu#==cpu_failed) nfailedcpus += PROB;
RESULT if(busy_terminals>0) busyterm += PROB*busy_terminals);
RESULT>> termutil = busyterm / NT;
RESULT>> if(retrying_terminals>0) retravg += (PROB*retrying_terminals);
RESULT>> if(failedcpus>0) repairutil += PROB;
RESULT if(waiting_terminals>0) waitall += (PROB*waiting_terminals);
RESULT>> resptime = (retravg + waitall) / NT / (prgen * termutil);
RESULT>> overallutil = busycpus + termutil*NT + repairutil;
```

SESSION 4

Queueing systems and Networks

PERFORMANCE OF A PARTIALLY SHARED BUFFER

Dieter Fiems, Bart Steyaert and Herwig Bruneel
SMACS Research Group, Department of Telecommunications and Information Processing
Ghent University, St-Pietersnieuwstraat 41, 9000 Gent, Belgium
E-mail: {df,bs,hb}@telin.UGent.be

ABSTRACT

We assess the performance of a bottleneck buffer in a multimedia network. The buffer temporarily stores packets of different video streams, awaiting for transmission on a single output link. These streams are encoded in a scalable way such that the bottleneck buffer may drop packets of enhancement layers to ensure delivery of base layer packets. In particular, we here adopt Partial Buffer Sharing to reduce the packet loss ratio of base layer packets. The arrival process of the video packets is modelled by means of a two-class discrete batch Markovian arrival process. Using a matrix-analytic approach, we retrieve various performance measures such as the packet loss ratio and the moments of the packet delay. We then illustrate our approach by means of some numerical examples.

1 INTRODUCTION

Scalable video coding is capable to cope with bandwidth fluctuations in packet based networks, see a.o. [1–3] and the references therein. By means of scalable coding techniques, a video stream is encoded in a base layer packet stream and one or more enhancement layer streams. Only the base layer is needed to decode and playback the video, although at a poor quality. Combined with the enhancement layers, the video can be played back at full quality. Intermediate network nodes should therefore drop packets of the enhancement layer to ensure delivery of base layer packets. In this way, the video quality can be reduced gracefully if this is required by the network conditions.

To ensure delivery of base layer packets when the network is congested, network nodes are required to support some sort of Quality of Service (QoS) differentiation. Partial Buffer Sharing (PBS) implements service differentiation by means of a threshold based packet acceptance policy. As long as the number of the packets in the buffer does not exceed a fixed threshold, both high-priority packets (base layer packets) and low-priority packets (enhancement layer packets) are accommodated by the buffer. Once the number of packets in the buffer exceeds the threshold, only high-priority packets are accepted. When the buffer is full, neither high- nor low-priority packets can enter the buffer. As we will see further, a PBS acceptance policy offers space priority at the cost of some overall throughput loss. However, it is quite easily implementable in practice as opposed to e.g. a push-out buffer [4,5]. The push-out policy allows high-priority packets to push out low priority packets if the buffer is full upon arrival.

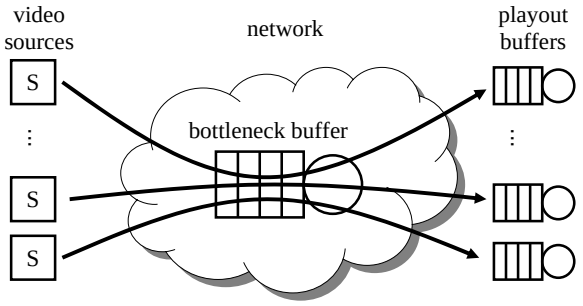


Figure 1: Network topology under consideration

In this contribution, we consider a multimedia network as depicted in figure 1. There are a fixed number of video sources. Each source encodes video information into a base and enhancement packet stream. All packets generated by the different sources traverse through a bottleneck buffer in the network before reaching their destination. The bottleneck buffer gives space priority to the base packets by means of PBS. Before being played out, packets at the destination are temporarily stored in a playout buffer. Playout at the destination then starts after some time: the playout thus allows to cope with delay variation in the network [6]. A packet can therefore be played out if (i) it is not dropped in the network and if (ii) it arrives in time in the playout buffer at the destination. Under the assumption that packet loss and delay in the network mainly come from the bottleneck buffer under consideration, a packet can be played out at the destination if the bottleneck buffer can accommodate the packet and if the delay introduced in the bottleneck buffer is limited.

The remainder of this contribution is organised as follows. In the next section, basic assumptions on the bottleneck buffer and video sources under consideration are given and notation is introduced. Performance of this buffer is then assessed by means of matrix analytic methods in section 3. Some numeri-

cal examples illustrate our approach in section 4 and conclusions are drawn in section 5.

2 QUEUEING MODEL

The bottleneck buffer under consideration operates synchronously. Time is divided into fixed length intervals or slots and all arrivals and departures are synchronised with respect to slot boundaries.

There are two traffic classes, say class 1 and class 2. Packets of these classes arrive in accordance with a discrete-time batch Markovian arrival process with two traffic classes (2-DBMAP), see a.o. Zhao et Al. [7]. Let the state space of the 2-DBMAP be denoted by $\mathcal{Q} = \{1, \ldots, Q\}$. The arrival process is then completely characterised by the doubly indexed sequence $\{A_{n,m}, n, m = 0, 1, \ldots \infty\}$ of substochastic $Q \times Q$ matrices. The matrix $A_{n,m}$ governs the transitions of the Markovian environment of the 2-DBMAP when there are n class 1 and m class 2 arrivals. That is, the ijth element $A_{n,m}(i,j)$ of the matrix $A_{n,m}$ is given by,

$$A_{n,m}(i,j) = \Pr[a_{1,k} = n, a_{2,k} = m, s_k = j | s_{k-1} = i] \,,$$

for $i, j \in \mathcal{Q}$. Here, $a_{1,k}$ and $a_{2,k}$ denote the number of class 1 and class 2 packet arrivals respectively at the kth slot boundary and s_k denotes the state of the 2-DBMAP during the following slot. Notice that a 2-DBMAP can capture time correlation as well as correlation between traffic classes. Also, DBMAPs have been proposed before to model video traffic, see a.o. Lombardo et Al. [8] and Moltchanov et Al. [9].

The transmission time of packets is fixed and equal to the slot length. Up to N packets can be transmitted at a slot boundary and the buffer can accommodate up to M packets, including the packets being transmitted. However, a class 2 packet is only admitted when there are no more than $T \leq M$ packets present upon arrival of this packet in accordance with the PBS acceptance policy as described in the introduction. T is referred to as the threshold.

Since there may be arrivals of both classes as well as departures at a slot boundary, one needs to specify the order in which these take place. We here assume the following order: (1) departures; (2) arrivals of class 1; (3) arrivals of class 2. Observation of the buffer "at slot boundaries" means after the departures but before the arrivals.

3 QUEUEING ANALYSIS

We first retrieve balance equations for the queueing system described above. Expressions for various performance measures in terms of the solution of these balance equations are then obtained.

3.1 Balance equations

Consider a random slot boundary, say slot boundary k and let u_k denote the number of packets in the buffer at this boundary. Further, let $\tilde{a}_{1,k}$ and $\tilde{a}_{2,k}$ denote the number of class 1 and class 2 arrivals at this boundary that the buffer can accommodate. We have,

$$u_{k+1} = (u_k + \tilde{a}_{1,k} + \tilde{a}_{2,k} - N)^+ \,,$$

with,

$$\begin{aligned} \tilde{a}_{1,k} &= \min(a_{1,k}, M - u_k) \,, \\ \tilde{a}_{2,k} &= \min(a_{2,k}, (T - u_k - a_{1,k})^+) \,. \end{aligned} \tag{1}$$

Here $(\cdot)^+$ denotes the standard shorthand notation for $\max(\cdot, 0)$. Notice that we used the fact that class 1 packets enter the buffer before class 2 packets at slot boundaries.

Let $C_{n,m}$ denote the $Q \times Q$ matrix with elements,

$$C_{n,m}(i,j) = \Pr[\tilde{a}_k = m, s_{k+1} = j | u_k = n, s_k = i] \,, \tag{2}$$

for $i, j \in \mathcal{Q}$ and with $\tilde{a}_k = \tilde{a}_{1,k} + \tilde{a}_{2,k}$. That is, $C_{n,m}$ is the matrix governing the transitions of the Markovian environment of the 2-DBMAP when there are m arrivals that the buffer can accommodate, given that there are n packets in the buffer. In view of the system equations above and by conditioning on the number of class 1 and class 2 arrivals during a slot, we find,

$$\begin{aligned} C_{n,m} = \sum_{g,h=0}^{\infty} A_{g,h} \, 1(&\min(g, M - n) \\ &+ \min(h, (T - n - g)^+) = m) \,. \end{aligned}$$

Here $1(\cdot)$ is the standard indicator function that evaluates to 1 if its argument is true and to 0 if this is not the case.

Further, let $D_{n,m}$ denote the $Q \times Q$ matrix with elements,

$$D_{n,m}(i,j) = \Pr[\tilde{a}_k \leq m, s_{k+1} = j | u_k = n, s_k = i] \,, \tag{3}$$

for $i, j \in \mathcal{Q}$. The matrix $D_{n,m}$ governs the transitions of the Markovian environment when there are at most m arrivals that the buffer can accommodate, given that there are n packets in the buffer. The following expression then immediately follows from the definitions (2) and (3),

$$D_{n,m} = \sum_{k=0}^{m} C_{n,k} \,.$$

We now retrieve the stationary distribution of the buffer content at a random slot boundary. Let π_i denote the row vector whose elements are the probabilities to find i packets in the buffer when the Markovian environment of the 2-DBMAP is in state $j \in \mathcal{Q}$,

$$\pi_i = \left[\Pr\left[u_k = i, s_k = j\right] \right]_{j \in \mathcal{Q}} \,,$$

for $i = 0, \ldots, N - M$. Notice that in accordance with equation (1) u_k cannot exceed $M - N$ at random slot

boundaries. In view of equations (1), (2) and (3), we obtain the following set of balance equations for the vectors π_i,

$$\pi_0 = \sum_{n=0}^{N} \pi_n D_{n,N-n}\,,$$
$$\pi_i = \sum_{n=0}^{\min(N+i,M-N)} \pi_n C_{n,N+i-n}\,,$$

for $i = 1, 2, \ldots, M-N$. In block matrix notation we have $\pi = \pi\mathcal{A}$ where the matrix $\mathcal{A}$ has the following block matrix structure,

$$\begin{pmatrix} D_{0,N} & C_{0,N+1} & \cdots & C_{0,M-N} & \cdots & C_{0,M} \\ D_{1,N-1} & C_{1,N} & \cdots & C_{1,M-N-1} & \cdots & C_{1,M-1} \\ \vdots & \vdots & & \vdots & & \vdots \\ D_{N,0} & C_{N,1} & \cdots & C_{N,M-2N} & \cdots & C_{N,M-N} \\ 0 & C_{N+1,0} & \cdots & C_{N+1,M-2N-1} & \cdots & C_{N+1,M-N-1} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & C_{M,N} & \cdots & C_{M-N,N} \end{pmatrix}$$

and with,

$$\pi = [\pi_0, \pi_1, \ldots, \pi_{M-N}]\,.$$

Under mild conditions, the Markov chain corresponding to the buffer system under consideration has only one ergodic class. There then exist a unique stationary distribution π — i.e., a normalised non-negative vector — that solves the balance equations.

If N is a divisor of $M-N+1$, one easily verifies that the matrix corresponding to the set of balance equations has an upper Hessenberg block structure with blocks of size $NQ \times NQ$. If N is not a divisor of $M-N+1$, this matrix still has an upper Hessenberg block structure. Now, all but the blocks on the bottom row and the right column are of size $NQ \times NQ$. In either case, we may use the reduction algorithm of Blondia and Casals [10] to obtain the vectors π_i, $i = 0, \ldots, M-N$ efficiently. The complexity of the algorithm is $O(NQ^3M^2)$. Given the vectors π_i, $i = 0, \ldots, M-N$, we now retrieve various performance characteristics.

3.2 Packet loss ratio

The class 1 and class 2 load — i.e., the mean number of class 1 and class 2 arrivals at a slot boundary, including packets that the buffer cannot accommodate — is given by,

$$\rho_1 = \tau \sum_{n,m=0}^{\infty} nA(n,m)e\,,$$
$$\rho_2 = \tau \sum_{n,m=0}^{\infty} mA(n,m)e\,,$$

where e is a column vector of ones of appropriate size and where τ is the unique normalised solution of,

$$\tau = \tau \sum_{m,n=0}^{\infty} A(m,n)\,.$$

For further use, let $\rho = \rho_1 + \rho_2$ denote the total load, i.e., the sum of the class 1 and class 2 load.

Let $C^{(f)}_{n,m}$ denote the matrix that governs the transitions of the Markovian environment of the arrival process when there are m class f arrivals that the buffer can accommodate, given that there are n packets in the buffer ($f = 1, 2$). That is, the ijth element of $C^{(f)}_{n,m}$ is given by,

$$C^{(f)}_{n,m}(i,j) = \Pr[\tilde{a}_{f,k} = m, s_{k+1} = j | u_k = n, s_k = i]\,.$$

In view of equation (1), we easily retrieve the following expressions for these matrices:

$$C^{(1)}_{n,m} = \sum_{k,l=0}^{\infty} A_{k,l}\, 1(\min(k, M-n) = m)\,,$$

for $n = 0, \ldots, M-N$ and $m = 0, \ldots, M-n$ and,

$$C^{(2)}_{n,m} = \sum_{k,l=0}^{\infty} A_{k,l}\, 1(\min(l, (T-n-k)^+) = m)\,.$$

for $n = 0, \ldots, M-N$ and $m = 0, \ldots, (T-n)^+$.

The effective class 1 load $\tilde{\rho}_1$ and class 2 load $\tilde{\rho}_2$ — i.e., the mean number of packet arrivals of class 1 and class 2 at a random slot boundary that the buffer can accommodate — are therefore given by,

$$\tilde{\rho}_1 = \sum_{n=0}^{M-N} \sum_{m=0}^{M-n} \pi_n m C^{(1)}_{n,m} e\,,$$
$$\tilde{\rho}_2 = \sum_{n=0}^{\min(T,M-N)} \sum_{m=0}^{T-n} \pi_n m C^{(2)}_{n,m} e\,.$$

Also let $\tilde{\rho} = \tilde{\rho}_1 + \tilde{\rho}_2$ denote the total effective load.

The former expressions of load and effective load now allow us to retrieve the packet loss ratio. Recall that the packet loss ratio (plr) of a class is defined as the fraction of all arriving packets of that class that the buffer cannot accommodate. Clearly, the class 1 plr, the class 2 plr and the plr of a random packet are given by,

$$\mathrm{plr}_1 = 1 - \frac{\tilde{\rho}_1}{\rho_1}\,, \quad \mathrm{plr}_2 = 1 - \frac{\tilde{\rho}_2}{\rho_2}\,, \quad \mathrm{plr} = 1 - \frac{\tilde{\rho}}{\rho}\,,$$

respectively.

3.3 Packet delay

Consider a random slot boundary and let $c(l,n,m)$ denote the probability that there are l packets in the buffer and that there are n class 1 and m class 2 packet arrivals that the buffer can accommodate,

$$c(l,n,m) = \Pr[u_k = l, \tilde{a}_{1,k} = n, \tilde{a}_{2,k} = m]$$
$$= \sum_{g,h=0}^{\infty} \pi_l A_{g,h} e\, 1(\min(g, M-l) = n \wedge \min(h, (T-l-g)^+) = m)\,.$$

for $l = 0, \ldots, M-N$, for $n = 0, \ldots, M-l$ and for $m = 0, \ldots, (T-l-n)^+$.

Further, let $\tilde{c}_1(l,n)$ denote the probability that there are n class 1 arrivals that the buffer can accommodate at a random slot boundary and that there are l packets in the buffer at that boundary. We easily retrieve,

$$\begin{aligned}\tilde{c}_1(l,n) &= \Pr[u_k = l, \tilde{a}_{1,k} = n] \\ &= \sum_{m=0}^{(T-l-n)^+} c(l,n,m)\,,\end{aligned}$$

for $l = 0, \ldots, M-N$ and for $n = 0, \ldots, M-l$.

Analogously, let $\tilde{c}_2(l,n)$ denote the probability that there are n class 2 arrivals that the buffer can accommodate at a slot boundary and that there are l packets in the buffer before the class 2 arrival instant at that boundary. Notice that all packets that are in the buffer at the slot boundary as well as all admitted class 1 arrivals are in the buffer immediately before the class 2 arrival instant. We easily retrieve,

$$\begin{aligned}\tilde{c}_2(l,n) &= \Pr[u_k + \tilde{a}_{1,k} = l, \tilde{a}_{2,k} = n] \\ &= \sum_{m=0}^{\min(l,M-N)} c(m, l-m, n)\,,\end{aligned}$$

for $l = 0, \ldots, M$ and $n = 0, \ldots (T-l)^+$.

Let $\nu_i(k)$ denote the fraction of slots where there is an arrival of class i that finds k packets in the buffer upon arrival. We have,

$$\begin{aligned}\nu_1(k) &= \sum_{l=0}^{\min(k,N)} \sum_{n=k-l+1}^{M-l} \tilde{c}_1(l,n)\,, \\ \nu_2(k) &= \sum_{l=0}^{k} \sum_{n=k-l+1}^{(T-l)^+} \tilde{c}_2(l,n)\,.\end{aligned}$$

for $k = 0, 1, \ldots, M-1$ and for $k = 0, 1, \ldots, T-1$ respectively.

As there is at most one such packet arrival at a slot boundary, we find that the probability $u_1(k)$ and $u_2(k)$ that a random class 1 and class 2 packet finds k packets upon arrival in the buffer equals,

$$u_1(k) = \frac{\nu_1(k)}{\tilde{\rho}_1}\,, \qquad u_2(k) = \frac{\nu_2(k)}{\tilde{\rho}_2}\,,$$

for $k = 0, 1, \ldots, M-1$ and for $k = 0, 1, \ldots, T-1$ respectively.

Finally, let packet delay be defined as the number of slots between a packet's arrival and departure slot boundary. Since up to N packets leave the buffer system at a slot boundary, the probability $d_i(n)$ that the delay of a class i packet equals n slots is given by,

$$d_i(n) = \sum_{k=(n-1)N}^{nN-1} u_i(k)\,,$$

for $n = 1, 2, \ldots, \lceil M/N \rceil$ and for $n = 1, 2, \ldots, \lceil T/N \rceil$ for the class 1 and class 2 delay respectively.

4 CASE STUDY

To illustrate our approach, we now return to the network of figure 1 of the introductory section. Let S denote the number of video sources whose packets are routed through the bottleneck buffer and suppose that each source can be modelled as an on/off source. Such a source generates a class 1 packet with probability p and a class 2 packet with probability q when it is on at a slot boundary. No packets are generated when it is off at a slot boundary. Further, given that a source is on (off) at a slot boundary, it is still on (off) at the consecutive slot boundary with probability α (β). That is, the consecutive on-periods (off-periods) constitute a series of geometrically distributed random variables with mean $1/(1-\alpha)$ ($1/(1-\beta)$). The arrival process at the buffer is thus completely characterised by the quintuple (S, p, q, α, β). For further use, we also introduce the following parameters,

$$\sigma = \frac{1-\beta}{2-\alpha-\beta}\,, \qquad K = \frac{1}{2-\alpha-\beta}\,.$$

Here σ denotes the fraction of time a source is on and the burstiness factor K is a measure for the absolute lengths of the on and off periods. K takes values between $\max(\sigma, 1-\sigma)$ and ∞. For $K = 1$ there is no arrival correlation. The quintuple (S, p, q, σ, K) also characterises the arrival process.

In the remainder we assume that there are $S = 8$ video sources. Further, the buffer size equals $M = 100$ packets and there are $N = 4$ servers. The total load (per server) equals $S\sigma(p+q)/N = 80\%$ and sources always produce packets when they are on: $p = 1-q$.

In figure 2(a), the plr of class 1 and class 2 is depicted vs. the threshold for various values of the burstiness factor K. Half of the traffic belongs to class 1 ($p = q = 0.5$). For each value of K, the upper and lower curves depict the class 2 and class 1 plr respectively. For $K = 10$, we also depict the plr of a random packet (middle curve). For increasing values of the threshold T, we observe an exponential decrease of the class 2 plr at the cost of an exponential increase of the class 1 plr. PBS clearly provides service differentiation in terms of the plr. For larger values of T, less class 2 packets are dropped. Therefore the class 2 plr decreases which also implies that there are on average more packets in the buffer. This then in turn implies that more class 1 packets are dropped. I.e., the class 1 plr increases. Further, the plr of a random packet increases for decreasing values of T. I.e., PBS causes additional packet loss. Finally, performance of the buffer system deteriorates when the arrival process is more bursty (larger K). The periods that the buffer is overloaded are then longer implying more packet loss (for both classes).

Figure 2(b) depicts the mean class 1 and class 2 delay vs. T for various values of K. Again, half of

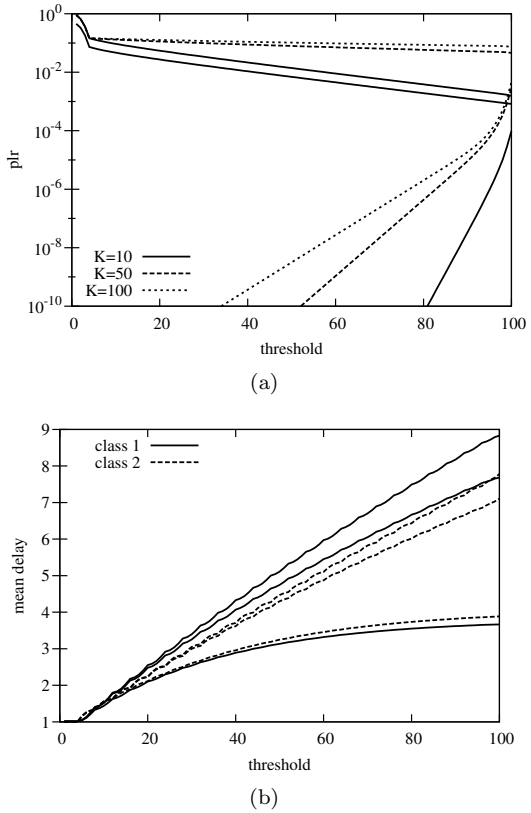


Figure 2: Class 1 and class 2 packet loss ratio (left) and mean delay (right) vs. threshold T

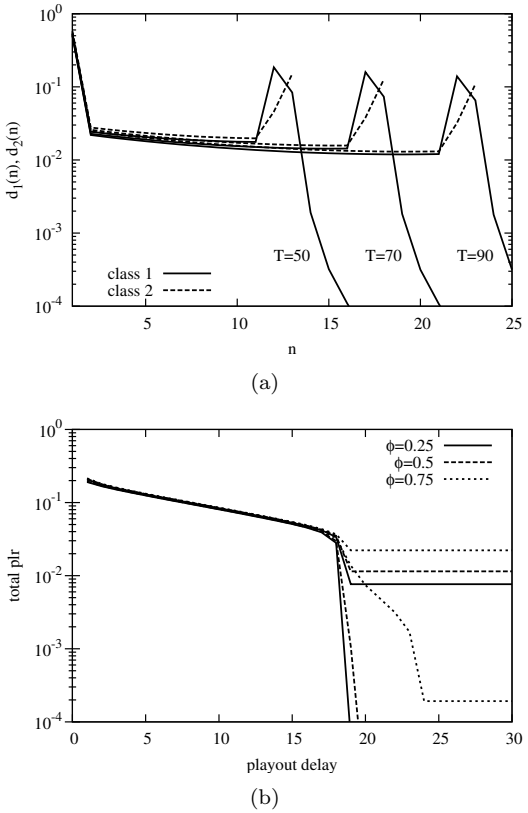


Figure 3: Probability mass function of the class 1 and class 2 delay (left) and total packet loss ratio of class 1 and class 2 vs. playout delay (right)

the traffic belongs to class 1 ($p = q = 0.5$). For each class, the lower, middle and upper curves correspond to $K = 10$, $K = 50$ and $K = 100$ respectively. An increase of T leads to a increase of the mean class 1 and class 2 waiting time. When T increases, class 2 packets are also admitted when there are more packets in the buffer. The waiting times of these packets are of course longer. Similarly, admitting more class 2 packets also implies an increase of the mean waiting of the class 1 packets since every admitted class 2 packet that is present in the buffer upon arrival of a class 1 packet is served before this class 1 packet. Further, notice that the curves are not smooth arcs. The shape comes from the interplay between the threshold and the number of servers. Finally, performance of the buffer system (again) deteriorates for increasing arrival correlation, i.e., for increasing values of K.

In figure 3(a), the probability mass function of the class 1 and class 2 delay are depicted for various values of the threshold T. Half of the traffic belongs to class 1 ($p = q = 0.5$) and the burstiness factor equals $K = 100$. For low delays n ($n \leq \lceil T/N \rceil$), the probability mass functions of the class 1 and class 2 delay are almost the same which is expected since we consider a FIFO buffer. For larger values of n ($n > \lceil T/N \rceil$), we see that longer class 2 delays cannot occur: a class 2 packet is dropped if it finds T packets or more in the buffer upon arrival. For larger values of n, the probability mass function of the class 1 delay quickly decreases. Since class 2 packets are no longer accommodated, the buffer load reduces to the class 1 load which explains the fast decrease.

Finally, figure 3(b) depicts the experienced packet loss ratio of the class 1 and class 2 video flows versus the playout delay. The experienced packet loss ratio includes both the packet loss in the bottleneck buffer as well as loss in the playout buffer due to underflow. For a given playout delay γ, the experienced plr of class i is calculated as follows,

$$\mathrm{plr}_{t,i} = 1 - (1 - \mathrm{plr}_i) \sum_{n=1}^{\gamma} d_i(n)\,.$$

That is, a packet is not lost if it is not dropped in the bottleneck buffer and if its delay does not exceed γ. In figure 3(b), the threshold equals $T = 80$ and the burstiness factor equals $K = 100$. Various values of the traffic mix ϕ — the amount of all arrival traffic that belongs to class 1, $\phi = p/(p+q)$ — are assumed as depicted. For low values of the playout delay, loss is mostly caused by buffer underflow in the playout buffer. Once the playout delay reaches $\lceil T/N \rceil$, class 2 packet loss is no longer possible in

the playout buffer since the delay in the bottleneck buffer is bounded by $\lceil T/N \rceil$. The experienced packet loss ratio of class 2 then equals the class 2 packet loss ratio in the bottleneck buffer. Further, we observe that also the total class 1 packet loss ratio drops fast. This is in accordance with the fast decay of the class 1 delay probability mass function (see figure 3(a)).

5 CONCLUSIONS

We considered a queueing system with partial buffer sharing and batch Markovian arrivals. Using matrix analytic techniques, we obtained various performance measures such as the packet loss ratio and delay of both class 1 and class 2. We then illustrated our approach by means of some numerical examples in the context of video transmission over packet switched networks. We observed that the buffer system under consideration clearly provides service differentiation in terms of the packet loss ratio.

ACKNOWLEDGEMENT

This work has been partly carried out in the framework of the CHAMP and Q-match projects sponsored by the Flemish Institute for the Promotion of Scientific and Technological Research in the Industry (IWT).

REFERENCES

[1] T. Turelli, S.F. Parisis, and J. Bolot. Experiments with a layered transmission scheme over the internet. Technical Report 3295, INRIA, 1997.

[2] H. Radha, Y. Chen, K. Parthasarathy, and R. Cohen. Scalable internet video using MPEG-4. *Signal Processing: Image Communication*, 15:95–126, 1999.

[3] J. Kangasharju, F. Hartanto, M. Reisslein, and K.W. Ross. Distributing layered encoded video through caches. *IEEE Transactions on Computers*, 51(6):622–636, 2002.

[4] H. Kröner, G. Hébuterne, P. Boyer, and A. Gravey. Priority management in ATM switching nodes. *IEEE Journal on Selected Areas in Communications*, 9(3):418–427, 1991.

[5] Jorge García; and Olga Casals. Stochastic models of space priority mechanisms with markovian arrival processes. *Annals of Operations Research*, 35(1-4):271–296, 1992.

[6] T. Hofkens, K. Spaey, and C. Blondia. Transient analysis of the D-BMAP/G/1 queue with an application to the dimensioning of a playout buffer for VBR video. *Lecture Notes in Computer Science*, 3042:1338–1343, 2004.

[7] J. Zhao, B. Li, X. Cao, and I. Ahmad. A matrix-analytic solution for the DBMAP/PH/1 priority queue. *Queueing Systems*, 53(3):127–145, 2006.

[8] A. Lombardo, G. Morabito, and G. Schembra. An accurate and treatable Markov model of MPEG-video traffic. In *Proceedings of 17th Conference on Computer Communications (INFOCOM 1998)*, pages 217–224, San Francisco, California, 1998.

[9] D. Moltchanov, Y. Koucheryavy, and J. Harju. The model of single smoothed MPEG traffic source based on the D-BMAP arrival process with limited state space. In *Proceedings of ICACT 2003*, pages 57–63, Phoenix Park, South Korea, 2003.

[10] C. Blondia and O. Casals. Statistical multiplexing of VBR sources: A matrix-analytic approach. *Performance Evaluation*, 16:5–20, 1992.

Analysis of a generic model for a bottleneck link in an integrated services communications network

Remco Litjens*
TNO ICT
Delft, The Netherlands
Phone +31 15 285 7184
Fax +31 15 285 7355
remco.litjens@tno.nl

Richard J. Boucherie
University of Twente
Enschede, The Netherlands
Phone +31 53 489 3432
Fax +31 53 489 3069
r.j.boucherie@utwente.nl

ABSTRACT

We develop and analyse a generic model for performance evaluation, parameter optimisation and dimensioning of a bottleneck link in an integrated services communications network. Possible application areas include IP, ATM and GSM/GPRS networks. The model enables analytical evaluation for a scenario of integrated speech, video and data services, selected for the fundamental differences in their service characteristics. While a speech call is assigned a single resource unit for its entire duration, both video and data calls can handle varying resource assignments. The principal distinction between these elastic call types, is that in case of video calls, a more generous resource assignment implies a better throughput and thus video quality, while for data calls the increased throughput implies a reduced transfer time. Markov chain analysis is applied to derive basic performance measures such as the expected resource utilization, service-specific blocking probabilities and the expected video and data QOS. Furthermore, analytical expressions are derived for the expected video and data QOS, conditional on the call duration or file size, respectively, and on the system state on arrival. As a potential application, these measures can be fed back to the caller as an indication of the expected QOS. A numerical study, focussing on a wireless access network, is included to demonstrate the merit of the presented generic model and performance analysis.

Keywords

Stochastic models, Markov models, performance modelling, communication systems and networks, wireless and mobile systems and networks.

1. INTRODUCTION

The incredible growth of data and multimedia communications both in wired (e.g. Internet) and wireless (e.g. Wifi) systems is undisputed, as well as the expectation of their convergence in the form of an integrated 'all IP' network handling all communications. Transmissions commonly experience high variation in transmission rates, due to concurrence with other traffic flows. This variation increases due to the inherit differences among the transmission types, such as voice, video and data, that each have their specific characteristics and resource requirements.

*Corresponding author.

Aside from the required technological network upgrades that enable the provisioning of the foreseen variety of services, it is essential to optimise link dimensioning and traffic management mechanisms to efficiently establish the desired service-specific Quality-Of-Service by means of the appropriate deployment of a.o. admission control, resource reservation and packet scheduling policies. Whereas such mechanisms were trivial or irrelevant in single service systems, in integrated services networks they are not only essential to avoid the loss of unsatisfied customers, but also offer an important means for differentiated service provisioning.

Contribution

The principal contribution of the present paper is the development and analysis of a *generic model* for performance evaluation, parameter optimisation and dimensioning in an integrated services communications network. The model enables analytical evaluation for a scenario of integrated speech, video and data services, with service-specific traffic characteristics and potentially offered in distinct priority classes. We note that the considered set of services cover the principal characteristics specifying the different traffic/QOS classes that are standardised for future integrated networks.

In the *performance analysis* presented to determine the QOS of the video and data service, the corresponding dynamics are modelled as queues in a random environment (see e.g. [12]). In our case the influence can be mutual, i.e. the arrival, service and departure process of all the different call types influence each other. Aside from basic performance measures such as service-specific call blocking probabilities, expected resource utilisation and expected video and data QOS, we also derive closed-form expressions for the expected video QOS (throughput) and data QOS (transfer time) conditional on the service requirement and the system state upon call arrival. A *numerical evaluation* is included in the example setting of a GSM/GPRS system to demonstrate the merit of the studied generic model and performance analysis.

Literature

There is a rich variety of models in which, for a single traffic type, the available service rate alternates between a positive value and complete absence of service, including unreliable servers, server vacations and service failures [6, 12, 14]. These models allow for closed-form solutions for many performance measures or structural decomposition results. When the service rate varies between several positive values explicit results are no longer available. Approximations for

transient analysis of single server queues with fluctuating service or arrival rate are studied in [3, 7, 11]. In particular, the mean queue length in a process with varying service rate is shown to exceed that in a process with a constant service rate (with the same mean). These papers consider only the case of a single customer type, and do not allow for generalisation to multiple types with priorities.

Analytical performance studies focusing on the impact of a random environment on the experienced QOS are rare. Based on general results for a queue in a random environment determined by a birth and death process [12], the conditional expected transfer time of data calls in an integrated system with stream and elastic traffic is analysed in [13] for an IP setting, and in [5, 9] for a wireless setting. An integrated system serving prioritised and best effort jobs is investigated in [2], presenting exact closed-form expressions and useful approximations for the expected sojourn times of prioritised and best effort jobs, respectively. The present paper generalises the frameworks mentioned above to also include a distinct service of the video type, for which the transmission time is fixed, but the perceived QOS is determined by the experienced throughput. With the inclusion of the video service, we present a tractable flow level integrated services model that covers all key service types.

Outline

Section 2 presents a generic model for a bottleneck link in an integrated services network, which is extensively analysed in Section 3. Considering a GSM/GPRS system as a possible application area for the generic model and analysis, Section 4 presents a set of illustrative numerical experiments. The proof of a key result is provided in the Appendix.

2. MODEL

This section defines the framework for the performance analysis of a bottleneck link in a communications network integrating speech, video, and data services. See Figure 1.

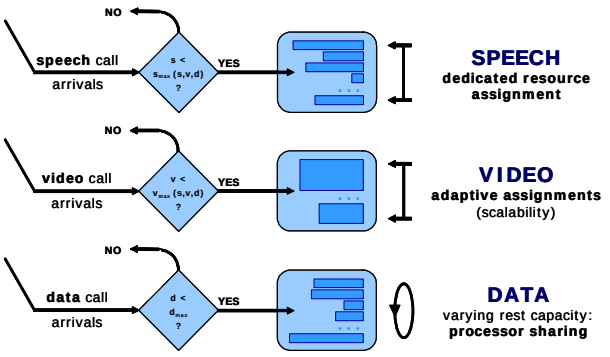


Figure 1: Illustration of the considered model of a bottleneck link in an integrated services communications network.

2.1 Call characteristics

The considered services have been selected for the fundamental differences in their characteristics.

Speech calls arrive according to a Poisson process with arrival intensity λ_{speech}, have an exponentially distributed duration with mean $1/\mu_{\text{speech}}$ and require a fixed assignment of one resource unit. The speech traffic load is given by $\rho_{\text{speech}} \equiv \lambda_{\text{speech}}/\mu_{\text{speech}}$.

Video calls arrive according to a Poisson process with arrival intensity λ_{video} and have an exponentially distributed duration with mean $1/\mu_{\text{video}}$. Video calls are modelled as continuous real-time streams that are *scalable* in the sense that the amount of assigned resources and thus the video quality is adaptive to the varying network load. Scaling is assumed to adhere to any resource reassignment ideally and instantaneously. Denote with r_{video} the fixed video information bit rate (in kbits/s) per assigned resource unit. As a minimum QOS requirement, each video call must be assigned no fewer than $\beta^{\min}_{\text{video}}$ resource units, corresponding to a bit rate of $r_{\text{video}}\,\beta^{\min}_{\text{video}}$ kbits/s. On the other hand, denote with $\beta^{\max}_{\text{video}} \geq \beta^{\min}_{\text{video}}$ the peak resource assignment for video calls, which may be dictated by the service or terminal characteristics. The video traffic load is defined as $\rho_{\text{video}} \equiv \beta^{\min}_{\text{video}}\lambda_{\text{video}}/\mu_{\text{video}}$.

Data calls arrive according to a Poisson process with arrival intensity λ_{data}. A data call is assumed to be the downlink transfer of a file with an exponentially distributed size

The data call size is expressed in units of the data information bit rate of r_{data} kbits/s per resource unit and has mean $1/\mu_{\text{data}}$, which corresponds to $r_{\text{data}}/\mu_{\text{data}}$ kbits. The data traffic load is given by $\rho_{\text{data}} \equiv \lambda_{\text{data}}/\mu_{\text{data}}$. Data calls are assumed to be *elastic* in the sense that they are delay tolerant and can handle varying resource assignments. In contrast to the video service, where the resource assignments do not affect the autonomously sampled call durations, the presence of a data call *is* affected by the resource assignment: the more generous the assignment, the shorted the transfer time. The number of resource units that can be assigned to a data call is limited to the service- or terminal-dictated maximum $\beta^{\max}_{\text{data}}$. We assume that for a given file there is at any time sufficient data available in the buffer feeding the considered communication link to be carried on the dynamically assigned resources. In integrated services models, the assumption of exponential data call sizes, required for analytical tractability, generally leads to some degree of *over*estimation of the expected transfer times, compared to scenarios with a greater call size variability, thus leading to conservative network dimensioning [10].

2.2 Call handling schemes

The considered bottleneck link, integrating speech, video and data services, is characterised by a capacity of C_{total} resource units. The proposed *resource sharing* scheme splits the pool of C_{total} resource units into three distinct subsets: C_{speech}, C_{video}, and C_{data} with $C_{\text{speech}} + C_{\text{video}} + C_{\text{data}} = C_{\text{total}}$. Based on these 'territories', we propose and consider the following resource sharing scheme, which establishes some form of capacity reservation for all service types, while still providing a high resource utilization through varying elastic call assignments.

C_{speech} resource units are reserved for speech calls with preemptive priority, i.e. video and data calls may use these resources whenever they are unused by the speech service, but must free them immediately once needed to support newly admitted speech calls. Furthermore, within these C_{speech} resource units, video calls are treated with strict preference over data calls. C_{video} resource units are shared by all call types with preference for speech and video calls. Video calls must downgrade their assignment (potentially

down to $\beta^{\min}_{\text{video}}$ resource units) only in support of newly admitted speech or video calls. Data calls may utilise the resources that cannot be assigned to the preferred speech or video calls. Note that it is in this resource pool that video calls must find their minimum assignment of $\beta^{\min}_{\text{video}}$ resource units, as resources grabbed elsewhere may have to be released again in favour of newly admitted calls. C_{data} resource units are reserved for data calls with preemptive priority, i.e. video calls can grab free resources in this pool due to their scalability property but only if such resources would otherwise be idle, i.e. if $\beta^{\max}_{\text{data}} d < C_{\text{data}}$, with d the number of present data calls. Speech calls are prohibited to use these resources.

The system evolution can be modelled as a continuous-time Markov chain $(S(t), V(t), D(t))_{t\geq 0}$ where $S(t)$, $V(t)$ and $D(t)$ are defined as the number of speech, video and data calls, respectively, that are present at time t. The system states are denoted (s, v, d) with state space $\mathbb{S}$. Using this notation, the service-specific *call admission control* and *resource assignment* schemes are defined as follows.

A *speech* call is blocked iff upon arrival no resource unit is, or can be made available to support the call, i.e. iff

$$s = s_{\max}(v) \equiv \left\lfloor C_{\text{speech}} + C_{\text{video}} - \beta^{\min}_{\text{video}} v \right\rfloor .$$

A *video* call is blocked iff upon arrival the minimum assignment of $\beta^{\min}_{\text{video}}$ resource units cannot be made available to support the call, i.e. iff

$$v = v_{\max}(s) \equiv \left\lfloor \frac{C_{\text{video}} - \max\{s - C_{\text{speech}}, 0\}}{\beta^{\min}_{\text{video}}} \right\rfloor ,$$

where $\max\{0, s - C_{\text{speech}}\}$ is the number of shared resource units that are in use by speech calls. The number of resource units available for video transfer is $\max\{C_{\text{data}} - \beta^{\max}_{\text{data}} d, 0\} + (C_{\text{video}} + C_{\text{speech}} - s)$. The first part of this expression indicates the resources that are available in the C_{data} pool, while the second part gives the available resources in the joint $C_{\text{video}} + C_{\text{speech}}$ pool. The available resources are then evenly distributed over the present video calls, effectively applying a Processor Sharing service (PS) discipline. The resource assignment function $\beta_{\text{video}}(s, v, d)$ prescribes the amount of resources that is assigned to each admitted video call in system state (s, v, d).

$$\beta_{\text{video}}(s, v, d) \equiv \min\{\beta^{\max}_{\text{video}}, \frac{\max\{C_{\text{data}} - \beta^{\max}_{\text{data}} d, 0\} + (C_{\text{video}} + C_{\text{speech}} - s)}{v}\} .$$

It is readily verified that $\beta_{\text{video}}(s, v, d) \geq \beta^{\min}_{\text{video}}$ for all $(s, v, d) \in \mathbb{S}$. As no minimum assignment is assumed for *data* calls, admission control is simply based on an exogenously given maximum on the number of present data calls in the system, denoted $d_{\max}$. As for video calls, at any time, the resources that are available for the data service are fairly shared by all admitted data calls according to a PS service discipline:

$$\beta_{\text{data}}(s, v, d) \equiv \min\left\{\beta^{\max}_{\text{data}}, \frac{C_{\text{total}} - s - \beta_{\text{video}}(s, v, d) v}{d}\right\} .$$

2.3 On the genericity of the model

A number of extensions and generalisations of the system model and, in particular, the call handling schemes can be made without complicating the performance analysis presented below, as long as model adjustments can be captured by admission control thresholds and resource assignment schemes that have the same general form as those given above. These generalisations have been consciously omitted here for clarity of presentation.

Among the feasible generalisations we mention the following: *(i)* the application of QOS differentiation between different classes of video and/or data calls; *(ii)* the introduction a service-specific FCFS access queue to hold calls that cannot be admitted immediately upon arrival; *(iii)* analysis of different resource sharing schemes; *(iv)* consideration of minimum resource assignments for data calls to ensure some minimum QOS; *(v)* restriction of (data or) video call assignments to be limited to a number of prefixed levels, e.g. 2, 4 or 8 resource units if this corresponds more accurately to an assumed scalable video coding algorithm (see e.g. [1]).

3. PERFORMANCE ANALYSIS

In this section, the system evolution of the model is formulated as a continuous-time Markov chain. Some basic performance measures and an extensive conditional analysis is presented for video and data calls, deriving the expected QOS as a function of the call duration/size and the system state upon call arrival.

3.1 Markov chain and equilibrium

The evolution of the system model can be described by an irreducible three-dimensional continuous-time Markov chain $(S(t), V(t), D(t))_{t\geq 0}$, with states denoted (s, v, d). The state space of the Markov chain is given by

$$\mathbb{S} \equiv \left\{ \; (s, v, d) : s \leq s_{\max}(v) \wedge v \leq v_{\max}(s) \wedge d \leq d_{\max} \; \right\} .$$

The speech, video and data call arrival rates are given by λ_{speech}, λ_{video} and λ_{data}, while the speech, video and data call departure rates in system state (s, v, d) are given by $s\mu_{\text{speech}}$, $v\mu_{\text{video}}$ and $\beta_{\text{data}}(s, v, d)d\mu_{\text{data}}$, respectively. Ordering $\mathbb{S}$ lexicographically in (s, v, d), the infinitesimal generator $\mathcal{Q}$ is of tridiagonal block structure, with above-diagonal blocks generating speech call arrivals, below-diago-nal blocks generating speech call terminations, and diagonal blocks generating video and data call arrival and termination events.

Since the considered finite state space Markov chain $(S(t), V(t), D(t))_{t\geq 0}$ is irreducible, a unique probability vector π exists that satisfies the system of global balance equations: $\boldsymbol{\pi}\mathcal{Q} = \mathbf{0}$, with $\mathbf{0}$ the vector with all entries zero, and $\boldsymbol{\pi}$ lexicographically ordered in $(s, v, d) \in \mathbb{S}$.

3.2 Basic performance measures

From a system's perspective, the resource efficiency can be measured by the *expected resource utilization:*

$$\mathbf{U} \equiv C^{-1}_{\text{total}} \sum_{(s,v,d)\in\mathbb{S}} \pi(s, v, d) \begin{pmatrix} s + \beta_{\text{video}}(s, v, d)v \\ +\beta_{\text{data}}(s, v, d)d \end{pmatrix} .$$

The service-specific *blocking probabilities* are readily derived from the equilibrium distribution using using the PASTA property. The video QOS is expressed in the *expected video throughput*. As the measure indicates the expected *per-call* video throughput, we must condition on the presence of at least one video call, obtaining

$$\mathbf{R}_{\text{video}} \equiv r_{\text{video}} \left(\frac{\sum_{(s,v,d)\in\mathbb{S}^+_{\text{video}}} \pi(s, v, d)\beta_{\text{video}}(s, v, d)}{\sum_{(s,v,d)\in\mathbb{S}^+_{\text{video}}} \pi(s, v, d)} \right) ,$$

with $\mathbb{S}^+_{\rm video} \equiv \{(s,v,d) \in \mathbb{S} : v > 0\}$. The data QOS is expressed in the *expected transfer time* of a data call, which is readily obtained using Little's law:

$$\mathbf{T}_{\rm data} \equiv \frac{\sum_{(s,v,d)\in\mathbb{S}} \pi(s,v,d)d}{\lambda_{\rm data}(1-\mathbf{P}_{\rm data})}. \quad (1)$$

Another relevant measure characterizing the data QOS is the *expected data throughput,* which is given by

$$\mathbf{R}_{\rm data} \equiv r_{\rm data}\left(\frac{\sum_{(s,v,d)\in\mathbb{S}^+_{\rm data}} \pi(s,v,d)\beta_{\rm data}(s,v,d)}{\sum_{(s,v,d)\in\mathbb{S}^+_{\rm data}} \pi(s,v,d)}\right),$$

with $\mathbb{S}^+_{\rm data} \equiv \{(s,v,d) \in \mathbb{S} : d > 0\}$.

3.3 Conditional performance measures

3.3.1 Conditional analysis of the video QOS

While $\mathbf{R}_{\rm video}$ is a *time*-average video throughput measure, in this section a *call*-average throughput measure is determined, which is undeniably the most appropriate throughput measure from a call's perspective.

For each state $(s,v,d) \in \mathbb{S}^+_{\rm video}$ define $x_{s,v,d}(\tau)$ as the random number of bits (*transfer volume*) transmitted by an admitted video call of duration τ, arriving at a given system state (s,v,d), where v *includes* the new video call, and let $\widehat{x}_{s,v,d}(\tau) \equiv \mathbf{E}\{x_{s,v,d}(\tau)\}$ denote its expectation. Then the corresponding expected throughput is equal to $\widehat{x}_{s,v,d}(\tau)/\tau$, while the expected throughput of an *admitted* video call of duration τ is given by

$$\mathbf{R}^\star_{\rm video}(\tau) \equiv \frac{\sum_{(s,v,d)\in\mathbb{S}^+_{\rm video}} \pi(s,v-1,d)\widehat{x}_{s,v,d}(\tau)}{\tau\,(1-\mathbf{P}_{\rm video})} \quad (2)$$

where $\pi(s,v-1,d)/(1-\mathbf{P}_{\rm video})$ is the equilibrium probability that the system is in state $(s,v-1,d)$, conditioned on the admission of an arriving video call. Integrating $\mathbf{R}^\star_{\rm video}(\tau)$ over the probability density function of τ yields the expected (call-average) throughput of an admitted video call:

$$\mathbf{R}^\star_{\rm video} \equiv \int_{\tau=0}^{\infty} \mathbf{R}^\star_{\rm video}(\tau)\,\mu_{\rm video}\,\exp\{-\tau\,\mu_{\rm video}\}\,d\tau.$$

We stress that in general the *time*-average video throughput $\mathbf{R}_{\rm video}$ and the *call*-average video throughput $\mathbf{R}^\star_{\rm video}$ need not be the same. Refer to [8] for a more extensive comparison of throughput measures in PS models. It is noted that the values of $\widehat{x}_{s,v,d}(\tau)/\tau$, $(s,v,d) \in \mathbb{S}^+_{\rm video}$, may be at least as valuable as $\mathbf{R}^\star_{\rm video}(\tau)$ or $\mathbf{R}^\star_{\rm video}$, since, given the system state at arrival, the appropriate value can be fed back to the source as an indication of the expected service quality.

In the following an explicit expression for the vector $\widehat{\mathbf{x}}(\tau) = \left(\widehat{x}_{s,v,d}(\tau),\ (s,v,d) \in \mathbb{S}^+_{\rm video}\right)$ is derived. To this end, we introduce the generator $\mathcal{Q}^\star_{\rm video}$, that is characterised by the presence of *one permanent video call* that never leaves the system, and shares in the available resources as if it were an ordinary video call. This permanent video call is the tagged call whose throughput is to be determined, while the behaviour of all other calls is unchanged. For all $(s,v,d) \in \mathbb{S}^+_{\rm video}$, the video call departure rates are modified as follows:

$$\mathcal{Q}^\star_{\rm video}\left((s,v,d);(s,v-1,d)\right) = (v-1)\,\mu_{\rm video}.$$

Let $\boldsymbol{\pi}^\star_{\rm video} \equiv (\pi^\star_{\rm video}(s,v,d),\ (s,v,d) \in \mathbb{S}^+_{\rm video})$ be the stationary distribution, i.e. $\boldsymbol{\pi}^\star_{\rm video}\mathcal{Q}^\star_{\rm video} = \mathbf{0}$. Let $\mathcal{B}_{\rm video} \equiv diag(\beta_{\rm video}(s,v,d),\ (s,v,d) \in \mathbb{S}^+_{\rm video})$ denote the diagonal matrix of average resource assignments, lexicographically ordered in (s,v,d). We may now formulate the following expression of the conditional expected transfer volume.

THEOREM 1. *Let* $\boldsymbol{\gamma}_{video} \equiv (\gamma_{video}(s,v,d),\ (s,v,d) \in \mathbb{S}^+_{video})$ *be the unique solution to*

$$\mathcal{Q}^\star_{video}\boldsymbol{\gamma}_{video} = (\boldsymbol{\pi}^\star_{video} r_{video}\mathcal{B}_{video}\mathbf{1})\ \mathbf{1} - r_{video}\mathcal{B}_{video}\mathbf{1}, \quad (3)$$

$$\boldsymbol{\pi}^\star\boldsymbol{\gamma}_{video} = \mathbf{0}. \quad (4)$$

Then the conditional expected throughput vector is given by

$$\frac{\widehat{\mathbf{x}}(\tau)}{\tau} = (\boldsymbol{\pi}^\star_{video} r_{video}\mathcal{B}_{video}\mathbf{1})\,\mathbf{1} + \tau^{-1}\left[\mathcal{I} - \exp\{\tau\mathcal{Q}^\star_{video}\}\right]\boldsymbol{\gamma}_{video},$$

which asymptotically converges to

$$\lim_{\tau\to\infty}\frac{\widehat{\mathbf{x}}(\tau)}{\tau} = (\boldsymbol{\pi}^\star_{video} r_{video}\mathcal{B}_{video}\mathbf{1})\,\mathbf{1}.$$

Hence the (conditional) expected (call-average) video throughput is asymptotically equal to the (conditional) expected (time-average) video throughput in a system with one permanent video call.

PROOF. See Appendix A. □

Observe that the asymptotic expression, given by $\widehat{\mathbf{x}}(\tau)/\tau = (\boldsymbol{\pi}^\star_{\rm video} r_{\rm video}\mathcal{B}_{\rm video}\mathbf{1})\,\mathbf{1} + \boldsymbol{\gamma}_{\rm video}/\tau$, is non-linear in τ, as will also be illustrated in Section 4. We further note that in a model without (noticeable) data traffic, i.e. if $\lambda_{\rm data} = 0$ and/or $C_{\rm data} = 0$, the asymptotic video throughput expression given above, holds not only for $\tau \to \infty$, but for any (finite) τ. Hence in this scenario the call-average video throughput is independent of the video call duration τ, and identical to the time-average throughput in a system with one permanent video call.

3.3.2 Conditional analysis of the data QOS

As demonstrated in Section 3.2, the expected transfer time $\mathbf{T}_{\rm data}$ of a data call is readily calculated from the equilibrium distribution $\pi(s,v,d)$, $(s,v,d) \in \mathbb{S}$. In this subsection, we determine $\mathbf{T}_{\rm data}(x)$, the expected transfer time of an admitted data call of size $x \geq 0$. Compared to $\mathbf{T}_{\rm data}$, the analysis of $\mathbf{T}_{\rm data}(x)$ is considerably more complicated. Since the conditional analysis is analogous to that presented in [9, 12, 13], we only state the main results here for completeness.

For each state $(s,v,d) \in \mathbb{S}^+_{\rm data}$ define $\tau_{s,v,d}(\tau)$ as the random transfer time of an admitted data call of size x, arriving at system state (s,v,d), where d *includes* the new data call, and let $\widehat{\tau}_{s,v,d}(x) \equiv \mathbf{E}\{\tau_{s,v,d}(x)\}$ denote its expectation. Then the expected transfer time of an *admitted* data call of size x is given by

$$\mathbf{T}_{\rm data}(x) \equiv \frac{\sum_{(s,v,d)\in\mathbb{S}^+_{\rm data}} \pi(s,v,d-1)\widehat{\tau}_{s,v,d}(x)}{1-\mathbf{P}_{\rm data}}. \quad (5)$$

The integral of $\mathbf{T}_{\rm data}(x)$ over all possible values of τ yields the expected (call-average) throughput $\mathbf{T}_{\rm data}$, see (1). As was noted for the video service, the obtained values of $\widehat{\tau}_{s,v,d}(x)$, $(s,v,d) \in \mathbb{S}^+_{\rm data}$, may be at least as valuable as $\mathbf{T}_{\rm data}(x)$ or $\mathbf{T}_{\rm data}$, since they can be fed back to the source as an indication of how long the transmission is expected to take.

In the following an explicit expression for the vector $\widehat{\boldsymbol{\tau}}(x) = (\widehat{\tau}_{s,v,d}(x),\ (s,v,d) \in \mathbb{S}^+_{\rm data})$, $x \in \mathbb{R}^+$, is derived. In a similar way as done in the video QOS analysis, for the data

transfer time analysis denote with $\mathcal{Q}^{\star}_{\rm data}$ the infinitesimal generator of the modified Markov chain, characterised by the presence of *one permanent data call*. Furthermore, let $\mathcal{B}_{\rm data} \equiv diag(\beta_{\rm data}(s,v,d),\ (s,v,d) \in \mathbb{S}^{+}_{\rm data})$ denote the diagonal matrix of average data resource assignments, lexicographically ordered in (s,v,d).

In case $C_{\rm data} = 0$, it may occur that no resources are available to the data calls, i.e. $\beta_{\rm data}\,(s,v,d) = 0$ for some states $(s,v,d) \in \mathbb{S}^{+}_{\rm data}$, depending on the model parameters. Partition $\mathbb{S}^{+}_{\rm data}$ into $\mathbb{S}^{+}_{\rm data,0} \equiv \{(s,v,d) \in \mathbb{S}^{+}_{\rm data} : \beta_{\rm data}\,(s,v,d) = 0\}$ and its complement $\mathbb{S}^{+}_{\rm data,+} \equiv \mathbb{S}^{+}_{\rm data} \backslash \mathbb{S}^{+}_{\rm data,0}$, and accordingly partition

$$\mathcal{Q}^{\star}_{\rm data} = \begin{bmatrix} \mathcal{Q}^{\star}_{++} & \mathcal{Q}^{\star}_{+0} \\ \mathcal{Q}^{\star}_{0+} & \mathcal{Q}^{\star}_{00} \end{bmatrix},\ \mathcal{B}_{\rm data} = \begin{bmatrix} \mathcal{B}_{+} & \mathcal{O} \\ \mathcal{O} & \mathcal{O} \end{bmatrix},$$

Let $\boldsymbol{\pi}^{\star}_{\rm data} \equiv (\pi^{\star}_{\rm data}(s,v,d),\ (s,v,d) \in \mathbb{S}^{+}_{\rm data})$ be the stationary distribution, i.e. $\boldsymbol{\pi}^{\star}_{\rm data}\mathcal{Q}^{\star}_{\rm data} = \mathbf{0}$, and apply the partitionining $\boldsymbol{\pi}^{\star}_{\rm data} = (\boldsymbol{\pi}^{\star}_{\rm data,0},\ \boldsymbol{\pi}^{\star}_{\rm data,+})$. For the general case where $\mathbb{S}^{+}_{\rm data,0} \neq \emptyset$, Theorem 2 presents analytical expressions for the conditional expected transfer times $\widehat{\boldsymbol{\tau}}(x)$.

THEOREM 2. *Let* $\boldsymbol{\gamma}_{data} \equiv (\gamma_{data}\,(s,v,d),\ (s,v,d) \in \mathbb{S}^{+}_{data,+})$ *uniquely solve the system of linear equations*

$$\mathcal{B}_{+}^{-1}\left(\mathcal{Q}^{\star}_{++} + \mathcal{Q}^{\star}_{+0}\left(-\mathcal{Q}^{\star}_{00}\right)^{-1}\mathcal{Q}^{\star}_{0+}\right)\boldsymbol{\gamma}_{data} = \frac{1}{\boldsymbol{\pi}^{\star}_{data,+}\mathcal{B}_{+}\mathbf{1}}\mathbf{1} - \mathcal{B}_{+}^{-1}\left(\mathcal{I} + \mathcal{Q}^{\star}_{+0}\left(-\mathcal{Q}^{\star}_{00}\right)^{-1}\right)\mathbf{1},$$

$$\boldsymbol{\pi}^{\star}_{data,+}\mathcal{B}_{+}\boldsymbol{\gamma}_{data} = 1.$$

The solution for $\widehat{\boldsymbol{\tau}}(x) = (\widehat{\boldsymbol{\tau}}_0(x), \widehat{\boldsymbol{\tau}}_{+}(x))$ *is then given by*

$$\widehat{\boldsymbol{\tau}}_0(x) = \left(-\mathcal{Q}^{\star}_{00}\right)^{-1}\left\{\mathbf{1} + \mathcal{Q}^{\star}_{0+}\widehat{\boldsymbol{\tau}}_{+}(x)\right\},$$

$$\widehat{\boldsymbol{\tau}}_{+}(x) = \frac{x}{\boldsymbol{\pi}^{\star}_{data,+}\mathcal{B}_{+}\mathbf{1}}\mathbf{1} + \left[\mathcal{I} - \exp\left\{x\mathcal{B}_{+}^{-1}\left(\mathcal{Q}^{\star}_{++} + \mathcal{Q}^{\star}_{+0}\left(-\mathcal{Q}^{\star}_{00}\right)^{-1}\mathcal{Q}^{\star}_{0+}\right)\right\}\right]\boldsymbol{\gamma}_{data},$$

while the asymptotic expressions are given by

$$\lim_{x\to\infty}\left\{\widehat{\boldsymbol{\tau}}_0(x) - \frac{x}{\boldsymbol{\pi}^{\star}_{data,+}\mathcal{B}_{+}\mathbf{1}}\mathbf{1}\right\} = \left(-\mathcal{Q}^{\star}_{00}\right)^{-1}\mathbf{1} + \left(-\mathcal{Q}^{\star}_{00}\right)^{-1}\mathcal{Q}^{\star}_{0+}\boldsymbol{\gamma}_{data}$$

and

$$\lim_{x\to\infty}\left\{\widehat{\boldsymbol{\tau}}_{+}(x) - \frac{x}{\boldsymbol{\pi}^{\star}_{data,+}\mathcal{B}_{+}\mathbf{1}}\mathbf{1}\right\} = \boldsymbol{\gamma}_{data}, \qquad (6)$$

indicating that for large data calls the expected transfer time is approximately linear in the size (fairness).

PROOF. The proof is a rather straightforward extension of Corollary 5.2 in [12] and therefore omitted. □

4. NUMERICAL RESULTS

This section presents a brief numerical study in order to demonstrate the merit of the considered model and performance analysis. Concentrating on a single cell in a GSM/GPRS radio access network as a typical example setting, Table 1 below gives an overview of all model parameters. Some comments regarding these parameters are made below.

SYSTEM PARAMETERS		CALL CHARACTERISTICS	
$C_{\rm total}$	21 channels	$\mu^{-1}_{\rm speech}$	50 seconds
$C_{\rm speech}$	12 channels	$\mu^{-1}_{\rm video}$	50 seconds
$C_{\rm video}$	6 channels	$\mu^{-1}_{\rm data}$	35.3591 seconds
$C_{\rm data}$	3 channels	$\beta^{\rm min}_{\rm video}$	2 channels
$\beta^{\rm max}_{\rm video}$	4 channels	$r_{\rm video}$	13.40 kbits/s
$\beta^{\rm max}_{\rm data}$	4 channels	$r_{\rm data}$	9.05 kbits/s
Q_t	10 data calls	$\varphi_{\rm speech}$	0.356625
		$\varphi_{\rm video}$	0.015547
		$\varphi_{\rm data}$	0.627828

Table 1: *Numerical results: parameter settings.*

The capacity of the considered GSM/ GPRS cell is prefixed by a typical assignment of 3 frequencies, which according to GSM's FD/TDMA technology provides $3 \times 8 = 24$ physical channels. Assigning 3 channels for control signalling purposes, this leaves $C_{\rm total} = 21$ traffic channels. As a typical GPRS terminal is characterised by a multichannel capability of four traffic channels, the upper bounds on the resource assignment is given by $\beta^{\rm max}_{\rm video} = \beta^{\rm max}_{\rm data} = 4$.

The average speech and video call holding time are both equal to 50 seconds. Assuming an average file size of 320 kbits and GPRS coding scheme CS-1 for maximum error correction potential, $r_{\rm data} = 9.05$ kb/s and hence the normalised data call size has an average of $320/9.05 = 35.3591$ transmission seconds, given a single dedicated channel assignment. For video calls, the less protective coding scheme CS-2 is assumed, which provides a channel rate of $r_{\rm video} = 13.4$ kb/s. The service mix is defined by the arrival fractions $\varphi_{\rm speech}$, $\varphi_{\rm video}$ and $\varphi_{\rm data}$, with $\varphi_{\rm speech} + \varphi_{\rm video} + \varphi_{\rm data} = 1$, which are determined as follows. Considering the given channel pool partitioning and channel sharing schemes, we determine for each service individually the maximum arrival rate such that the blocking probability is no more than 1%, assuming an otherwise empty system. This exercise yields $\lambda_{\rm speech} = 0.20874$, $\lambda_{\rm video} = 0.0091$ and $\lambda_{\rm data} = 0.36748$, which in turn yields the given relative arrival fractions.

4.1 Basic performance measures

In the first set of experiments we show the impact of the traffic load on the different basic performance measures. The traffic load is varied via the aggregate arrival rate $\lambda_{\rm speech} + \lambda_{\rm video} + \lambda_{\rm data}$, while fixing the ratio $\lambda_{\rm speech} : \lambda_{\rm video} : \lambda_{\rm data}$ to $\varphi_{\rm speech} : \varphi_{\rm video} : \varphi_{\rm data}$.

The results are depicted in the upper pair of charts in Figure 2. Not surprisingly, in the left chart the channel utilisation and service-specific blocking probabilities are increasing in the traffic load. Observe that, although the same target blocking probability of 1% was considered to determine the default arrival rates, the obtained blocking probabilities at the default traffic load are not only significantly larger than 1%, due to presence of other, competing traffic, but also significantly different, due to the distinct policies in the territories. The right chart depicts the time-average video and data throughput measures, the expected data transfer time, as well as derived (asymptotic) approximation for the call-average video throughput. Observe that both video throughput curves are very similar. For low traffic loads, the QOS curves are flat at the best achievable levels, imposed by the limitations imposed by $\beta^{\rm max}_{\rm video,data}$, while under heavier traffic all curves show monotonously worsening QOS

levels. Observe that the video QOS remains well above its minimum guarantee of $r_{\text{video}}\beta^{\min}_{\text{video}} = 26.8$ kb/s, indicating that the system is not really congested yet, from the video service's perspective.

4.2 Conditional performance measures

The middle pair of charts in Figure 2 show the conditional expected video QOS. The 3D chart on the left depicts the expected video throughput $\widehat{x}_{s,v,0}(\tau)/\tau$ experienced by a tagged video call of average duration ($\tau = 50$) as a function of s and v ($d = 0$). Note the domain $\{(s, v, 0) : s + \beta^{\min}_{\text{video}}\,(v + 1) \leq C_{\text{speech}} + C_{\text{video}} = 18$ and $v+1 \leq C_{\text{video}}/\beta^{\min}_{\text{video}} = 3\}$. As expected, $\widehat{x}_{s,v,0}(\tau)/\tau$ is decreasing in both s and v, while it is most sensitive to a change in the number of *video* calls, since an additional video call claims $\beta^{\min}_{\text{video}} = 2$ times as many traffic channels as an additional speech call. The chart on the right demonstrates the conditional video QOS conditioned only on the video call duration, as well as the corresponding derived asymptote, which appears to provide a very tight approximation, and both an upper and lower bound, corresponding with the best- ($\widehat{x}_{0,1,0}(\tau)/\tau$) and worst-case curves ($\widehat{x}_{12,3,10}(\tau)/\tau$) of the conditional expected QOS, given an empty or full system upon arrival of the considered call, respectively (see dashed curves).

Similarly, the lower pair of charts in Figure 3 show the conditional expected data QOS. The 3D chart on the left depicts the expected transfer time $\widehat{\tau}_{s,0,d}(x)$ experienced by a tagged data call of average size ($x = 320/9.05 = 35.3591$) as a function of s and d ($v = 0$). Theorem 2 has been applied to obtain the conditional expected data call transfer times. The right chart shows the conditional expected transfer time, conditioned only on the (normalised) data call size, accompanied by the derived asymptote and the upper/lower bounds, given by with $\widehat{\tau}_{12,3,10}(x)$ and $\widehat{\tau}_{0,0,1}(x)$, respectively.

5. CONCLUDING REMARKS

We have developed and analysed a generic model for performance evaluation, parameter optimisation and dimensioning of a bottleneck in an integrated services communication network. Markov chain analysis applying both dedicated resource assignments and Processor Sharing-type service disciplines, has been applied to derive a variety of performance measures, including exact expressions for the expected video and data QOS, conditional on the call duration or file size, respectively, and on the system state of arrival.

A number of extensions and generalisations of the system model and, in particular, the call handling schemes can be made without complicating the performance analysis presented below, e.g. *(i)* the application of QOS differentiation between different classes of video and/or data calls; *(ii)* the introduction a service-specific FCFS access queue to hold calls that cannot be admitted immediately upon arrival; *(iii)* analysis of different resource sharing schemes; *(iv)* restriction of (data or) video call assignments to be limited to certain prefixed levels, if this corresponds more accurately to an assumed scalable video coding algorithm (see e.g. [1]).

6. REFERENCES

[1] L. Begain, "Scalable multimedia services in GSM-based networks: an analytical approach," *Proc. of the ITC specialist seminar on Mobile systems and mobility*, Lillehammer, Norway, 2000.

[2] J.L. van den Berg, R. van der Mei, B. Gijsen, M. Pikaart and R. Vranken, "Processing times for transaction servers with quality of service differentiation", *Proc. of the 11th GI/ITG conference on Measuring, modelling and evaluation of computer and communications systems*, Aachen, Germany, 2001.

[3] S.K. Cheung, R. Nunez-Queija and R.J. Boucherie, "Effective load and adjusted stability in queues with fluctuating service rates", *Internal report*, Universiteit Twente, The Netherlands, 2006

[4] E.A. Coddington and N. Levinson, *"Theory of ordinary differential equations"*, McGraw-Hill, USA, 1955.

[5] G. Fodor and M. Telek, "Performance analysis of the uplink of a CDMA cell supporting elastic services, *Proc. of Networking '05*, Waterloo, Canada, 2005.

[6] S. Fuhrmann and R. Cooper, "Stochastic decompositions in the M/G/1 queue with generalized vacations", *Operations research*, 33 (5), 1985.

[7] V. Gupta, M. Harchol-Balter, A.S. Wolf and U. Yechiali, "Fundamental characteristics of queues with fluctuating load", *Proc. of Sigmetrics/ Performance '06*, Saint Malo, France, 2006.

[8] R. Litjens, J.L. van den Berg and R.J. Boucherie, "Throughput measures for processor sharing models," *submitted*, 2003.

[9] R. Litjens and R.J. Boucherie, " Performance analysis of fair channel sharing policies in an integrated cellular voice/data network," *Telecommunications systems*, 19 (2), 2002.

[10] R. Litjens and R.J. Boucherie, "Elastic calls in an integrated services network: the greater the variability the better the Quality-of-Service," *Performance evaluation*, 52 (4), 2003.

[11] W.A. Massey and W. Whitt, "Uniform acceleration expansions for Markov chains with time-varying rates", *Annals of applied probability*, 8 (4), 1997.

[12] R. Núñez Queija, "Sojourn times in non-homogeneous QBD processes with processor sharing," *Stochastic models*, 17, 2001.

[13] R. Núñez Queija, J.L. van den Berg, and M.R.H. Mandjes, "Performance evaluation of strategies for integration of elastic and stream traffic," *Proc. of ITC 16*, Edinburgh, Scotland, 1999.

[14] H. Takagi, *"Queueing analysis, vacations and priority systems"*, part 1, vol. 1, Elsevier Science Publishers, The Netherlands, 1991.

[15] H.C. Tijms, *"Stochastic modelling and analysis: a computational approach,"* Wiley, England, 1986.

APPENDIX

A. PROOF OF THEOREM 1

Define the Laplace-Stieltjes transform of the distribution of $x_{s,v,d}(\tau)$ by $X_{s,v,d}(\zeta, \tau) \equiv \mathbf{E}\left\{\exp\left\{-\zeta\, x_{s,v,d}(\tau)\right\}\right\}$, for $\text{Re}(\zeta) \geq 0$, $(s, v, d) \in \mathbb{S}^{+}_{\text{video}}$, and let $\mathbf{X}(\zeta, \tau) \equiv (X_{s,v,d}(\zeta, \tau),$ $(s, v, d) \in \mathbb{S}^{+}_{\text{video}})$ be lexicographically ordered.

LEMMA 1. *For $\tau \geq 0$ and $\text{Re}(\zeta) \geq 0$, $\mathbf{X}(\zeta, \tau)$ satisfies the*

following differential equation and initial condition:

$$\frac{\partial}{\partial\tau}\mathbf{X}(\zeta,\tau) = \left(\mathcal{Q}^{\star}_{video} - \zeta r_{video}\mathcal{B}_{video}\right)\mathbf{X}(\zeta,\tau), \quad (7)$$

$$\mathbf{X}(\zeta,0) = \mathbf{1}, \quad (8)$$

and hence the unique solution is given by

$$\mathbf{X}(\zeta,\tau) = \exp\left\{\tau\left(\mathcal{Q}^{\star}_{video} - \zeta r_{video}\mathcal{B}_{video}\right)\right\}\mathbf{1}. \quad (9)$$

PROOF. Consider a time interval of length $\Delta > 0$, with Δ sufficiently small such that the tagged video call cannot terminate within this time. Condition on all the possible events occurring in this interval, starting out in state $(s,v,d) \in \mathbb{S}^{+}_{\text{video}}$ (for notational convenience and readability, the boundary constraints are not explicitly considered):

$$X_{s,v,d}(\zeta,\tau) \equiv \mathbf{E}\left\{\exp\left\{-\zeta\, x_{s,v,d}(\tau)\right\}\right\}$$

$$\begin{aligned}
= \;& \lambda_{\text{speech}}\Delta X_{s+1,v,d}(\zeta,\tau-\Delta) \\
&\quad \times \exp\left[-\zeta r_{\text{video}}\left(\begin{array}{c}\beta_{\text{video}}(s,v,d)\left(\Delta - O(\Delta)\right)\\ +\beta_{\text{video}}(s+1,v,d)\,O(\Delta)\end{array}\right)\right] \\
&+ s\mu_{\text{speech}}\Delta X_{s-1,v,d}(\zeta,\tau-\Delta) \\
&\quad \times \exp\left[-\zeta r_{\text{video}}\left(\begin{array}{c}\beta_{\text{video}}(s,v,d)\left(\Delta - O(\Delta)\right)\\ +\beta_{\text{video}}(s-1,v,d)\,O(\Delta)\end{array}\right)\right] \\
&+ \lambda_{\text{video}}\Delta X_{s,v+1,d}(\zeta,\tau-\Delta) \\
&\quad \times \exp\left[-\zeta r_{\text{video}}\left(\begin{array}{c}\beta_{\text{video}}(s,v,d)\left(\Delta - O(\Delta)\right)\\ +\beta_{\text{video}}(s,v+1,d)\,O(\Delta)\end{array}\right)\right] \\
&+ (v-1)\,\mu_{\text{video}}\Delta X_{s,v-1,d}(\zeta,\tau-\Delta) \\
&\quad \times \exp\left[-\zeta r_{\text{video}}\left(\begin{array}{c}\beta_{\text{video}}(s,v,d)\left(\Delta - O(\Delta)\right)\\ +\beta_{\text{video}}(s,v-1,d)\,O(\Delta)\end{array}\right)\right] \\
&+ \lambda_{\text{data}}\Delta X_{s,v,d+1}(\zeta,\tau-\Delta) \\
&\quad \times \exp\left[-\zeta r_{\text{video}}\left(\begin{array}{c}\beta_{\text{video}}(s,v,d)\left(\Delta - O(\Delta)\right)\\ +\beta_{\text{video}}(s,v,d+1)\,O(\Delta)\end{array}\right)\right] \\
&+ d\beta_{\text{data}}(s,v,d)\,\mu_{\text{data}}\Delta X_{s,v,d-1}(\zeta,\tau-\Delta) \\
&\quad \times \exp\left[-\zeta r_{\text{video}}\left(\begin{array}{c}\beta_{\text{video}}(s,v,d)\left(\Delta - O(\Delta)\right)\\ +\beta_{\text{video}}(s,v,d-1)\,O(\Delta)\end{array}\right)\right] \\
&+ \left(\begin{array}{c}-\lambda_{\text{speech}}\Delta - s\mu_{\text{speech}}\Delta - \lambda_{\text{video}}\Delta - \\ (v-1)\,\mu_{\text{video}}\Delta - \lambda_{\text{data}}\Delta - d\beta_{\text{data}}(s,v,d)\mu_{\text{data}}\Delta\end{array}\right) \\
&\quad \times X_{s,v,d}(\zeta,\tau-\Delta)\exp\left[-\zeta r_{\text{video}}\beta_{\text{video}}(s,v,d)\,\Delta\right] \\
&+ \left(\begin{array}{c}1-\zeta r_{\text{video}}\beta_{\text{video}}(s,v,d)\,\Delta \\ +\sum_{j=2}^{\infty}\frac{(-\zeta r_{\text{video}}\beta_{\text{video}}(s,v,d)\Delta)^{j}}{j!}\end{array}\right) X_{s,v,d}(\zeta,\tau-\Delta) \\
&+ o(\Delta).
\end{aligned}$$

Rearranging terms, letting $\Delta \downarrow 0$ and writing the resulting system of differential equations in matrix notation yields expression (7). Initial condition (8) simply reflects the fact that the transfer volume $x_{s,v,d}(0)$ of a video call with a duration of zero seconds equals zero bits. In order to prove that the system of differential equations (7) with initial condition (8) has a *unique* solution, note that it is a system of the form $\frac{\partial}{\partial\tau}\mathbf{X}(\zeta,\tau) = A\,\mathbf{X}(\zeta,\tau) \equiv f(\mathbf{X}(\zeta,\tau))$ where f is a linear function with continuous partial derivatives with respect to the entries of its argument vector. The existence and uniqueness of a solution $\mathbf{X}(\zeta,\tau)$ for every initial vector, immediately follows from e.g. [4, Chapter 1, Section 8]. To conclude the proof, it is readily verified that the claimed solution (9) indeed satisfies the system of differential equations (7) with initial condition (8). □

Using closed-form expression (9) for the Laplace-Stieltjes transform of the distribution of $x_{s,v,d}(\tau)$, Theorem 1 follows as a corollary to Lemma 1, as proven below.

PROOF. The existence of a vector $\boldsymbol{\gamma}_{\text{video}}$ that satisfies (3) and its uniqueness up to a translation along the vector $\mathbf{1}$, are guaranteed by results in Markov decision theory. Interpreting $\boldsymbol{\gamma}_{\text{video}}$ as the vector of relative values in a Markov reward chain governed by the generator $\mathcal{Q}^{\star}_{\text{video}}$ and with immediate cost vector $\frac{1}{\eta}\left(r_{\text{video}}\mathcal{B}_{\text{video}}\mathbf{1} - \left(\boldsymbol{\pi}^{\star}_{\text{video}}r_{\text{video}}\mathcal{B}_{\text{video}}\mathbf{1}\right)\mathbf{1}\right)$ where η is the maximum rate of change in the Markov chain, and understanding that the long-term average costs are zero, $\frac{1}{\eta}\boldsymbol{\pi}^{\star}_{\text{video}}\left(r_{\text{video}}\mathcal{B}_{\text{video}}\mathbf{1} - \left(\boldsymbol{\pi}^{\star}_{\text{video}}r_{\text{video}}\mathcal{B}_{\text{video}}\mathbf{1}\right)\mathbf{1}\right) = 0$, e.g. [15, Theorem 3.1, page 167] can be directly applied after uniformization of the continuous-time Markov chain. Hence in (3) a single degree of freedom exists in choosing $\boldsymbol{\gamma}_{\text{video}}$, which is used to normalise $\boldsymbol{\gamma}_{\text{video}}$ as in (4).

The vector of conditional expected transfer volumes $\widehat{\mathbf{x}}(\tau)$ is then obtained by taking the derivative of $\mathbf{X}(\zeta,\tau)$ with respect to ζ, and subsequently setting $\zeta = 0$.

$$\begin{aligned}
\widehat{\mathbf{x}}(\tau) &= -\frac{\partial}{\partial\zeta}\mathbf{X}(\zeta,\tau)\Big|_{\zeta=0} \\
&= -\frac{\partial}{\partial\zeta}\sum_{k=0}^{\infty}\frac{\left(\left(\tau\mathcal{Q}^{\star}_{\text{video}}\right) + \left(-\zeta\,\tau r_{\text{video}}\mathcal{B}_{\text{video}}\right)\right)^{k}}{k!}\mathbf{1}\Big|_{\zeta=0} \\
&= -\left(\sum_{k=1}^{\infty}\sum_{i=0}^{k-1}\frac{\left(\tau\mathcal{Q}^{\star}_{\text{v}}\right)^{k-i-1}\left(-\tau r_{\text{v}}\mathcal{B}_{\text{v}}\right)\left(\tau\mathcal{Q}^{\star}_{\text{v}}\right)^{i}}{k!}\right)\mathbf{1} \\
&= \left(\sum_{k=1}^{\infty}\frac{\left(\tau\mathcal{Q}^{\star}_{\text{video}}\right)^{k-1}}{k!}\right)\tau r_{\text{video}}\mathcal{B}_{\text{video}}\mathbf{1} \\
&= \tau\left(\boldsymbol{\pi}^{\star}_{\text{video}}\mathcal{B}_{\text{video}}\mathbf{1}\right)\mathbf{1} + \\
&\quad \left(\sum_{k=1}^{\infty}\frac{\left(\tau\mathcal{Q}^{\star}_{\text{video}}\right)^{k-1}}{k!}\right)\left[\begin{array}{c}\tau r_{\text{video}}\mathcal{B}_{\text{video}}\mathbf{1} - \\ \tau\left(\boldsymbol{\pi}^{\star}_{\text{video}}r_{\text{video}}\mathcal{B}_{\text{video}}\mathbf{1}\right)\mathbf{1}\end{array}\right] \\
&= \tau\left(\boldsymbol{\pi}^{\star}_{\text{video}}r_{\text{video}}\mathcal{B}_{\text{video}}\mathbf{1}\right)\mathbf{1} - \\
&\quad \left(\sum_{k=1}^{\infty}\frac{\left(\tau\mathcal{Q}^{\star}_{\text{video}}\right)^{k-1}}{k!}\right)\tau\mathcal{Q}^{\star}_{\text{video}}\boldsymbol{\gamma}_{\text{video}} \\
&= \tau\left(\boldsymbol{\pi}^{\star}_{\text{v}}r_{\text{v}}\mathcal{B}_{\text{v}}\mathbf{1}\right)\mathbf{1} + \left[\mathcal{I} - \exp\left\{\tau\mathcal{Q}^{\star}_{\text{v}}\right\}\right]\boldsymbol{\gamma}_{\text{v}}
\end{aligned}$$

where after the third equality sign only those matrix cross-products appear that remain after differentiating the terms in the preceding expression, and setting ζ to 0. The subsequent equality sign uses $\mathcal{Q}^{\star}_{\text{video}}\mathbf{1} = 0$, so that all terms with $i > 0$ disappear. A similar argument is used to obtain the fifth equality. Equation (3) is used for the sixth equality.

With regards to the asymptotic expressions, note that since $\mathcal{Q}^{\star}_{\text{video}}$ is the generator of an irreducible finite state space Markov chain, with equilibrium distribution vector $\boldsymbol{\pi}^{\star}_{\text{video}}$, it holds that $\lim_{\tau\to\infty}\exp\left\{\tau\mathcal{Q}^{\star}_{\text{video}}\right\} = \mathbf{1}\,\boldsymbol{\pi}^{\star}_{\text{video}}$, and thus $\lim_{\tau\to\infty}\left[\mathcal{I} - \exp\left\{\tau\mathcal{Q}^{\star}_{\text{video}}\right\}\right]\boldsymbol{\gamma}_{\text{video}} = \boldsymbol{\gamma}_{\text{video}}$, using (4), while $\lim_{\tau\to\infty}\tau^{-1}\left[\mathcal{I} - \exp\left\{\tau\mathcal{Q}^{\star}_{\text{video}}\right\}\right]\boldsymbol{\gamma}_{\text{video}} = \mathbf{0}$. □

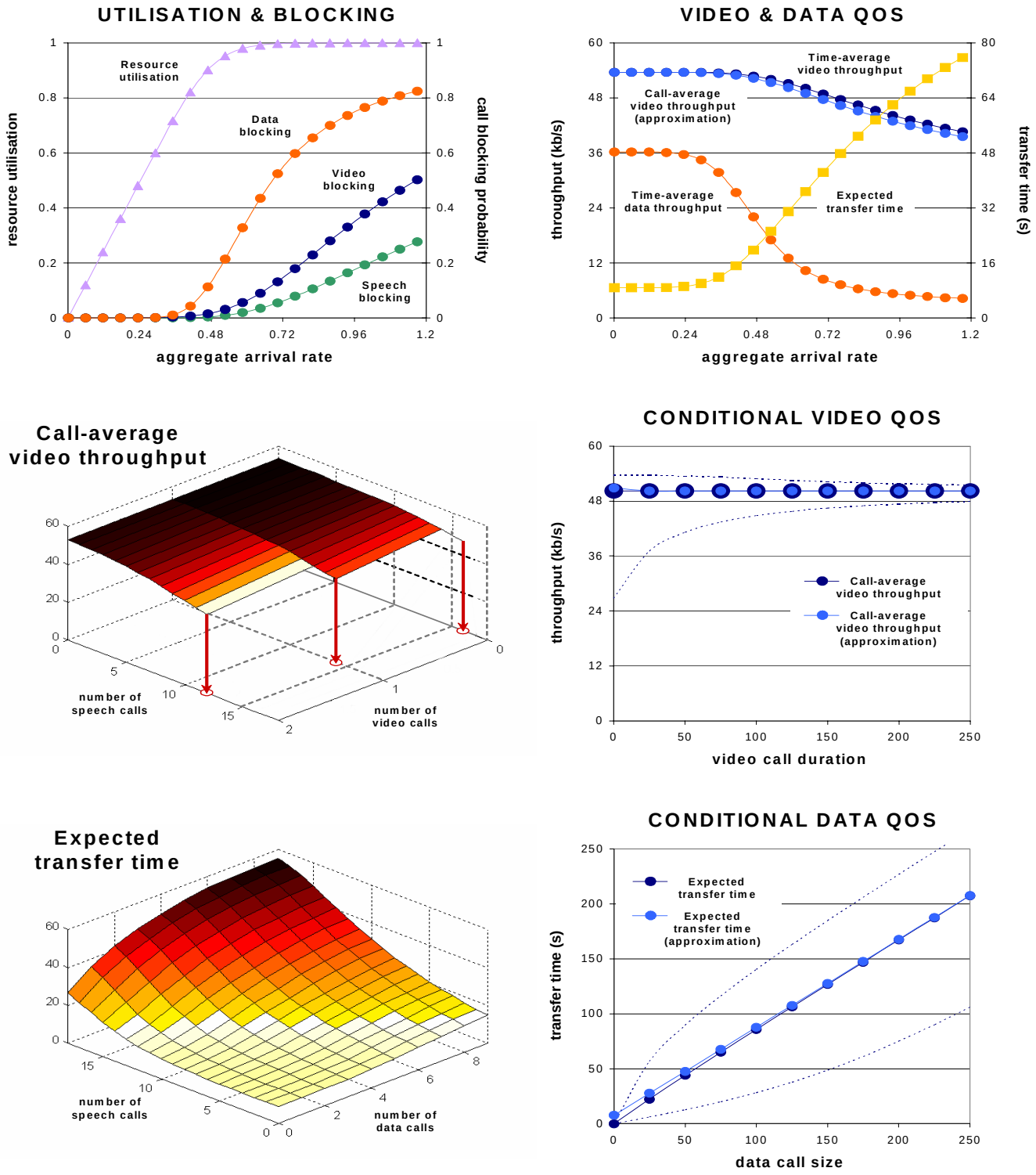


Figure 2: Results of some illustrative numerical experiments. For a given scenario, the upper two charts depict some basic performance measures as a function of the aggregate traffic load. The middle and lower pairs of charts concentrate on some conditional video and data QOS measures, respectively.

A NUMERICAL ANALYSIS OF THE DELAY IN A TANDEM QUEUE WITH BLOCKING AND INTERFERENCE

M.E. Martos
Department of Statistics and Operations Research I
Faculty of Mathematics
Complutense University of Madrid
28040, Madrid, Spain
Email: manuel-escribano@mat.ucm.es

KEYWORDS

Blocking, Markovian arrival process, Phase type distribution, Tandem queue.

ABSTRACT

This paper examines a two-stage tandem queue with blocking and interference in the access to servers. The input of units is modelled by a Markovian arrival process (MAP) and the service requirements in Stations 1 and 2 are of phase type (PH). Whereas an unlimited queue is allowed to accumulate in front of Station 1, a finite queue of capacity $K \geq 0$ is allowed between servers. This queue can be studied as a continuous-time Markov chain of *GI/M/1* type. The interest is in studying numerically the delay suffered by an arriving unit. To that end, we compute the sojourn time of an arriving unit through times until absorption in suitably defined continuous-time Markov chains.

INTRODUCTION

A particular class of queueing problems in modelling of data communications and production lines is characterized by a blocking and interference production environment with finite intermediate storage allowed between servers. A blocking production environment typically lies in the processing technology itself and/or in the absence of storage capacity between operations of a job. For instance, in the serial manufacturing scheme, blocking is forcing a departure from a machine to stop temporarily, so that the server at the exit machine must stop processing jobs that wait in its queue until the blocked job can enter into its downstream machine. Consequently, this blocking phenomenon results in an effective service rate for the exit machine that is lower than its actual service rate. Interference is a characteristic of some systems such as the gas pump model of A.B. Clarke (Clarke 1977) and phenomena observed in cafeteria lines and certain manufacturing systems (see (Neuts 1994)). The current paper deals with the A.B. Clarke model under a more general point of view. The main interest in this model and its applications can be consulted in (Lillo and Neuts 1999).

Consider a two-stage tandem queue with blocking with an infinite queue allowed before the first server and an intermediate buffer of finite capacity $K \geq 0$ allowed between servers. The arrival of units into the tandem queue is modelled by a MAP (see e.g. (Lucantoni et al. 1990)) corresponding to the decomposition $\mathbf{C} = \mathbf{C}_0 + \mathbf{C}_1$ of the $c \times c$ irreducible generator matrix $\mathbf{C}$ of the underlying phase process. Let $\lambda = \boldsymbol{\eta}\mathbf{C}_1\mathbf{e}_c$ be the rate of arrival point process, where $\boldsymbol{\eta}$ satisfies $\boldsymbol{\eta}\mathbf{C} = \mathbf{0}_c^T$ and $\boldsymbol{\eta}\mathbf{e}_c = 1$ (i.e. $\boldsymbol{\eta}$ is the stationary vector of $\mathbf{C}$). Here $\mathbf{e}_c$ is a column vector of size c with all components one, $\mathbf{0}_c$ is the null column vector of size c and ^{T} denotes transposition. Units are served in order of arrival, but they only receive one service from one of the servers. To be concrete, if one arriving unit finds both servers free, it proceeds directly to the second server. When there is a job in course in Station 2 and the first server is free, a new arriving unit starts service in Station 1. If the first server is busy, then this unit waits in the buffer of Station 1, so that it cannot bypass the first server to gain access to the second one. When the second server is busy, a unit whose service is completed in Station 1 cannot leave the system immediately. This interfered unit must join the buffer of Station 2 and a new unit, if there is one, can start service in Station 1. If the first server is busy, the second one is free and the service time in Station 1 finishes, then the completed unit passes out of the system. If, in such an epoch, units are waiting, the first unit starts service in Station 2 and the second one, if there is one, starts processing in Station 1.

The finiteness of the buffer in Station 2, which holds units with completed service in Station 1, results in blocking of the first server whenever a unit having completed its service in Station 1 cannot enter in the buffer of Station 2 due to $K + 1$ units are accommodated into it. Under a BAS (*Blocking After Service*) blocking mechanism, the first server cannot continue working. At the end of a job in Station 2, this served unit, all those that have already been served in Station 1 and the blocked unit (i.e. $K + 2$ units) leave the system.

Service times at Stations 1 and 2 are governed by PH laws (see e.g. (Neuts 1994)) with representations $(\boldsymbol{\alpha}, \mathbf{T})$ and $(\boldsymbol{\beta}, \mathbf{S})$, respectively. Here $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are row vectors of respective sizes t and s, and $\mathbf{T}$ and $\mathbf{S}$ are

Print 0-9553018-4-X, CD 0-9553018-3-1

square matrices with negative diagonal elements and non-negative off-diagonal elements, which are assumed to be stable. Let $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$ be the respective stationary vectors of the infinitesimal generators $\mathbf{T}+\mathbf{t}_0\boldsymbol{\alpha}$ and $\mathbf{S}+\mathbf{s}_0\boldsymbol{\beta}$, with $\mathbf{t}_0=-\mathbf{T}\mathbf{e}_t$ and $\mathbf{s}_0=-\mathbf{S}\mathbf{e}_s$. Note that $\boldsymbol{\omega}=(\boldsymbol{\alpha}\mathbf{T}^{-1}\mathbf{e}_t)^{-1}\boldsymbol{\alpha}\mathbf{T}^{-1}$ and $\boldsymbol{\theta}=(\boldsymbol{\beta}\mathbf{S}^{-1}\mathbf{e}_s)^{-1}\boldsymbol{\beta}\mathbf{S}^{-1}$, from it follows that $\mu=(\boldsymbol{\alpha}(-\mathbf{T}^{-1})\mathbf{e}_t)^{-1}$ and $\nu=(\boldsymbol{\beta}(-\mathbf{S}^{-1})\mathbf{e}_s)^{-1}$ correspond to the service rates at Station 1 and 2, respectively.
The streams of arriving units and the service times are assumed to be mutually independent.

The remainder of the paper is organized in three sections. In first section, we observe the underlying continuous-time Markov chain and we study the positive recurrence condition and the steady-state probability vector. The second section includes the analysis of the sojourn time of an arriving unit. After that, we illustrate the system behavior through some numerical examples.

UNDERLYING CONTINUOUS-TIME MARKOV CHAIN

We study the above queueing model as a continuous-time Markov chain defined by

$$\mathcal{X} = \{(\xi(u),\tau(u),\eta(u),\chi_1(u),\chi_2(u)) : u\geq 0\},$$

where $\xi(u)$ is the number of units (queued or in service) at Station 1, $\tau(u)$ denotes the total number of units (queued or in service) at Station 2 or blocked in the first server, and $\gamma(u)$ records the phase of the arrival process at time u. If the kth server is busy at time u, then $\chi_k(u)$ represents the phase of the service time, for $k\in\{1,2\}$. Indeed, $\mathcal{X}$ constitutes a special Markov chain of *GI/M/1* type on the state space

$$\mathcal{S} = \bigcup_{n=0}^{\infty} l(n),$$

where the levels $l(n)$, for $n\geq 0$, are defined as

$$l(n) = \bigcup_{m=0}^{K+2} \mathcal{L}(n,m), \quad n\geq 0.$$

The sub-levels $\mathcal{L}(n,m)$ are defined by sets of states as $\mathcal{L}(0,0)=\{(0,0,i) : 1\leq i\leq c\}$, $\mathcal{L}(0,m)=\{(0,m,i,k) : 1\leq i\leq c, 1\leq k\leq s\}$, for $1\leq m\leq K+2$, $\mathcal{L}(n,0)=\{(n,0,i,j) : 1\leq i\leq c, 1\leq j\leq t\}$, for $n\geq 1$, $\mathcal{L}(n,m)=\{(n,m,i,j,k) : 1\leq i\leq c, 1\leq j\leq t, 1\leq k\leq s\}$, for $1\leq m\leq K+1$ and $n\geq 1$, and $\mathcal{L}(n,K+2)=\{(n,K+2,i,k) : 1\leq i\leq c, 1\leq k\leq s\}$, for $n\geq 1$. Thus, the levels $l(0)$ and $l(n)$, for $n\geq 1$, consist of $J_0=c(1+(K+2)s)$ and $J_1=c(s+t(1+(K+1)s))$ states, respectively.

Let $\mathbf{I}_c$ be the identity matrix of dimension c and $\otimes$ and $\oplus$ the Kronecker product and sum, respectively. By listing states of $\mathcal{S}$ in lexicographic order we summarize transitions among sub-levels as follows:

- The matrix $\mathbf{C}_0$ records transition rates to states in $\mathcal{L}(0,0)$ from states of $\mathcal{L}(0,0)$.
- The matrix $\mathbf{C}_1\otimes\boldsymbol{\beta}$ records transition rates to states in $\mathcal{L}(0,1)$ from states of $\mathcal{L}(0,0)$.
- The matrix $\mathbf{C}_1\otimes\boldsymbol{\alpha}\otimes\mathbf{I}_s$ records transition rates to states in $\mathcal{L}(1,m)$ from states of $\mathcal{L}(0,m)$, for $1\leq m\leq K+1$.
- The matrix $\mathbf{I}_c\otimes\mathbf{s}_0$ records transition rates to states in $\mathcal{L}(0,0)$ from states of $\mathcal{L}(0,m)$, for $1\leq m\leq K+2$.
- The matrix $\mathbf{C}_0\oplus\mathbf{S}$ records transition rates to states in $\mathcal{L}(0,m)$ from states of $\mathcal{L}(0,m)$, for $1\leq m\leq K+1$ and to states in $\mathcal{L}(n,K+2)$ from states of $\mathcal{L}(n,K+2)$, for $n\geq 0$.
- The matrix $\mathbf{I}_c\otimes\mathbf{t}_0$ records transition rates to states in $\mathcal{L}(0,0)$ from states of $\mathcal{L}(1,0)$.
- The matrix $\mathbf{I}_c\otimes\mathbf{t}_0\otimes\mathbf{I}_s$ records transition rates to states in $\mathcal{L}(0,m+1)$ from states of $\mathcal{L}(1,m)$, for $1\leq m\leq K+1$, and to states in $\mathcal{L}(n-1,K+2)$ from states of $\mathcal{L}(n,K+1)$, for $n\geq 2$.
- The matrix $\mathbf{I}_c\otimes(\mathbf{s}_0\boldsymbol{\beta})$ records transition rates to states in $\mathcal{L}(0,1)$ from states of $\mathcal{L}(1,K+2)$.
- The matrix $\mathbf{I}_c\otimes\mathbf{t}_0\otimes\boldsymbol{\beta}$ records transition rates to states in $\mathcal{L}(0,1)$ from states of $\mathcal{L}(2,0)$.
- The matrix $\mathbf{C}_1\otimes\mathbf{I}_t$ records transition rates to states in $\mathcal{L}(n+1,0)$ from states of $\mathcal{L}(n,0)$, for $n\geq 1$.
- The matrix $\mathbf{I}_c\otimes(\mathbf{t}_0\boldsymbol{\alpha})\otimes\boldsymbol{\beta}$ records transition rates to states in $\mathcal{L}(n-2,1)$ from states of $\mathcal{L}(n,0)$, for $n\geq 3$.
- The matrix $\mathbf{C}\oplus\mathbf{T}$ records transition rates to states in $\mathcal{L}(n,0)$ from states of $\mathcal{L}(n,0)$, for $n\geq 1$.
- The matrix $\mathbf{C}_1\otimes\mathbf{I}_{ts}$ records transition rates to states in $\mathcal{L}(n+1,m)$ from states of $\mathcal{L}(n,m)$, for $n\geq 1$ and $1\leq m\leq K+1$.
- The matrix $\mathbf{I}_c\otimes(\mathbf{t}_0\boldsymbol{\alpha})\otimes\mathbf{I}_s$ records transition rates to states in $\mathcal{L}(n-1,m+1)$ from states of $\mathcal{L}(n,m)$, for $n\geq 2$ and $1\leq m\leq K$.
- The matrix $\mathbf{I}_{ct}\otimes\mathbf{s}_0$ records transition rates to states in $\mathcal{L}(n,0)$ from states of $\mathcal{L}(n,m)$, for $n\geq 1$ and $1\leq m\leq K+1$.
- The matrix $\mathbf{C}_0\oplus\mathbf{T}\oplus\mathbf{S}$ records transition rates to states in $\mathcal{L}(n,m)$ from states of $\mathcal{L}(n,m)$, for $n\geq 1$ and $1\leq m\leq K+1$.
- The matrix $\mathbf{C}_1\otimes\mathbf{I}_s$ records transition rates to states in $\mathcal{L}(n+1,K+2)$ from states of $\mathcal{L}(n,K+2)$, for $n\geq 0$.
- The rate matrix $\mathbf{I}_c\otimes\boldsymbol{\alpha}\otimes(\mathbf{s}_0\boldsymbol{\beta})$ records transition rates to states in $\mathcal{L}(n-1,1)$ from states of $\mathcal{L}(n,K+2)$, for $n\geq 2$.

The infinitesimal generator of $\mathcal{X}$ is given by the block-structured matrix

$$\mathbf{Q} = \begin{pmatrix} \mathbf{B}_1 & \mathbf{B}_0 & & & & & \\ \mathbf{B}_2 & \mathbf{A}_1 & \mathbf{A}_0 & & & & \\ \mathbf{B}_3 & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & & & \\ & \mathbf{A}_3 & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & & \\ & & \mathbf{A}_3 & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \\ & & & \ddots & \ddots & \ddots & \ddots \end{pmatrix},$$

where $\mathbf{A}_k$, for $k\in\{0,1,2,3\}$, and $\mathbf{B}_1$ are square matrices of sizes J_1 and J_0, respectively. The block matrices $\mathbf{B}_0$ and $\mathbf{B}_k$, for $k\in\{2,3\}$, are of dimensions $J_0\times J_1$ and $J_1\times J_0$, respectively. Expressions for these blocks

are readily derived from the above-mentioned transitions among sub-levels and are thus omitted.

Let $\boldsymbol{\pi} = (\boldsymbol{\pi}(0), \boldsymbol{\pi}(1), \ldots, \boldsymbol{\pi}(K+2))$ be the stationary vector of $\mathbf{A} = \mathbf{A}_0 + \mathbf{A}_1 + \mathbf{A}_2 + \mathbf{A}_3$, where $\boldsymbol{\pi}(0)$ is a row vector of size ct, $\boldsymbol{\pi}(m)$ is a row vector of size cts, for $1 \leq m \leq K+1$, and $\boldsymbol{\pi}(K+2)$ is a row vector of size cs. From the general theory of *GI/M/1* type Markov chains it is found that $\mathcal{X}$ is positive recurrent, null recurrent or transient if and only if $\rho < 1$, $\rho = 1$ or $\rho > 1$, respectively, where the traffic load is defined as

$$\rho \quad = \quad \lambda \left(\mu(1 - \boldsymbol{\pi}(K+2)\mathbf{e}_{cs}) + \nu(1 - \boldsymbol{\pi}(0)\mathbf{e}_{ct})\right)^{-1}.$$

Let $\mathbf{x}$ be the steady-state probability vector of the infinitesimal generator $\mathbf{Q}$ in the positive recurrent case. We partition $\mathbf{x}$ into sub-vectors $\mathbf{x} = (\mathbf{x}(0), \mathbf{x}(1), \ldots)$, where $\mathbf{x}(0)$ has J_0 entries and $\mathbf{x}(n)$, for $n \geq 1$, has J_1 entries. The vector $\mathbf{x}$ can be computed from the matrix geometric form given by the equations

$$\begin{aligned} \mathbf{x}(0) &= \mathbf{x}(1)\left(\mathbf{B}_2 + \mathbf{R}\mathbf{B}_3\right)\left(-\mathbf{B}_1^{-1}\right), \\ \mathbf{x}(n) &= \mathbf{x}(1)\mathbf{R}^{n-1}, \quad n \geq 1. \end{aligned}$$

The rate matrix $\mathbf{R}$ is the minimal solution to the matrix equation

$$\mathbf{A}_0 + \mathbf{R}\mathbf{A}_1 + \mathbf{R}^2\mathbf{A}_2 + \mathbf{R}^3\mathbf{A}_3 \quad = \quad \mathbf{0}_{J_1 \times J_1}, \qquad (1)$$

in the set of non-negative matrices, where $\mathbf{0}_{J_1 \times J_1}$ is the null matrix of size $J_1 \times J_1$.
From (1) we find that

$$\mathbf{R} \quad = \quad \left(\mathbf{A}_0 + \mathbf{R}^2\mathbf{A}_2 + \mathbf{R}^3\mathbf{A}_3\right)\left(-\mathbf{A}_1^{-1}\right).$$

Then we can can start somewhat arbitrarily with $\mathbf{R}(0) = \mathbf{0}_{J_1 \times J_1}$ and iteratively evaluate the sequence $\{\mathbf{R}(k) : k \geq 1\}$ as follows:

$$\mathbf{R}(k+1) \quad = \quad \left(\mathbf{A}_0 + \mathbf{R}^2(k)\mathbf{A}_2 + \mathbf{R}^3(k)\mathbf{A}_3\right)\left(-\mathbf{A}_1^{-1}\right),$$

for $k \geq 0$. To define a stopping criterion, the standard practice is to keep track of two successive matrices in each iteration and stop iterations when $||\mathbf{R}(k) - \mathbf{R}(k-1)||_\infty < \epsilon$, for a small value $\epsilon > 0$. Algorithm 1 includes the necessary steps to compute the rate matrix $\mathbf{R}$.

Algorithm 1: Algorithm to compute the matrix $\mathbf{R}$

Step 1. $\mathbf{H}_1 := \left(-\mathbf{A}_1^{-1}\right)$;
$\mathbf{H}_0 := \mathbf{A}_0\mathbf{H}_1$;
$\mathbf{H}_2 := \mathbf{A}_2\mathbf{H}_1$;
$\mathbf{H}_3 := \mathbf{A}_3\mathbf{H}_1$;
$\mathbf{Z} := \mathbf{H}_0$.
Step 2. Repeat until $||\mathbf{W} - \mathbf{Z}||_\infty \leq \epsilon$
$\mathbf{W} := \mathbf{Z}$;
$\mathbf{Z} := \mathbf{W}^2$;
$\mathbf{Z} := \mathbf{H}_0 + \mathbf{Z}\left(\mathbf{H}_2 + \mathbf{W}\mathbf{H}_3\right)$.
Step 3. $\mathbf{R} := \mathbf{Z}$.

Neuts (Neuts 1994) has shown that the sequence $\{\mathbf{R}(k) : k \geq 1\}$ is a monotonically increasing sequence that converges to the minimal non-negative solution to $\sum_{k=0}^{3} \mathbf{R}^k\mathbf{A}_k = \mathbf{0}_{J_1 \times J_1}$, and that this solution is the solution that uniquely provides the matrix $\mathbf{R}$ that satisfies such an equation.
Once the matrix $\mathbf{R}$ is numerically computed we can evaluate the sub-vector $\mathbf{x}(1)$ from the equations

$$\begin{aligned} \mathbf{0}_{J_1}^T =& \mathbf{x}(1)\Big(\mathbf{A}_1 + \mathbf{R}\mathbf{A}_2 + \mathbf{R}^2\mathbf{A}_3 \\ &+ \left(\mathbf{B}_2 + \mathbf{R}\mathbf{B}_3\right)\left(-\mathbf{B}_1^{-1}\right)\mathbf{B}_0\Big), \\ 1 =& \mathbf{x}(1)\Big(\left(\mathbf{B}_2 + \mathbf{R}\mathbf{B}_3\right)\left(-\mathbf{B}_1^{-1}\right)\mathbf{e}_{J_0} + (\mathbf{I}_{J_1} - \mathbf{R})^{-1}\mathbf{e}_{J_1}\Big). \end{aligned}$$

After computing the rate matrix $\mathbf{R}$, and the sub-vector $\mathbf{x}(1)$, we can compute some probabilistic descriptors, such as the steady-state blocking probability P_b. Indeed,

$$\begin{aligned} P_b &= \lim_{u \to \infty} P\left(\tau(u) = K+2\right) \\ &= \mathbf{x}(1)\Big(\left(\mathbf{B}_2 + \mathbf{R}\mathbf{B}_3\right)\left(-\mathbf{B}_1^{-1}\right)\boldsymbol{\gamma} + (\mathbf{I}_{J_1} - \mathbf{R})^{-1}\tilde{\boldsymbol{\gamma}}\Big), \end{aligned}$$

where $\boldsymbol{\gamma}$ and $\tilde{\boldsymbol{\gamma}}$ are column vectors given by

$$\boldsymbol{\gamma} = \begin{pmatrix} \mathbf{0}_{(1+(K+1)s)c} \\ \mathbf{e}_{cs} \end{pmatrix} \quad \text{and} \quad \tilde{\boldsymbol{\gamma}} = \begin{pmatrix} \mathbf{0}_{(1+(K+1)s)ct} \\ \mathbf{e}_{cs} \end{pmatrix}.$$

Next section studies the sojourn time of an arriving unit through times until absorption in continuous-time Markov chains following the methodology of section 3 in (Gómez-Corral 2004). We only present results, for details see (Gómez-Corral and Martos 2007).

SOJOURN TIME OF AN ARRIVING UNIT

Let ζ_W be the length of the waiting time in Station 1, ζ_k the service time at Station k, for $k \in \{1, 2\}$, ζ_B the blocking delay suffered by the arriving unit in the first server and ζ_D the delay suffered in the buffer of Station 2. Our aim here is to study the Laplace-Stieltjes transform $\phi(\theta_0, \theta_1, \theta_2, \theta_3, \theta_4)$ given by

$$E\left[\exp\left\{-(\theta_0\zeta_W + \theta_1\zeta_1 + \theta_2\zeta_B + \theta_3\zeta_D + \theta_4\zeta_2)\right\}\right],$$

for $Re(\theta_k) \geq 0$ and $k \in \{0, 1, 2, 3, 4\}$.

We compute $\phi(\theta_0, \theta_1, \theta_2, \theta_3, \theta_4)$ as the product of the Laplace-Stieltjes transform of ζ_W and the Laplace-Stieltjes transform of the remaining time until the exit of the system according to the state of the network. To that end, we first study the state of the tandem queue, just after the epoch ζ_W, concerning only with those units found by the arriving unit at its arrival epoch.
In order to define $\phi(\theta_0, \theta_1, \theta_2, \theta_3, \theta_4)$ in the simplest possible fashion, we set the matrices

$$\begin{aligned} \boldsymbol{\Lambda} &= \begin{pmatrix} \mathbf{C}_1\mathbf{e}_c & & \\ & \mathbf{I}_{K+1} \otimes (\mathbf{C}_1\mathbf{e}_c) \otimes \mathbf{I}_s & \\ \mathbf{0}_{cs} & & \end{pmatrix}, \\ \mathbf{W}_0 &= \lambda^{-1}(\mathbf{B}_2 + \mathbf{R}\mathbf{B}_3)\left(-\mathbf{B}_1^{-1}\right)\boldsymbol{\Lambda}, \\ \mathbf{V}_0 &= \begin{pmatrix} \mathbf{0}_{c(1+(K+1)s) \times s} \\ (\mathbf{C}_1\mathbf{e}_c) \otimes \mathbf{I}_s \end{pmatrix}, \\ \mathbf{W}_1 &= \left(\lambda^{-1}(\mathbf{B}_2 + \mathbf{R}\mathbf{B}_3)\left(-\mathbf{B}_1^{-1}\right)\mathbf{V}_0, \mathbf{W}_2\right), \\ \mathbf{W}_2 &= \lambda^{-1}(\mathbf{V}_1, \mathbf{R}\mathbf{V}_1), \end{aligned}$$

with $\mathbf{V}_1 = \mathrm{diag}((\mathbf{C}_1\mathbf{e}_c)\otimes\mathbf{I}_t, \mathbf{I}_{K+1}\otimes(\mathbf{C}_1\mathbf{e}_c)\otimes\mathbf{I}_{ts}, (\mathbf{C}_1\mathbf{e}_c)\otimes\mathbf{I}_s)$. Besides, we consider the block-matrices

$$\bar{\mathbf{A}}_1 = \begin{pmatrix} \tilde{\mathbf{A}}_1 & \\ \tilde{\mathbf{A}}_2 & \tilde{\mathbf{A}}_1 \end{pmatrix}, \quad \bar{\mathbf{A}}_2 = \begin{pmatrix} \tilde{\mathbf{A}}_3 & \tilde{\mathbf{A}}_2 \\ & \tilde{\mathbf{A}}_3 \end{pmatrix},$$
$$\bar{\mathbf{B}}_1 = \begin{pmatrix} \tilde{\mathbf{B}}_1 & \\ \tilde{\mathbf{C}}_2 & \tilde{\mathbf{A}}_1 \end{pmatrix}, \quad \bar{\mathbf{B}}_2 = \begin{pmatrix} \tilde{\mathbf{B}}_2 \\ \tilde{\mathbf{B}}_3 \end{pmatrix},$$
$$\bar{\mathbf{C}}_2 = \begin{pmatrix} \tilde{\mathbf{C}}_3 & \tilde{\mathbf{A}}_2 \\ & \tilde{\mathbf{A}}_3 \end{pmatrix},$$

where $\tilde{\mathbf{A}}_k$, $\tilde{\mathbf{B}}_k$, and $\tilde{\mathbf{C}}_k$ are given by

$$\tilde{\mathbf{A}}_1 = \begin{pmatrix} \mathbf{T} & & \\ \mathbf{e}_{K+1}\otimes\mathbf{I}_t\otimes\mathbf{s}_0 & \mathbf{I}_{K+1}\otimes(\mathbf{T}\oplus\mathbf{S}) & \\ & & \mathbf{S} \end{pmatrix},$$

$$\tilde{\mathbf{A}}_2 = \begin{pmatrix} \mathbf{0}_{t\times t} & & \\ & \mathbf{I}_K\otimes(\mathbf{t}_0\boldsymbol{\alpha})\otimes\mathbf{I}_s & \\ & & \mathbf{t}_0\otimes\mathbf{I}_s \\ \boldsymbol{\alpha}\otimes(\mathbf{s}_0\boldsymbol{\beta}) & & \mathbf{0}_{s\times s} \end{pmatrix},$$

$$\tilde{\mathbf{A}}_3 = \begin{pmatrix} \mathbf{0}_{t\times t} & (\mathbf{t}_0\boldsymbol{\alpha})\otimes\boldsymbol{\beta} & \mathbf{0}_{t\times(1+Kt)s} \\ & \mathbf{0}_{(1+(K+1)t)s\times ts} & \end{pmatrix},$$

$$\tilde{\mathbf{B}}_1 = \begin{pmatrix} \mathbf{S} & & & \\ & \mathbf{T} & & \\ & \mathbf{e}_K\otimes\mathbf{I}_t\otimes\mathbf{s}_0 & \mathbf{I}_K\otimes(\mathbf{T}\oplus\mathbf{S}) & \\ \mathbf{t}_0\otimes\mathbf{I}_s & \mathbf{I}_t\otimes\mathbf{s}_0 & & \mathbf{T}\oplus\mathbf{S} \\ & & & \mathbf{S} \end{pmatrix},$$

$$\tilde{\mathbf{B}}_2 = \begin{pmatrix} \mathbf{s}_0 & & \\ \mathbf{t}_0 & & \\ & \mathbf{I}_K\otimes\mathbf{t}_0\otimes\mathbf{I}_s & \\ & \mathbf{0}_{ts\times Ks} & \\ \mathbf{s}_0\boldsymbol{\beta} & & \end{pmatrix},$$

$$\tilde{\mathbf{B}}_3 = \begin{pmatrix} \mathbf{0}_t & \mathbf{t}_0\otimes\boldsymbol{\beta} & \mathbf{0}_{t\times Ks} \\ & \mathbf{0}_{(1+(K+1)t)s\times s} & \end{pmatrix},$$

$$\tilde{\mathbf{C}}_2 = \begin{pmatrix} \mathbf{0}_{t\times(t+s)} & & \\ & \mathbf{I}_K\otimes(\mathbf{t}_0\boldsymbol{\alpha})\otimes\mathbf{I}_s & \\ & & \mathbf{t}_0\otimes\mathbf{I}_s \\ \boldsymbol{\alpha}\otimes(\mathbf{s}_0\boldsymbol{\beta}) & & \mathbf{0}_{s\times s} \end{pmatrix},$$

$$\tilde{\mathbf{C}}_3 = \begin{pmatrix} \mathbf{0}_{t\times(t+s)} & (\mathbf{t}_0\boldsymbol{\alpha})\otimes\boldsymbol{\beta} & \mathbf{0}_{t\times(1+Kt)s} \\ & \mathbf{0}_{(1+(K+1)t)s\times ts} & \end{pmatrix}.$$

Note that these blocks constitute an absorbing QBD (*Quasi-Birth-and-Death*) process. Moreover, we correspond the distribution until absorption in this QBD to the distribution of ζ_W.

The joint Laplace-Stieltjes transform $\boldsymbol{\phi}(\theta_0,\theta_1,\theta_2,\theta_3,\theta_4)$ of the random vector $(\zeta_W,\zeta_1,\zeta_B,\zeta_D,\zeta_2)$ is given by

$$\boldsymbol{\Phi}(\theta_0)\begin{pmatrix} \boldsymbol{\beta}\boldsymbol{\Omega}_0(\theta_4) \\ \mathbf{e}_K\otimes((\boldsymbol{\alpha}\otimes\mathbf{I}_s)\boldsymbol{\Omega}_1(\theta_1,\theta_3)) \\ (\boldsymbol{\alpha}\otimes\mathbf{I}_s)\boldsymbol{\Omega}_1(\theta_1,\theta_2) \end{pmatrix}, \qquad (2)$$

for $Re(\theta_k)\geq 0$ and $k\in\{0,1,2,3,4\}$, where

$$\begin{aligned}\boldsymbol{\Phi}(\theta_0) &= \mathbf{x}(1)\Big(\mathbf{W}_0+\mathbf{W}_1\left(\theta_0\mathbf{I}_{\bar{J}_1}-\bar{\mathbf{B}}_1\right)^{-1}\bar{\mathbf{B}}_2 \\ &\quad +\mathbf{R}^2\boldsymbol{\Psi}(\theta_0)\left(\theta_0\mathbf{I}_{\bar{J}_2}-\bar{\mathbf{A}}_1\right)^{-1}\bar{\mathbf{C}}_2 \\ &\quad \times\left(\theta_0\mathbf{I}_{\bar{J}_1}-\bar{\mathbf{B}}_1\right)^{-1}\bar{\mathbf{B}}_2\Big), \quad Re(\theta_k)\geq 0,\end{aligned}$$

with $\bar{J}_0 = 1+(K+1)s$ and $\bar{J}_1 = s+2(s+t(1+(K+1)s))$ is the row vector of size $\bar{J}_0$ of Laplace-Stieltjes transform of the absorbing time in a continuous-time Markov chain defined by the block matrices $\bar{\mathbf{A}}_k$, $\bar{\mathbf{B}}_k$ and $\bar{\mathbf{C}}_k$ and $\boldsymbol{\Omega}_0(\theta_4) = (\theta_4\mathbf{I}_s-\mathbf{S})^{-1}\mathbf{s}_0$ and $\boldsymbol{\Omega}_1(\theta_1,\theta_2) = (\theta_1\mathbf{I}_{ts}-\mathbf{T}\oplus\mathbf{S})^{-1}(\mathbf{t}_0\otimes((\theta_2\mathbf{I}_s-\mathbf{S})^{-1}\mathbf{s}_0)+((\theta_1\mathbf{I}_t-\mathbf{T})^{-1}\mathbf{t}_0)\otimes\mathbf{s}_0)$. The matrix $\boldsymbol{\Psi}(\theta_0)$ satisfies the equality

$$\boldsymbol{\Psi}(\theta_0) = \mathbf{W}_2+\mathbf{R}^2\boldsymbol{\Psi}(\theta_0)\left(\left(\theta_0\mathbf{I}_{\bar{J}_2}-\bar{\mathbf{A}}_1\right)^{-1}\bar{\mathbf{A}}_2\right),$$

for $Re(\theta_0)\geq 0$ and $\bar{J}_2 = 2(s+t(1+(K+1)s))$. Therefore, $\boldsymbol{\Psi}(\theta_0)$ can be expressed in terms of the direct sum of its rows. Towards this end, we introduce the row vector function $\psi(\boldsymbol{\Psi}(\theta_0)) = (\boldsymbol{\Psi}(\theta_0|1),\boldsymbol{\Psi}(\theta_0|2),\ldots,\boldsymbol{\Psi}(\theta_0|J_1))$, where $\boldsymbol{\Psi}(\theta_0|m)$ is the mth row vector of $\boldsymbol{\Psi}(\theta_0)$. By using classical properties of Kronecker's products (see Section 3.9 in (Neuts 1994), we find that the direct sum $\psi(\boldsymbol{\Psi}(\theta_0))$ is explicitly given by

$$\begin{aligned}\psi(\boldsymbol{\Psi}(\theta_0)) &= \psi(\mathbf{W}_2)+\psi(\mathbf{W}_2)\Big(\mathbf{I}_{J_1}\otimes\left(\theta_0\mathbf{I}_{\bar{J}_2}-\bar{\mathbf{A}}_1\right) \\ &\quad -\left(\mathbf{R}^2\right)^T\otimes\bar{\mathbf{A}}_2\Big)^{-1}\Big(\left(\mathbf{R}^2\right)^T\otimes\bar{\mathbf{A}}_2\Big),\end{aligned}$$

for $Re(\theta_0)\geq 0$. The proof of the existence of $(\mathbf{I}_{J_1}\otimes(\theta_0\mathbf{I}_{\bar{J}_2}-\bar{\mathbf{A}}_1)-(\mathbf{R}^2)^T\otimes\bar{\mathbf{A}}_2)^{-1}$ mostly repeat arguments of Corollary 3.9.1 in (Neuts 1994).

If we partition the row vector $\boldsymbol{\Phi}(\theta_0)$ of Laplace-Stieltjes transforms as $(\Phi(\theta_0|0),\boldsymbol{\Phi}(\theta_0|1),\ldots,\boldsymbol{\Phi}(\theta_0|K+1))$, where $\Phi(\theta_0|0)$ is a scalar and $\boldsymbol{\Phi}(\theta_0|m)$, for $1\leq m\leq K+1$ is a vector of size s, then it is straightforward from (2) that the Laplace-Stieltjes transform of the blocking delay ζ_B has the form

$$E\left[e^{-\theta\zeta_B}\right] = 1-\boldsymbol{\Phi}(0|K+1)(\mathbf{e}_s-(\boldsymbol{\alpha}\otimes\mathbf{I}_s)\boldsymbol{\Omega}_1(0,\theta)),$$

for $Re(\theta)\geq 0$, and thus the blocking probability can be computed as

$$P(\zeta_B>0) = (\boldsymbol{\alpha}\otimes\boldsymbol{\Phi}(0|K+1))(-\mathbf{T}\oplus\mathbf{S})^{-1}(\mathbf{t}_0\otimes\mathbf{e}_s).$$

Similarly, the Laplace-Stieltjes transform of the delay ζ_D suffered in the buffer of Station 2 is given by

$$E\left[e^{-\theta\zeta_D}\right] = 1-\sum_{m=1}^{K}\boldsymbol{\Phi}(0|m)(\mathbf{e}_s-(\boldsymbol{\alpha}\otimes\mathbf{I}_s)\boldsymbol{\Omega}_1(0,\theta)),$$

for $Re(\theta)\geq 0$. Furthermore,

$$P(\zeta_D>0)=\left(\boldsymbol{\alpha}\otimes\sum_{m=1}^{K}\boldsymbol{\Phi}(0|m)\right)(-\mathbf{T}\oplus\mathbf{S})^{-1}(\mathbf{t}_0\otimes\mathbf{e}_s).$$

NUMERICAL EXAMPLES

To illustrate the system behavior we consider a MAP with arrival rate $\lambda = 1.0$ and matrices $\mathbf{C}_0$ and $\mathbf{C}_1$ given

Table 1: Stability abscissa ν^*

	$K=0$	$K=1$	$K=3$	$K=5$
$\mu=0.5\nu$	0.88888	0.87804	0.87518	0.87500
$\mu=\nu$	1.38460	1.48871	1.54328	1.55315
$\mu=2\nu$	1.75435	2.14702	2.50193	2.63317

Table 2: P_b versus ν^{-1}

ν^{-1}		$K=0$	$K=1$	$K=3$	$K=5$
0.1	$\mu=0.5\nu$	0.00258	0.00013	$<10^{-6}$	$<10^{-6}$
	$\mu=\nu$	0.00452	0.00035	$<10^{-6}$	$<10^{-6}$
	$\mu=2\nu$	0.00649	0.00061	$<10^{-6}$	$<10^{-6}$
0.2	$\mu=0.5\nu$	0.00883	0.00078	$<10^{-6}$	$<10^{-6}$
	$\mu=\nu$	0.01507	0.00196	0.00004	$<10^{-6}$
	$\mu=2\nu$	0.02146	0.00344	0.00010	$<10^{-6}$
0.3	$\mu=0.5\nu$	0.01773	0.00207	0.00004	$<10^{-6}$
	$\mu=\nu$	0.02947	0.00499	0.00019	$<10^{-6}$
	$\mu=2\nu$	0.04166	0.00868	0.00048	0.00002
0.4	$\mu=0.5\nu$	0.02887	0.00408	0.00011	$<10^{-6}$
	$\mu=\nu$	0.04676	0.00936	0.00052	0.00003
	$\mu=2\nu$	0.06562	0.01610	0.00128	0.00011
0.5	$\mu=0.5\nu$	0.04205	0.00682	0.00024	$<10^{-6}$
	$\mu=\nu$	0.06639	0.01497	0.00107	0.00008
	$\mu=2\nu$	0.09248	0.02546	0.00259	0.00029
0.6	$\mu=0.5\nu$	0.05714	0.01034	0.00044	0.00002
	$\mu=\nu$	0.08802	0.02170	0.00186	0.00018
	$\mu=2\nu$	0.12173	0.03650	0.00447	0.00062
0.7	$\mu=0.5\nu$	0.07410	0.01465	0.00073	0.00003
	$\mu=\nu$	0.11139	0.02948	0.00292	0.00033
	$\mu=2\nu$	0.15300	0.04903	0.00693	0.00112
0.8	$\mu=0.5\nu$	0.09288	0.01979	0.00111	0.00006
	$\mu=\nu$	0.13632	0.03822	0.00424	0.00054
	$\mu=2\nu$	0.18600	0.06288	0.00997	0.00183
1.0	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	0.19037	0.05835	0.00769	0.00115
	$\mu=2\nu$	0.25645	0.09402	0.01769	0.00390
1.25	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	0.26480	0.08795	0.01343	0.00231
	$\mu=2\nu$	0.35145	0.13838	0.03016	0.00777
1.5	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	——	——	0.02066	0.00390
	$\mu=2\nu$	0.45280	0.18770	0.04536	0.01302
1.75	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	——	——	——	——
	$\mu=2\nu$	0.55948	0.24116	0.06290	0.01952
2.0	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	——	——	——	——
	$\mu=2\nu$	——	0.29814	0.08246	0.02712
2.25	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	——	——	——	——
	$\mu=2\nu$	——	——	0.10375	0.03570
2.6	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	——	——	——	——
	$\mu=2\nu$	——	——	——	0.04910

by

$$\mathbf{C}_0 = \begin{pmatrix} -\frac{5}{6} & 0 & 0 \\ \frac{5}{9} & -\frac{5}{3} & 0 \\ \frac{5}{6} & \frac{5}{6} & -\frac{25}{6} \end{pmatrix},$$

$$\mathbf{C}_1 = \begin{pmatrix} \frac{5}{18} & \frac{5}{18} & \frac{5}{18} \\ \frac{5}{9} & \frac{5}{9} & 0 \\ \frac{5}{6} & \frac{5}{6} & \frac{5}{6} \end{pmatrix}.$$

We suppose that service times at Stations 1 and 2 have an Erlang$(2, \mu_0)$ and an exponential distribution of rate ν, respectively.

Table 3: $P(\zeta_B > 0)$ versus ν^{-1}

ν^{-1}		$K=0$	$K=1$	$K=3$	$K=5$
0.1	$\mu=0.5\nu$	0.02589	0.00134	$<10^{-6}$	$<10^{-6}$
	$\mu=\nu$	0.04528	0.00351	0.00002	$<10^{-6}$
	$\mu=2\nu$	0.06494	0.00618	0.00006	$<10^{-6}$
0.2	$\mu=0.5\nu$	0.04417	0.00390	0.00004	$<10^{-6}$
	$\mu=\nu$	0.07535	0.00982	0.00022	$<10^{-6}$
	$\mu=2\nu$	0.10733	0.01721	0.00054	0.00001
0.3	$\mu=0.5\nu$	0.05911	0.00692	0.00013	$<10^{-6}$
	$\mu=\nu$	0.09824	0.01665	0.00066	0.00003
	$\mu=2\nu$	0.13889	0.02893	0.00161	0.00009
0.4	$\mu=0.5\nu$	0.07218	0.01020	0.00028	$<10^{-6}$
	$\mu=\nu$	0.11690	0.02342	0.00131	0.00008
	$\mu=2\nu$	0.16405	0.04027	0.00320	0.00028
0.5	$\mu=0.5\nu$	0.08410	0.01365	0.00048	0.00001
	$\mu=\nu$	0.13279	0.02994	0.00215	0.00017
	$\mu=2\nu$	0.18497	0.05092	0.00518	0.00059
0.6	$\mu=0.5\nu$	0.09524	0.01724	0.00074	0.00003
	$\mu=\nu$	0.14670	0.03618	0.00311	0.00030
	$\mu=2\nu$	0.20289	0.06083	0.00745	0.00104
0.7	$\mu=0.5\nu$	0.10586	0.02094	0.00104	0.00005
	$\mu=\nu$	0.15913	0.04212	0.00417	0.00047
	$\mu=2\nu$	0.21857	0.07004	0.00991	0.00161
0.8	$\mu=0.5\nu$	0.11611	0.02474	0.00139	0.00008
	$\mu=\nu$	0.17041	0.04778	0.00530	0.00067
	$\mu=2\nu$	0.23250	0.07860	0.01246	0.00229
1.0	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	0.19037	0.05835	0.00769	0.00115
	$\mu=2\nu$	0.25645	0.09402	0.01769	0.00390
1.25	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	0.21184	0.07036	0.01074	0.00185
	$\mu=2\nu$	0.28116	0.11071	0.02413	0.00622
1.5	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	——	——	0.01377	0.00260
	$\mu=2\nu$	0.30186	0.12513	0.03024	0.00868
1.75	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	——	——	——	——
	$\mu=2\nu$	0.31970	0.13780	0.03594	0.01115
2.0	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	——	——	——	——
	$\mu=2\nu$	——	0.14907	0.04123	0.01356
2.25	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	——	——	——	——
	$\mu=2\nu$	——	——	0.04611	0.01586
2.6	$\mu=0.5\nu$	——	——	——	——
	$\mu=\nu$	——	——	——	——
	$\mu=2\nu$	——	——	——	0.01888

The interest in this paper is in studying the delay in the system. To that end, we illustrate the effect of the mean service time ν^{-1} in Station 2 on the probabilities P_b, $P(\zeta_B > 0)$, when the buffer capacity $K \in \{0, 1, 3, 5\}$ and the service rate in Station 1 (given by $\mu = \mu_0/2$) is chosen as a function of ν; i.e., $\mu = 0.5\nu$, ν and 2ν.
In Table 1, we list the stability abscissa values ν^* such the positive recurrence condition $\rho < 1$ is equivalent to $\nu^{-1} < \nu^*$.

Table 2 shows the steady-state blocking probability. When we fix the pair (ν^{-1}, μ), P_b is a decreasing function of K. Nevertheless, if we fix the pairs (ν^{-1}, K) and (μ, K), P_b is an increasing function of μ and ν^{-1}, respectively. Table 3 let us deduce a similar behavior on $P(\zeta_B > 0)$. Comparing both tables we observe that $P_b \leq P(\zeta_B > 0)$ when $\nu^{-1} \leq 1.0$ and $P_b > P(\zeta_B > 0)$ when $\nu^{-1} > 1.0$, irrespective of K, μ and ν. Moreover, it seems that $P_b \simeq \nu^{-1} P(\zeta_B > 0)$ when $\nu^{-1} \leq 1.0$. Because we fix $\lambda = 1.0$, we can extend the above conclusion and set $P_b \leq P(\zeta_B > 0)$ when $\nu^{-1} \leq \lambda^{-1}$ and

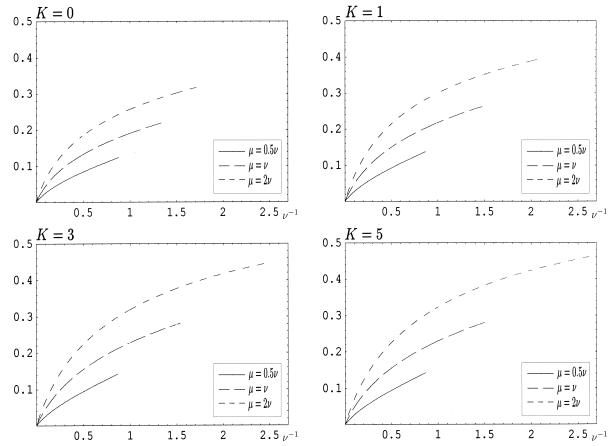


Figure 1. $P(\zeta_B + \zeta_D > 0)$ versus ν^{-1}.

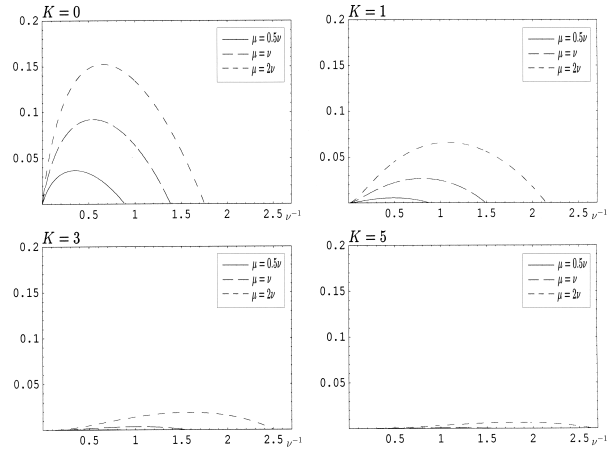


Figure 2. $E[\zeta_B]/E[\zeta]$ versus ν^{-1}

$P_b > P(\zeta_B > 0)$ when $\nu^{-1} > \lambda^{-1}$, with independence of K, μ and ν.

Figure 1 illustrates the probability $P(\zeta_B + \zeta_D > 0)$ of an arriving unit can be delayed (because blocking or it must wait in the buffer of Station 2). Although the figure is itself explanatory, we point out that $P(\zeta_B + \zeta_D > 0)$ increase with increasing values of ν^{-1}, for fixed values of K. Besides, the highest magnitudes correspond to the case $\mu = 2\nu$. Note that $P(\zeta_D > 0)$ equals 0 in the case $K = 0$, so for this case we draw the quantities given in the third column of Table 3.

From the Laplace-Stieltjes transform of ζ_B and ζ_D it is straightforward to obtain the mean values

$$E[\zeta_B] = (\boldsymbol{\alpha} \otimes \boldsymbol{\Phi}(0|K+1))(-\mathbf{T} \oplus \mathbf{S})^{-1}\boldsymbol{\Delta},$$

$$E[\zeta_D] = \left(\boldsymbol{\alpha} \otimes \sum_{m=1}^{K} \boldsymbol{\Phi}(0|m)\right)(-\mathbf{T} \oplus \mathbf{S})^{-1}\boldsymbol{\Delta},$$

where $\boldsymbol{\Delta} = (\mathbf{t}_0 \otimes ((-\mathbf{S}^{-1})\mathbf{e}_s))$.
Let $\zeta = \zeta_W + \zeta_1 + \zeta_B + \zeta_D + \zeta_2$ be the sojourn time of an arriving unit. Expressions for $E[\zeta_W]$ and $E[\zeta_k]$, with $k \in \{1, 2\}$, are readily derived from (2). In Figures 2-3 we show the fractions $E[\zeta_B]/E[\zeta]$ and $E[\zeta_D]/E[\zeta]$, respectively. From these figures, we see that these mean

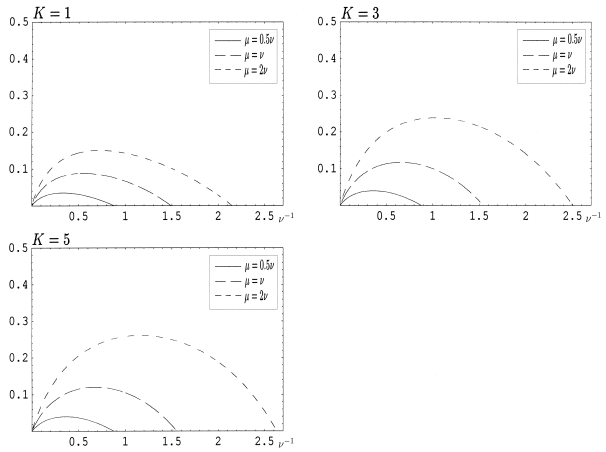


Figure 3. $E[\zeta_D]/E[\zeta]$ versus ν^{-1}

times increase till a maximum value and after that, they decrease to 0. Furthermore, $E[\zeta_B]/E[\zeta]$ (respectively, $E[\zeta_D]/E[\zeta]$) increase (respectively, decrease) with increasing values of K. Note that $E[\zeta_D] = 0$ when $K = 0$.

To conclude, we remark that the accuracy required at Step 2 in Algorithm 1 has been set to $\epsilon = 10^{-11}$. Solution algorithms were implemented in Fortran programming language with double precision floating point arithmetic.

Acknowledgements

The autor thanks anonymous referees for their valuable comments and suggestions. The research of the author was supported by MEC, Grant No. MTM 2005-01248.

REFERENCES

Clarke, A.B. 1977. "A two-server tandem queueing system with storage between servers". *Mathematical Report* no. 50. Western Michigan University, Kalamazoo.

Gómez-Corral, A. 2004. "Sojourn times in a two-stage queueing network with blocking". *Naval Research Logistics* 51, 1068-1089.

Gómez-Corral, A. and M.E. Martos. "A.B. Clarke's Tandem Queue Revisited - Sojourn Times". Submitted.

Lillo, R.E. and M.F. Neuts. 1999. "Two service units with interference in the access to servers". *Journal of Applied Mathematics and Stochastic Analysis* 12, 357-370.

Lucantoni, D.M.; K.S. Meier-Hellstern and M.F. Neuts. 1990. "A single-server queue with server vacations and a class of non-renewal arrival processes". *Advances in Applied Probability* 22, 676-705.

Neuts, M.F. 1994. "Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach". Second Edition. Dover Publications, Inc., New York.

MANUEL E. MARTOS is a Ph.D. candidate at the Department of Statistics and Operations Research I, Complutense University of Madrid, under the direction of A. Gómez-Corral. He is a Bachellor of Statistics since 2001 from the University of Granada, Spain. He joined Complutense University of Madrid in 2003 as a PhD grantee. His current research interests include performance modelling and analysis of queueing systems through matrix-analytic methods.

DECOMPOSITION OF EXPONENTIAL G-NETWORKS WITH DEPENDENT SERVICE AND ROUTE CHANGE

Ciro D'Apice
Department of Information Engineering
and Applied Mathematics
University of Salerno
Via Ponte don Melillo, 84084 Fisciano (SA), Italy
Email: dapice@diima.unisa.it

Alexander Pechinkin
Institute of Informatics Problems, RAS
Vavilova 44, bldg 2, Moscow, 119333, Russia
Email: APechinkin@ipiran.ru

Sergey Shorgin
Institute of Informatics Problems, RAS
Vavilova 44, bldg 2, Moscow 119333, Russia
Email: SShorgin@ipiran.ru

KEYWORDS
Exponential networks, negative customers, dependent service, product form solution.

ABSTRACT

Queueing networks with negative customers (G-networks), Poisson flow of positive customers, multi-server exponential nodes, and dependent service at different nodes are studied. Every customer arriving at the network is defined by a set of random parameters: customer route, customer route length, customer volume and its service time at each route stage. A negative customer arriving to a queuing system removes (or "kills") one of the ordinary (or "positive") customers if any is present. However, the "killed" customer does not quit the network but continues his way along the new random route. For such G-networks, the multidimensional stationary distribution of the network state probabilities is shown to be representable in product form.

INTRODUCTION

Queueing networks are widely used for analytical modelling of information systems. One of the major problems of information systems modelling is the analysis of influence of various external interference factors, including viruses, etc.
The investigation of networks with so called negative customers, or the G-networks introduced by E. Gelenbe (Gelenbe 1989a; Gelenbe 1989b; Gelenbe 1990a; Gelenbe 1990b) has attracted a lot of interest in recent years due to the wide range of applications. G-networks are open queueing networks for which, along with usual (positive) customers, additional flows of negative customers are considered (Gelenbe 1991; Gelenbe et al. 1991). A negative customer, unlike positive customers, upon arrival at a network node, without receiving any service, "kills" one positive customer if there is any at this node and then quits the network. G-networks, initially motivated by neural network and associative memory, have been applied to evaluate the performance of unreliable information systems and to model virus behavior in a computer network. Many examples of applications of G-networks were given in (Cramer and Gelenbe 2000; Feng and Gelenbe 1999; Gelenbe et al. 2001; Gelenbe and Shachnai 2000)
Numerous scientific works have been devoted to the analysis of G-networks. In particular, a review on G-networks with an extensive bibliography is presented in (Bocharov and Vishnevskii 2003).
A non-traditional open queueing networks with negative customers and dependent service have been studied in (Bocharov et al. 2004a,b,c,d; Bocharov et al. 2006). In the present paper which extends the mentioned results, a new variant of exponential G-networks with dependent service and route change of "killed" customers is considered. In the new variant, a "killed" customer does not leave the network but continues to move through the network according to a new random route, inducing an additional loading. For such G-networks, the multidimensional stationary distribution of the network state probabilities is shown to be representable in product form.

NETWORK DESCRIPTION

We consider an open queueing network with M multi-server exponential nodes. Each node has an infinite buffer. We denote by c_s the number of servers in sth node, $s = \overline{1, M}$. In what follows random variables (RV) are denoted with capital Latin letters, and their realizations with corresponding lower case letters. Additionally, we use a semiboldface font for vector random variables and vectors.
A Poisson flow of (usual, positive) customers of intensity λ enters the network. Each customer arriving at the network is characterized by a set of ran-

dom variables $(L, \boldsymbol{R}, \boldsymbol{Y}, \boldsymbol{X})$, which depend neither on analogous random variables for other customers nor on network pre-history, where:
— L is the customer route random length, i.e. the number of stages (nodes) at which he will be served;
— $\boldsymbol{R} = (R_1, \ldots, R_L)$ is a random route which is a set of node numbers (the same nodes at different stages are allowed) that the customer passes through in consecutive order at all L stages;
— $\boldsymbol{Y} = (Y_1, \ldots, Y_L)$ are customer random volumes at route stages the customer consecutively passes through (the case when these volumes are different at different stages is considered too);
— $\boldsymbol{X} = (X_1, \ldots, X_L)$ are customer random service times at the route stages the customer consecutively passes through.
It is obvious that under this network description the volume Y_n and the service time X_n define the service of a customer at node R_n. Let us recall that routes $\boldsymbol{R}$ with repeated numbers are allowed, i.e. a customer can be served at the same node s several times (but probably with different customer volumes). Service time X_n at node $R_n = s$ depends on neither service processes of other customers, nor the route $\boldsymbol{R}$, nor any of the volumes Y_k (including the volume Y_n), nor any of the times X_k of customer service at other route stages and has exponential probability distribution function (PDF) with parameter μ_s.
Stochastic characteristics of a random variable $(L, \boldsymbol{R}, \boldsymbol{Y})$ are given by the joint PDF

$$G(l, \boldsymbol{r}, \boldsymbol{y}) = \mathbf{P}\{L = l,\, R_n = r_n,\, Y_n \le y_n, \quad n = \overline{1,l}\}.$$

We shall make an additional technical assumption on PDF $G(l, \boldsymbol{r}, \boldsymbol{y})$. Namely, we suppose that the PDF $G(l, \boldsymbol{r}, \boldsymbol{y})$ are absolutely continuous, and denote by $g(l, \boldsymbol{r}, \boldsymbol{y})$ its density, i.e.

$$g(l, \boldsymbol{r}, \boldsymbol{y}) = \frac{\partial^l}{\partial y_1 \cdots \partial y_l} G(l, \boldsymbol{r}, \boldsymbol{y}).$$

This assumption could be easily neglected if we interpret derivatives as generalized ones.
Along with the flow of positive customers described above, flows of negative customers arrive at the network. These flows are defined in the following way.
Neg 1. The flows arriving at different nodes are independent.
Neg 2. A customer flow arriving at node s is a Poisson one of intensity $c_s \gamma_s$.
Neg 3. A negative customer arriving at node s with the same probability $1/c_s$ chooses one of the servers. If at the chosen server a "not killed" customer is being served, the negative customer immediately "kills" it and quits the network. However, the "killed" positive customer does not leave the system but passes to the last place in node R_1^* and continues to be served according to the new RV $(L^*, \boldsymbol{R}^*, \boldsymbol{Y}^*, \boldsymbol{X}^*)$. The distribution of RV $(L^*, \boldsymbol{R}^*, \boldsymbol{Y}^*)$ depends only on RV $(L, \boldsymbol{R}, \boldsymbol{Y})$ of "killed" customer and N, i.e., number of stage at which a customer has been "killed" in node $R_N = s$. The RV has, under the condition that $(L, \boldsymbol{R}, \boldsymbol{Y}) = (l, \boldsymbol{r}, \boldsymbol{y})$ and the customer has been "killed" at the stage with number $N = n$, the following conditional distribution function

$$H(l^*, \boldsymbol{r}^*, \boldsymbol{y}^* \mid l, \boldsymbol{r}, \boldsymbol{y}, n) =$$

$$\mathbf{P}\big\{L^* = l^*,\, R_m^* = r_m^*,\, Y_m^* \le y_m^*,\ m = \overline{1,l^*} \,\big|\, (L, \boldsymbol{R}, \boldsymbol{Y}) = (l, \boldsymbol{r}, \boldsymbol{y}),\ N = n\big\}.$$

Further, for simplification of notation, we shall assume that $H(l^*, \boldsymbol{r}^*, \boldsymbol{y}^* \mid l, \boldsymbol{r}, \boldsymbol{y}, n)$ considered as function of argument $\boldsymbol{y}^*$ is absolutely continuous. We shall denote its density by $h(l^*, \boldsymbol{r}^*, \boldsymbol{y}^* \mid l, \boldsymbol{r}, \boldsymbol{y}, n)$ which is equal to

$$h(l^*, \boldsymbol{r}^*, \boldsymbol{y}^* \mid l, \boldsymbol{r}, \boldsymbol{y}, n) =$$

$$\frac{\partial^l}{\partial y_1^* \cdots \partial y_l^*} H(l^*, \boldsymbol{r}^*, \boldsymbol{y}^* \mid l, \boldsymbol{r}, \boldsymbol{y}, n).$$

As for "not killed" customers, the time X_m^* of a "killed" customer service in node $R_m^* = s$ does not depend on other network parameters and has an exponential distribution with parameter μ_s. If there is an already "killed" positive customer on the chosen server, it simply passes to the node corresponding to the consequent service stage and continues to be served accordingly to its set of parameters. Finally, if at the moment of a negative customer's arrival into some node there are no positive customers, then the negative customer quits the network without inducing any action.

AUXILIARY FUNCTIONS

Let us set

$$\omega_n(l, \boldsymbol{r}, \boldsymbol{y}) = \frac{\mu_{r_n}}{\mu_{r_n} + \gamma_{r_n}}, \quad n = \overline{1,l},$$

$$\omega_n^*(l, \boldsymbol{r}, \boldsymbol{y}) = \prod_{i=1}^{n-1} \omega_i(l, \boldsymbol{r}, \boldsymbol{y}), \quad n = \overline{1,l+1},$$

$$g_n^*(l, \boldsymbol{r}, \boldsymbol{y}) = \omega_n^*(l, \boldsymbol{r}, \boldsymbol{y})\, g(l, \boldsymbol{r}, \boldsymbol{y}), \quad n = \overline{1,l},$$

$$h(l, \boldsymbol{r}, \boldsymbol{y}) = \sum_{k,s,n} \int_{\mathbb{R}^k} \frac{\gamma_{s_n} \omega_n^*(k, \boldsymbol{s}, \boldsymbol{u})}{\mu_{s_n} + \gamma_{s_n}} \times$$

$$h(l, \boldsymbol{r}, \boldsymbol{y} \mid k, \boldsymbol{s}, \boldsymbol{u}, n)\, g(k, \boldsymbol{s}, \boldsymbol{u})\, \boldsymbol{du}.$$

For the sake of brevity we shall use the notations

$$\int_{\mathbb{R}^l} \ldots \boldsymbol{dy} = \int \ldots \int_{\mathbb{R}^l} \ldots dy_1 \cdots dy_l.$$

Let us introduce the following quantities for the sth node

$$\rho_s^+ = \frac{\lambda}{\mu_s + \gamma_s} \sum_{l,\boldsymbol{r}} \int\limits_{\mathbb{R}^l} \sum_{n=1}^{l} g_n^*(l, \boldsymbol{r}, \boldsymbol{y})\, \delta_{s-r_n}\, \boldsymbol{dy},$$

$$\rho_s^- = \frac{\lambda}{\mu_s + \gamma_s} \sum_{l,\boldsymbol{r}} \int\limits_{\mathbb{R}^l} \sum_{n=1}^{l} h(l, \boldsymbol{r}, \boldsymbol{y})\, \delta_{s-r_n}\, \boldsymbol{dy},$$

$$\rho_s = \rho_s^+ + \rho_s^-,$$

where δ_j is the Kronecker symbol.
These functions have a very transparent physical meaning:
— $\omega_n(l, \boldsymbol{r}, \boldsymbol{y})$ is the probability that a positive customer with parameters $(l, \boldsymbol{r}, \boldsymbol{y})$ "not killed" until the nth stage will not be "killed" at this stage (at node r_n);
— $\omega_n^*(l, \boldsymbol{r}, \boldsymbol{y})$ is the probability that a positive customer with parameters $(l, \boldsymbol{r}, \boldsymbol{y})$ will not be "killed" until the nth stage;
— ρ_s is the general traffic intensity at the sth node;
— ρ_s^+ is the traffic intensity of "not killed" customers at the sth node;
— ρ_s^- is the traffic intensity of "killed" but still being served customers at the sth node.

MARKOV PROCESS

Let us define the Markov process that describes the behavior of our queueing network.
We shall denote a network state by a set $\boldsymbol{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M)$, where $\boldsymbol{z}_s = (k_s, \boldsymbol{z}_{s1}, \ldots, \boldsymbol{z}_{sk_s})$, $s = \overline{1, M}$, in turn, describes the state of the sth node: k_s is the number of customers at the sth node and $\boldsymbol{z}_{si}$, $s = \overline{1, M}$, $i = \overline{1, k_s}$, with components $\boldsymbol{z}_{si} = (l_{si}, \boldsymbol{r}_{si}, \boldsymbol{y}_{si}, n_{si}, w_{si})$, stores the following information $(l_{si}, \boldsymbol{r}_{si}, \boldsymbol{y}_{si})$ on the ith customer at the sth node, and its position (n_{si}, x_{si}, w_{si}) in the network:
— l_{si} is the route length;
— $\boldsymbol{r}_{si} = (r_{si1}, \ldots, r_{sil_{si}})$ is the route;
— $\boldsymbol{y}_{si} = (y_{si1}, \ldots, y_{sil_{si}})$ are customer volumes at its route stages;
— n_{si} is the number of the route's stage at which the customer exists (while being served or waiting for service); clearly, $n_{si} \le l_{si}$;
— w_{si} is the function which shows customer state; we set $w_{si} = 0$ if the customer is "not killed", and $w_{si} = 1$ if the customer is "killed" (but is being served).
It is evident that due to the notations introduced above, we have $r_{sin_{si}} = s$. It is also clear that $\boldsymbol{z}_s = 0$ if $k_s = 0$, i.e. when there are no customers at the sth node, and $\boldsymbol{z} = \boldsymbol{0} = (0, \ldots, 0)$ in the case, when there are no customers in the network at all.
Henceforth, customers at nodes are numbered in the order of their arrival at the node.

The set of states of the network is denoted by $\mathcal{Z} = \{\boldsymbol{z}\}$.
To describe the behavior of our queueing network, let us consider the process

$$\boldsymbol{Z}(t) = \boldsymbol{z}, \quad \begin{array}{l}\text{if the network exists}\\ \text{in state } \boldsymbol{z} \text{ at instant } t.\end{array}$$

It is obviously a Markov process.
In the sequel, we also consider the (non-Markovian) processes

$$\boldsymbol{Z}_s(t) = \boldsymbol{z}_s, \quad \begin{array}{l}\text{if the network exists}\\ \text{in state } \boldsymbol{z} \text{ at instant } t,\end{array} \quad s = \overline{1, M},$$

describing the behavior of individual network nodes.

STATIONARY DISTRIBUTION OF THE MARKOV PROCESS

The stationary density of the state probability distribution of the process $\boldsymbol{Z}(t)$ is denoted by $p(\boldsymbol{z})$.

Theorem. *If for all nodes the condition $\rho_s < c_s$ is fulfilled, then there exists a limiting (stationary) distribution of probabilities of states of the process $\boldsymbol{Z}(t)$ with probability distribution density*

$$p(\boldsymbol{z}) = \prod_{s=1}^{M} p_s(\boldsymbol{z}_s), \tag{1}$$

thereby

$$p_s(\boldsymbol{z}_s) = p_s(0)\, d_s(k_s) \times$$
$$\prod_{i=1}^{k_s} \frac{\lambda(\delta_{w_{si}} g_{n_{si}}^*(l_{si}, \boldsymbol{r}_{si}, \boldsymbol{y}_{si}) + \delta_{1-w_{si}} h_{n_{si}}(l_{si}, \boldsymbol{r}_{si}, \boldsymbol{y}_{si}))}{\mu_s + \gamma_s}, \tag{2}$$

where

$$p_s(0) = \left(\sum_{i=0}^{c_s} \frac{\rho_s^i}{i!} + \frac{\rho_s^{c_s+1}}{c_s!\,(c_s - \rho_s)} \right)^{-1}, \tag{3}$$

$$d_s(k_s) = \begin{cases} 1/k_s! & \text{for } k_s \le c_s, \\ 1/(c_s!\, c_s^{k_s - c_s}) & \text{for } k_s > c_s. \end{cases} \tag{4}$$

Proof. It is easily seen that the Markov process $\boldsymbol{Z}(t)$ is regular and positive Harris recurrent, because the state $\boldsymbol{0}$ is a positive recurrent atom for it. Therefore, to prove the theorem it is sufficient to show that the function $p(\boldsymbol{z})$, satisfies the system of equations for the stationary probability density of states of the process $\boldsymbol{Z}(t)$. In analogy with the discrete case, we refer to this system of equations as the system of equilibrium equations.
Let us associate to every state $\boldsymbol{z} \in \mathcal{Z}$ a state set $\tilde{\mathcal{Z}}(\boldsymbol{z})$. The states belonging to the state $\tilde{\mathcal{Z}}(\boldsymbol{z})$ are called the predecessors of the state $\boldsymbol{z}$. Predecessor states are introduced for the reason that a direct transition to the state $\boldsymbol{z}$ is only possible from these states.
In turn, let us subdivide all predecessors of a state $\boldsymbol{z}$ into 5 disjoint classes. A definite type of transition to the state $\boldsymbol{z}$ corresponds to every class.

The first class $\mathcal{Z}^+(\boldsymbol{z}) = \mathcal{Z}_1^+(\boldsymbol{z}) \cup \mathcal{Z}_2^+(\boldsymbol{z})$ consists of two (intersecting) subclasses $\mathcal{Z}_1^+(\boldsymbol{z})$ and $\mathcal{Z}_2^+(\boldsymbol{z})$.
The first subclass $\mathcal{Z}_1^+(\boldsymbol{z}) = \{\boldsymbol{z}^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y})\}$ is a set of network states from which a transition to the state $\boldsymbol{z}$ is possible owing to the exit of a "not killed" customer from the network upon completion of its service (at the last node in the route) and contains only the states $\boldsymbol{z}^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y})$ of the form

$$\boldsymbol{z}_1^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y}) = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{s-1}, \boldsymbol{z}_s^*, \boldsymbol{z}_{s+1}, \ldots, \boldsymbol{z}_M),$$

where

$$s = r_l, \quad \boldsymbol{z}_s^* = (k_s+1, \boldsymbol{z}_{s1}^*, \ldots, \boldsymbol{z}_{s,k_s+1}^*),$$

$$i = \overline{1, \min\{k_s+1, c_s\}},$$

$$\boldsymbol{z}_{sj}^* = \begin{cases} \boldsymbol{z}_{sj}, & j < i, \\ (l, \boldsymbol{r}, \boldsymbol{y}, l, 0), & j = i, \\ \boldsymbol{z}_{s,j-1}, & j > i, \end{cases}$$

and $(l, \boldsymbol{r}, \boldsymbol{y})$ are the parameters of the customer that leaves the ith place at the sth node upon full completion of its service. These parameters may take any (possible) value.
The second subclass $\mathcal{Z}_2^+(\boldsymbol{z}) = \{\boldsymbol{z}_2^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y}, n)\}$ is similar to the first one but corresponds to states of a network from which the transition to a state $\boldsymbol{z}$ is possible due to exit of "killed" customer upon the full completion of its service (at the last node in the route) or due to the action on it, also at the last node in the route, of a newly arrived negative customer. This class contains all states $\boldsymbol{z}_2^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y})$ which have the form

$$\boldsymbol{z}_2^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y}) = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{s-1}, \boldsymbol{z}_s^*, \boldsymbol{z}_{s+1}, \ldots, \boldsymbol{z}_M),$$

where

$$s = r_l, \quad \boldsymbol{z}_s^* = (k_s+1, \boldsymbol{z}_{s1}^*, \ldots, \boldsymbol{z}_{s,k_s+1}^*),$$

$$i = \overline{1, \min\{k_s+1, c_s\}},$$

$$\boldsymbol{z}_{sj}^* = \begin{cases} \boldsymbol{z}_{sj}, & j < i, \\ (l, \boldsymbol{r}, \boldsymbol{y}, l, 1), & j = i, \\ \boldsymbol{z}_{s,j-1}, & j > i, \end{cases}$$

and $(l, \boldsymbol{r}, \boldsymbol{y})$ are parameters of the "killed" customer that leaves the system from the ith place in sth node. These parameters may take any (possible) values, too.
The second class $\mathcal{Z}^-(\boldsymbol{z})$ is introduced as follows. Let us consider co-ordinate $\boldsymbol{z}_s$ of vector $\boldsymbol{z}$ for any node s which satisfied $k_s > 0$.
Let $n_{sk_s} = 1$ and $w_{sk_s} = 0$. Then we say that the node s belongs to the node set $\mathcal{S}^-(\boldsymbol{z})$. Let $\boldsymbol{z}^-(\boldsymbol{z}, s)$ denote the state

$$\boldsymbol{z}^-(\boldsymbol{z}, s) = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{s-1}, \boldsymbol{z}_s^*, \boldsymbol{z}_{s+1}, \ldots, \boldsymbol{z}_M),$$

where

$$\boldsymbol{z}_s^* = (k_s - 1, \boldsymbol{z}_{s1}, \ldots, \boldsymbol{z}_{s,k_s-1}).$$

The states $\boldsymbol{z}^-(\boldsymbol{z}, s)$ are called predecessor states of the class $\mathcal{Z}^-(\boldsymbol{z})$. Obviously, the class $\mathcal{Z}^-(\boldsymbol{z})$ consists of states from which a transition to the state $\boldsymbol{z}$ is possible due to the arrival of a new customer, and the set $\mathcal{S}^-(\boldsymbol{z})$ consists of nodes in which the last "not killed" customer that had arrived at the network exists at the last place under the state $\boldsymbol{z}$.
Let us determine **the third class** $\mathcal{Z}_1(\boldsymbol{z})$. We say that the node s belongs to the state set $\mathcal{S}_1(\boldsymbol{z})$, if $n_{sk_s} > 1$ and $w_{sk_s} = 0$. The class $\mathcal{Z}_1(\boldsymbol{z})$ defines the predecessor states from which a transition to the state $\boldsymbol{z}$ takes place when a ("not killed") customer upon completion of service in a node moves to another node (at the last place). This class consists of disjoint subclasses $\mathcal{Z}_1(\boldsymbol{z}, s)$, which are defined as follows. Let $\mathcal{Z}_1(\boldsymbol{z}, s) = \{\boldsymbol{z}(\boldsymbol{z}, s, j)\}$. Every state $\boldsymbol{z}(\boldsymbol{z}, s, j)$ is of the form

$$\boldsymbol{z}_1(\boldsymbol{z}, s, j) = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{s'-1}, \boldsymbol{z}_{s'}^*, \boldsymbol{z}_{s'+1}, \ldots, \boldsymbol{z}_{s-1}, \boldsymbol{z}_s^*, \boldsymbol{z}_{s+1}, \ldots, \boldsymbol{z}_M)$$

(obviously, s can precede s' and even coincide with s', if a customer once again arrives at the sth node from the sth node), where

$$s' = r_{s,k_s,n_{sk_s}-1}, \quad \boldsymbol{z}_{s'}^* = (k_{s'}+1, \boldsymbol{z}_{s'1}^*, \ldots, \boldsymbol{z}_{s',k_{s'}+1}^*),$$

$$j = \overline{1, \min\{k_{s'}+1, c_{s'}\}},$$

$$\boldsymbol{z}_{s'i}^* = \begin{cases} \boldsymbol{z}_{s'i}, & i < j, \\ (l_{sk_s}, \boldsymbol{r}_{sk_s}, \boldsymbol{y}_{sk_s}, n_{sk_s}-1, 0), & i = j, \\ \boldsymbol{z}_{s',i-1}, & i > j, \end{cases}$$

$$\boldsymbol{z}_s^* = (k_s - 1, \boldsymbol{z}_{s1}, \ldots, \boldsymbol{z}_{s,k_s-1}).$$

The fourth class $\mathcal{Z}_2(\boldsymbol{z})$ is analogous to class $\mathcal{Z}_1(\boldsymbol{z})$ but corresponds to preceding states, transition from which to state $\boldsymbol{z}$ is carried out at the completion of "killed" customer service in a node and its arrival for service to other nodes (to the last place).The node s is attributed to the set of nodes $\mathcal{S}_2(\boldsymbol{z})$ if $n_{sk_s} > 1$ and $w_{sk_s} = 1$. The class $\mathcal{Z}_2(\boldsymbol{z})$ consists of not intersecting subclasses $\mathcal{Z}_2(\boldsymbol{z}, s)$ determined as follows. Let us denote $\mathcal{Z}_2(\boldsymbol{z}, s) = \{\boldsymbol{z}_2(\boldsymbol{z}, s, j)\}$. At every state $\boldsymbol{z}_2(\boldsymbol{z}, s, j)$ is of the form

$$\boldsymbol{z}_2(\boldsymbol{z}, s, j) = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{s'-1}, \boldsymbol{z}_{s'}^*, \boldsymbol{z}_{s'+1}, \ldots, \boldsymbol{z}_{s-1}, \boldsymbol{z}_s^*, \boldsymbol{z}_{s+1}, \ldots, \boldsymbol{z}_M)$$

(as above, s can precede s' and even coincide with s'), where

$$s' = r_{s,k_s,n_{sk_s}-1}, \quad \boldsymbol{z}_{s'}^* = (k_{s'}+1, \boldsymbol{z}_{s'1}^*, \ldots, \boldsymbol{z}_{s',k_{s'}+1}^*),$$

$$j = \overline{1, \min\{k_{s'}+1, c_{s'}\}},$$

$$\boldsymbol{z}_{s'i}^* = \begin{cases} \boldsymbol{z}_{s'i}, & i < j, \\ (l_{sk_s}, \boldsymbol{r}_{sk_s}, \boldsymbol{y}_{sk_s}, n_{sk_s}-1, 1), & i = j, \\ \boldsymbol{z}_{s',i-1}, & i > j, \end{cases}$$

$$\boldsymbol{z}^*_s = (k_s - 1, \boldsymbol{z}_{s1}, \ldots, \boldsymbol{z}_{s,k_s-1}).$$

At last, **the fifth class** $\mathcal{Z}_3(\boldsymbol{z})$ is constructed as follows. If $n_{sk_s} = 1$ and $w_{sk_s} = 1$, the node s is attributed to the node set $\mathcal{S}_3(\boldsymbol{z})$. The class $\mathcal{Z}_3(\boldsymbol{z})$ consists of not intersecting subclasses $\mathcal{Z}_3(\boldsymbol{z}, s) = \{\boldsymbol{z}(\boldsymbol{z}, s, s', i, l, \boldsymbol{r}, \boldsymbol{y}, n)\}$. At any state $\{\boldsymbol{z}(\boldsymbol{z}, s, s', i, l, \boldsymbol{r}, \boldsymbol{y}, n)\}$ has the form

$$\{\boldsymbol{z}(\boldsymbol{z}, s, s', i, l, \boldsymbol{r}, \boldsymbol{y}, n)\} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{s'-1}, \boldsymbol{z}^*_{s'}, \boldsymbol{z}_{s'+1}, \ldots, \boldsymbol{z}_{s-1}, \boldsymbol{z}^*_s, \boldsymbol{z}_{s+1}, \ldots, \boldsymbol{z}_M)$$

(we would remind that s can precede s' and even coincide with s'), where

$$n = \overline{1, l}, \quad \boldsymbol{z}^*_{s'} = (k_{s'} + 1, \boldsymbol{z}^*_{s'1}, \ldots, \boldsymbol{z}^*_{s',k_{s'}+1}),$$

$$i = \overline{1, \min\{k_{s'} + 1, c_{s'}\}},$$

$$\boldsymbol{z}^*_{s'j} = \begin{cases} \boldsymbol{z}_{s'j}, & j < i, \\ (l, \boldsymbol{r}, \boldsymbol{y}, l, 0), & j = i, \\ \boldsymbol{z}_{s',j-1}, & j > i, \end{cases}$$

$$\boldsymbol{z}^*_s = (k_s - 1, \boldsymbol{z}_{s1}, \ldots, \boldsymbol{z}_{s,k_s-1}),$$

and $(l, \boldsymbol{r}, \boldsymbol{y})$ are the parameters of customer "killed" at ith place at s'th node and transferred to the last place of sth node. These parameters may take any (possible) value. The state $\{\boldsymbol{z}(\boldsymbol{z}, s, s', i, l, \boldsymbol{r}, \boldsymbol{y}, n)\}$ is the preceding state for which the transition from it to state $\boldsymbol{z}$ is accomplished when a negative customer arrives to ith server of node s', "kills" the "not killed" customer which have been served at this place, and transfers the last to the node s at the last one place. Let us once again note that each of the classes $\mathcal{Z}^-(\boldsymbol{z})$, $\mathcal{Z}_1(\boldsymbol{z})$, $\mathcal{Z}_2(\boldsymbol{z})$ and $\mathcal{Z}_3(\boldsymbol{z})$ can be empty. In particular, all classes $\mathcal{Z}^-(\boldsymbol{z})$, $\mathcal{Z}_1(\boldsymbol{z})$, $\mathcal{Z}_2(\boldsymbol{z})$ and $\mathcal{Z}_3(\boldsymbol{z})$ are empty for the state $\mathbf{0} = (0, \ldots, 0)$.

Let us write the global balance equations for a state $\boldsymbol{z}$. For this, we compute the probability flows arriving at and departing from the state $\boldsymbol{z}$.

Departure from the state $\boldsymbol{z}$ takes place upon arrival of a new customer at the network (with intensity λ). The probability flow departing from the state $\boldsymbol{z}$ due to the arrival of a new customer is $\lambda p(\boldsymbol{z})$.

Departure from the state $\boldsymbol{z}$ also takes place upon completion of service of a customer at the ith server at the sth node (with intensity μ_s which does not depend whether this customer is "killed" or not) or arrival of a negative customer at this server at this node (with intensity γ_s). The total probability flow departing from the state $\boldsymbol{z}$ due to completion of service of customers or arrivals of negative customers is

$$\sum_{s=1}^{M} \sum_{i=1}^{\min\{k_s, c_s\}} (\mu_s + \gamma_s)\, p(\boldsymbol{z})$$

(let us note that in the sum over s the nodes for which $k_s = 0$ are not included).

Arrival at a state $\boldsymbol{z}$ may take place from the state $\boldsymbol{z}^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y}) \in \mathcal{Z}_1^+(\boldsymbol{z})$ upon completion of service of the "not killed" customer at the ith server at the r_lth node (at the last route's node). Since the intensity of completion service at the r_lth node is μ_{r_l}, the total probability flow of arrivals at the state $\boldsymbol{z}$ from the states of the subclass $\mathcal{Z}_1^+(\boldsymbol{z})$ is

$$\sum_{s=1}^{M} \sum_{i=1}^{\min\{k_s+1, c_s\}} \sum_{l, \boldsymbol{r}} \int_{\mathbb{R}^l} \mu_s \delta_{s-r_l} p(\boldsymbol{z}_1^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y}))\, \boldsymbol{dy}.$$

Analogously, the total probability flow of arrivals at the state $\boldsymbol{z}$ from the states of the subclass $\mathcal{Z}_2^+(\boldsymbol{z})$ (which is connected with the termination of service of "killed" customers or with the impact of negative customers on them at the last stage) is

$$\sum_{s=1}^{M} \sum_{i=1}^{\min\{k_s+1, c_s\}} \sum_{l, \boldsymbol{r}} \int_{\mathbb{R}^l} (\mu_s + \gamma_s) \times \delta_{s-r_l} p(\boldsymbol{z}_2^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y}))\, \boldsymbol{dy}.$$

Further, the state $\boldsymbol{z}$ can be reached from the state $\boldsymbol{z}^-(\boldsymbol{z}, s) \in \mathcal{Z}^-(\boldsymbol{z})$ if a new customer arrives at the network at the last place of the node s (with intensity $\lambda g(l_{sk_s}, \boldsymbol{r}_{sk_s}, \boldsymbol{y}_{sk_s})$). The total probability flow of arrivals to the state $\boldsymbol{z}$ due to such transitions is

$$\sum_{s \in \mathcal{S}^-(\boldsymbol{z})} \lambda g(l_{sk_s}, \boldsymbol{r}_{sk_s}, \boldsymbol{y}_{sk_s})\, p(\boldsymbol{z}(\boldsymbol{z}, s)).$$

The state $\boldsymbol{z}$ can be also reached from $\boldsymbol{z}(\boldsymbol{z}, s, i) \in \mathcal{Z}_1(\boldsymbol{z})$ upon completion of service of the "not killed" customer at ith server of the node $s' = r_{s,k_s,n_{sk_s}-1}$ (with intensity $\mu_{s'}$) and its transition at the sth node place. The total probability flow of arrivals to the state $\boldsymbol{z}$ from the states of the subclass $\mathcal{Z}_1(\boldsymbol{z})$ is

$$\sum_{s \in \mathcal{S}_1(\boldsymbol{z})} \sum_{i=1}^{\min\{k_{s'}+1, c_{s'}\}} \mu_{s'} p(\boldsymbol{z}_1(\boldsymbol{z}, s, i)).$$

Analogously, total probability flow of arrivals at the state $\boldsymbol{z}$ from the states of the subclass $\mathcal{Z}_2(\boldsymbol{z})$ (which is connected with transitions of "killed" customers) is

$$\sum_{s \in \mathcal{S}_2(\boldsymbol{z})} \sum_{i=1}^{\min\{k_{s'}+1, c_{s'}\}} \mu_{s'} p(\boldsymbol{z}_2(\boldsymbol{z}, s, i)).$$

Finally, the arrival to the state $\boldsymbol{z}$ may be fulfilled from the state $\{\boldsymbol{z}(\boldsymbol{z}, s, s', i, l, \boldsymbol{r}, \boldsymbol{y}, n)\} \in \mathcal{Z}_3(\boldsymbol{z})$ when a negative customer appears in node s' at ith server (with intensity $\gamma_{s'}$) and a "killed" customer at this place and stage n customer with parameters $(l, \boldsymbol{r}, \boldsymbol{y})$ passes to node s at the last place (with probability density $h(l_{sk_s}, \boldsymbol{r}_{sk_s}, \boldsymbol{y}_{sk_s} \mid l, \boldsymbol{r}, \boldsymbol{y}, n)$). In the case, the total probability flow of arrivals at the state $\boldsymbol{z}$ the states of the class $\mathcal{Z}_3(\boldsymbol{z})$ is equal to

$$\sum_{s \in \mathcal{S}_3(\boldsymbol{z})} \sum_{s'=1}^{M} \sum_{i=1}^{\min\{k_s+1, c_s\}} \sum_{l, r, n} \int_{\mathbb{R}^l} \gamma_{s'} \delta_{s'-r_n} \times$$

$$h(l_{sk_s}, \boldsymbol{r}_{sk_s}, \boldsymbol{y}_{sk_s} \mid l, \boldsymbol{r}, \boldsymbol{y}, n) \times$$
$$p(\boldsymbol{z}(\boldsymbol{z}, s, s', i, l, \boldsymbol{r}, \boldsymbol{y}, n))\, \boldsymbol{dy}.$$

Now we can write the system of equilibrium equations for the state $\boldsymbol{z}$:

$$\lambda p(\boldsymbol{z}) + \sum_{s=1}^{M} \sum_{i=1}^{\min\{k_s, c_s\}} (\mu_s + \gamma_s)\, p(\boldsymbol{z}) =$$

$$\sum_{s=1}^{M} \sum_{i=1}^{\min\{k_s+1, c_s\}} \sum_{l,\boldsymbol{r}} \int_{\mathbb{R}^l} \mu_s \delta_{s-r_l} p(\boldsymbol{z}_1^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y}))\, \boldsymbol{dy} +$$

$$\sum_{s=1}^{M} \sum_{i=1}^{\min\{k_s+1, c_s\}} \sum_{l,\boldsymbol{r}} \int_{\mathbb{R}^l} (\mu_s + \gamma_s) \times$$

$$\delta_{s-r_l} p(\boldsymbol{z}_2^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y}))\, \boldsymbol{dy} +$$

$$\sum_{s \in \mathcal{S}^-(z)} \lambda g(l_{sk_s}, \boldsymbol{r}_{sk_s}, \boldsymbol{y}_{sk_s})\, p(\boldsymbol{z}(\boldsymbol{z}, s)) +$$

$$\sum_{s \in \mathcal{S}_1(z)} \sum_{i=1}^{\min\{k_{s'}+1, c_{s'}\}} \mu_{s'} p(\boldsymbol{z}_1(\boldsymbol{z}, s, i)) +$$

$$\sum_{s \in \mathcal{S}_2(z)} \sum_{i=1}^{\min\{k_{s'}+1, c_{s'}\}} \mu_{s'} p(\boldsymbol{z}_2(\boldsymbol{z}, s, i)) +$$

$$\sum_{s \in \mathcal{S}_3(z)} \sum_{s'=1}^{M} \sum_{i=1}^{\min\{k_s+1, c_s\}} \sum_{l,\boldsymbol{r},n} \int_{\mathbb{R}^l} \gamma_{s'} \delta_{s'-r_n} \times$$

$$h(l_{sk_s}, \boldsymbol{r}_{sk_s}, \boldsymbol{y}_{sk_s} \mid l, \boldsymbol{r}, \boldsymbol{y}, n) \times$$

$$p(\boldsymbol{z}(\boldsymbol{z}, s, s', i, l, \boldsymbol{r}, \boldsymbol{y}, n))\, \boldsymbol{dy}. \qquad (5)$$

Let us verify whether the function $p(\boldsymbol{z})$ defined by the formulas (1)–(4) satisfies the equations (5). For this purpose, we rewrite the equations (5) as

$$\left[\lambda p(\boldsymbol{z}) - \sum_{s=1}^{M} \sum_{i=1}^{\min\{k_s+1, c_s\}} \sum_{l,\boldsymbol{r}} \int_{\mathbb{R}^l} \mu_s \times \right.$$

$$\delta_{s-r_l} p(\boldsymbol{z}_1^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y}))\, \boldsymbol{dy} -$$

$$\sum_{s=1}^{M} \sum_{i=1}^{\min\{k_s+1, c_s\}} \sum_{l,\boldsymbol{r}} \int_{\mathbb{R}^l} (\mu_s + \gamma_s) \times$$

$$\left. \delta_{s-r_l} p(\boldsymbol{z}_2^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y}))\, \boldsymbol{dy} \right] +$$

$$\sum_{s \in \mathcal{S}^-(z)} \left[\sum_{i=1}^{\min\{k_s, c_s\}} (\mu_s + \gamma_s)\, p(\boldsymbol{z}) - \right.$$

$$\left. \lambda g(l_{sk_s}, \boldsymbol{r}_{sk_s}, \boldsymbol{y}_{sk_s})\, p(\boldsymbol{z}(\boldsymbol{z}, s)) \right] +$$

$$\sum_{s \in \mathcal{S}_1(z)} \left[\sum_{i=1}^{\min\{k_s, c_s\}} (\mu_s + \gamma_s)\, p(\boldsymbol{z}) - \right.$$

$$\left. \sum_{i=1}^{\min\{k_{s'}+1, c_{s'}\}} \mu_{s'} p(\boldsymbol{z}_1(\boldsymbol{z}, s, i)) \right] +$$

$$\sum_{s \in \mathcal{S}_2(z)} \left[\sum_{i=1}^{\min\{k_s, c_s\}} (\mu_s + \gamma_s)\, p(\boldsymbol{z}) - \right.$$

$$\left. \sum_{i=1}^{\min\{k_{s'}+1, c_{s'}\}} \mu_{s'} p(\boldsymbol{z}_2(\boldsymbol{z}, s, i)) \right] +$$

$$\sum_{s \in \mathcal{S}_3(z)} \left[\sum_{i=1}^{\min\{k_s, c_s\}} (\mu_s + \gamma_s)\, p(\boldsymbol{z}) - \right.$$

$$\sum_{s'=1}^{M} \sum_{i=1}^{\min\{k_s+1, c_s\}} \sum_{l,\boldsymbol{r},n} \int_{\mathbb{R}^l} \gamma_{s'} \delta_{s'-r_n} \times$$

$$h(l_{sk_s}, \boldsymbol{r}_{sk_s}, \boldsymbol{y}_{sk_s} \mid l, \boldsymbol{r}, \boldsymbol{y}, n) \times$$

$$\left. p(\boldsymbol{z}(\boldsymbol{z}, s, s', i, l, \boldsymbol{r}, \boldsymbol{y}, n))\, \boldsymbol{dy} \right] = 0. \qquad (6)$$

Lemma formulated below defines some relation for the function $p(\boldsymbol{z})$ (1)–(4)

Lemma. *Function $p(\boldsymbol{z})$ defined by the formulas (1)–(4) satisfies $\boldsymbol{z}$ the following relations for all states $\boldsymbol{z}$:*

1. The following identity holds

$$\lambda p(\boldsymbol{z}) = \sum_{s=1}^{M} \sum_{i=1}^{\min\{k_s+1, c_s\}} \sum_{l,\boldsymbol{r}} \int_{\mathbb{R}^l} \mu_s \times$$

$$\delta_{s-r_l} p(\boldsymbol{z}_1^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y}))\, \boldsymbol{dy} +$$

$$\sum_{s=1}^{M} \sum_{i=1}^{\min\{k_s+1, c_s\}} \sum_{l,\boldsymbol{r}} \int_{\mathbb{R}^l} (\mu_s + \gamma_s) \times$$

$$\delta_{s-r_l} p(\boldsymbol{z}_2^+(\boldsymbol{z}, i, l, \boldsymbol{r}, \boldsymbol{y}))\, \boldsymbol{dy}. \qquad (7)$$

2. For any node $s \in \mathcal{S}^-(\boldsymbol{z})$

$$\sum_{i=1}^{\min\{k_s, c_s\}} (\mu_s + \gamma_s)\, p(\boldsymbol{z}) =$$

$$\lambda g(l_{sk_s}, \boldsymbol{r}_{sk_s}, \boldsymbol{y}_{sk_s})\, p(\boldsymbol{z}(\boldsymbol{z}, s)). \qquad (8)$$

3. For any node $s \in \mathcal{S}_1(\boldsymbol{z})$

$$\sum_{i=1}^{\min\{k_s, c_s\}} (\mu_s + \gamma_s)\, p(\boldsymbol{z}) =$$

$$\sum_{i=1}^{\min\{k_{s'}+1, c_{s'}\}} \mu_{s'} p(\boldsymbol{z}_1(\boldsymbol{z}, s, i)). \qquad (9)$$

4. For any node $s \in \mathcal{S}_2(\boldsymbol{z})$

$$\sum_{i=1}^{\min\{k_s, c_s\}} (\mu_s + \gamma_s)\, p(\boldsymbol{z}) =$$

$$\sum_{i=1}^{\min\{k_{s'}+1,c_{s'}\}} \mu_{s'} p(\boldsymbol{z}_2(\boldsymbol{z},s,i)). \qquad (10)$$

5. For any node $s \in \mathcal{S}_3(\boldsymbol{z})$

$$\sum_{i=1}^{\min\{k_s,c_s\}} (\mu_s+\gamma_s)\, p(\boldsymbol{z}) =$$

$$\sum_{s'=1}^{M} \sum_{i=1}^{\min\{k_s+1,c_s\}} \sum_{l,\boldsymbol{r},n} \int_{\mathbb{R}^l} \gamma_{s'} \delta_{s'-r_n} \times$$

$$h(l_{sk_s}, \boldsymbol{r}_{sk_s}, \boldsymbol{y}_{sk_s} \mid l, \boldsymbol{r}, \boldsymbol{y}, n) \times$$

$$p(\boldsymbol{z}(\boldsymbol{z},s,s',i,l,\boldsymbol{r},\boldsymbol{y},n))\, \boldsymbol{dy}. \qquad (11)$$

For the Proof of the lemma, it is sufficient to directly substitute into expressions (7)–(11), the value of the function $p(\boldsymbol{z})$ is determined by formulas (1)-(4).
The statement of the lemma shows that each expression enclosed in square brackets in formula (6) is equal to zero. Hence, function $p(\boldsymbol{z})$ really satisfies (5).
In order to conclude the proof of the theorem it is necessary to check up that function $p(\boldsymbol{z})$ satisfies the normalizing condition. For this, it is enough to show that each function $p_s(\boldsymbol{z}_s)$, $s = \overline{1,M}$, satisfies the normalizing condition. The proof is left to the reader.

Remark In the proof of the theorem we actually assumed that a "killed" customer cannot leave the system at once after his "killing", and should visit at least one node. It is possible to get rid of this assumption, considering new queueing network which differs from the investigated one only in the following: a "killed" customer which in investigated system left it immediately after "killing", in the new system will pass to some fictitious node, and then he will leave the network.

SOME COROLLARIES

Some important corollaries follow from this theorem. The proofs of these corollaries are omitted since they are easy to prove.

Corollary 1. *The marginal stationary distribution density of the process* $\boldsymbol{Z}_s(t)$ *is defined by formulas (2)–(4). The stationary state probability distribution of the process* $\boldsymbol{Z}(t)$ *has a product form, i.e., is representable as the product of stationary probability distribution of the process* $\boldsymbol{Z}_s(t)$.

Corollary 2. *The stationary distribution of the number of customers (regardless of their parameters) at a node* s *is of the form*

$$p_s(k) = p_s(0)\, d_s(k) \rho_s^k.$$

Corollary 3. *The stationary intensity* λ_s *of the input flow at node* s *is defined by the following formula*

$$\lambda_s = \sum_{l,\boldsymbol{r}} \int_{\mathbb{R}^l} \sum_{n=1}^{l} (g_n^*(l,\boldsymbol{r},\boldsymbol{y}) + h(l,\boldsymbol{r},\boldsymbol{y}))\, \delta_{s-r_n}\, \boldsymbol{dy}.$$

Corollary 4. *The stationary distribution of* k *not "killed" customers (regardless of their parameters) at a node* s *is of the form*

$$p_s^+(k) = p_s^+(0)\, d_s(k)\, (\rho_s^+)^k,$$

where

$$p_s^+(0) = \left(\sum_{i=0}^{c_s} \frac{(\rho_s^+)^i}{i!} + \frac{(\rho_s^+)^{c_s+1}}{c_s!\,(c_s-\rho_s^+)} \right)^{-1}.$$

Corollary 5. *The stationary distribution of* k *"killed" customers (regardless of their parameters) at a node* s *is of the form*

$$p_s^-(k) = p_s^-(0)\, d_s(k)\, (\rho_s^-)^k,$$

where

$$p_s^-(0) = \left(\sum_{i=0}^{c_s} \frac{(\rho_s^-)^i}{i!} + \frac{(\rho_s^-)^{c_s+1}}{c_s!\,(c_s-\rho_s^-)} \right)^{-1}.$$

Corollary 6. *The joint stationary distribution of* k *"not killed" and* l *"killed" customers (regardless of their parameters) at a node* s *is of the form*

$$p_s(k,l) = p_s(0)\, d_s(k+l) \binom{k+l}{k} (\rho_s^+)^k (\rho_s^-)^l.$$

SPECIAL CASES

1. Let always be $L^* = 0$. It corresponds to the case when a "killed" customer at once leaves the system. In this case, the proposed theorem is a consequence (for exponential networks) of the theorem proved in (Bocharov et al. 2004).

2. Let be

$$(L^*, \boldsymbol{R}^*, \boldsymbol{Y}^*) = (L-N, R_{N+1}, \ldots, R_L, Y_{N+1}, \ldots, Y_L).$$

This variant reflects the situation when a negative customer simply moves "instantly" a positive customer to a subsequent stage of service (simultaneously "killing" it if it is not yet "killed").
Besides, it is possible to consider another variant of this case: a negative customer "kills" a positive customer which, nevertheless, continues to be served in the same way, as well as "not killed", staying in a node s (including one where it has been "killed") during a random time distributed exponentially with average value μ_s. For this purpose it is necessary to introduce new parameters: $\mu_s^* = \mu_s^2/(\mu_s+\gamma_s)$ and $\gamma_s^* = \mu_s\gamma_s/(\mu_s+\gamma_s)$ where μ_s^* and γ_s^* are, respectively, the average value of the exponential distributions of customer service time in nodes and of negative customers flows entering nodes.

3. Let be

$$(L^*, \boldsymbol{R}^*, \boldsymbol{Y}^*) = (K, S_1, \ldots, S_K, U_1, \ldots, U_K).$$

Unlike case 1, in this case a "killed" customer does not leave the system at once but becomes a "stirring" one, creating additional loading for the network.

4. Let be

$$(L^*, \boldsymbol{R}^*, \boldsymbol{Y}^*) = (L-N+K, R_{N+1}, \ldots, R_L, S_1, \ldots, S_K, Y_{N+1}, \ldots, Y_L, U_1, \ldots, U_K).$$

Now a "killed" customer becomes a "stirring" one only when it passes all the way which was assigned to it as a positive customer.

5. Let RV L^* be independent of other parameters of the system and have a geometrical distribution with parameter $\mu_s/(\mu_s+\gamma_s)$ where s is the number of the node in which the positive customer is "killed", and let be

$$(L^*, \boldsymbol{R}^*, \boldsymbol{Y}^*) = (L^*, R_N, \ldots, R_N, Y_N, \ldots, Y_N).$$

In this case, we obtain a queueing system considered in (Bocharov et al. 2006).

CONCLUSION

In the present paper a new variant of exponential G-networks with dependent service and route change of "killed" customers is considered. In the new variant, a "killed" customer does not leave the network but continues to move through a network according to a new random route, inducing an additional loading. For such G-networks, the multidimensional stationary distribution of the network state probabilities is shown to be representable in product form. Unfortunately, authors could not insert in the present paper examples of numerical calculations due to the paper size limitation.

ACKNOWLEDGEMENTS

The research has been performed with the support of the Russian Foundation for Basic Research (grants 06-07-89056 and 05-07-90103).

REFERENCES

Bocharov P.P.; C. D'Apice; E.V. Gavrilov; and A.V. Pechinkin. 2004a. "Decomposition of Queueing Networks with Dependent Service and Negative Customers". *Automation and Remote Control*, vol.65, 86–103.

Bocharov P.P.; C. D'Apice; E.V. Gavrilov; and A.V. Pechinkin. 2004b. "Product form solution for G-networks with dependent service". *RAIRO Oper. Res.*, vol.38, 105–119.

Bocharov P.P.; C. D'Apice; E.V. Gavrilov; and A.V. Pechinkin. 2004c. "Exponential Queuing Network with Dependent Servicing, Negative Customers, and Modification of the Customer Type". *Automation and Remote Control*, vol.65, 1066–1088.

Bocharov P.P.; E.V. Gavrilov; and A.V. Pechinkin. 2004d. "Decomposition of G-networks with dependent service and completion of service of killed customers". *J. of Information Processes*, vol.4, 58–75 (in Russian).

Bocharov P.P.; C. D'Apice; and A.V. Pechinkin. 2006. "Product form solution for exponential G-networks with dependent service and completion of service of "killed" customers". *Computational Management Science*, No.3, 177–192.

Bocharov P.P. and V.M. Vishnevskii. 2003. "G-networks: development of the theory of multiplicative networks". *Automation and Remote Control*, vol.64, 714–739.

Cramer C. and E. Gelenbe. 2000. "Video quality and traffic QoS in learning-based subsampled and receiver-interpolated video sequences". *IEEE Journal on Selected Areas in Communications*, vol.18, 150–167.

Feng Y. and E. Gelenbe. 1999. "Adaptive object tracking and video compression". *Network and Information Systems Journal*, vol.1, 371–400.

Gelenbe E. 1989a. "Random neural networks with positive and negative signals and product form solution". *Neural Computation*, vol.1, 502–510.

Gelenbe E. 1989b. Reseaux stochastiques ouverts avec clients negatifs et positifs, et reseaux neuronaux". *Comptes-Rendus Acad. Sci. Paris II*, vol.309, 979–982.

Gelenbe E. 1990a. "Reseaux neuronaux aleatoires stables". *Comptes-Rendus Acad. Sci. II*, vol.310, 177–180.

Gelenbe E. 1990b. "Stable random neural networks". *Neural Computation*, vol.2, 239–247.

Gelenbe E. 1991. "Queuing networks with negative and positive customers". *Journal of Applied Probability*, vol.28, 656–663.

Gelenbe E.; P. Glynn; and K. Sigman. 1991. "Queues with negative arrivals". *Journal of Applied Probability*, vol.28, 245–250.

Gelenbe E.; E. Seref; and Z. Xu. 2001. "Simulation with learning agents". *Proceedings of the IEEE*, vol.89, 148–157.

Gelenbe E. and H. Shachnai. 2000. "On G-networks and resource allocation in multimedia systems". *European J. Opns. Res.*, vol.126, 308–318.

AUTHOR BIOGRAPHIES

CIRO D'APICE was born in Castellamare di Stabia, Italy and obtained PhD degrees in Mathematics in 1997. He is Director of the Department of Information Engineering and Applied Mathematics in University of Salerno. His research interests include: computer aided learning, queueing theory, hybrid systems, homogenization problems, spatial behaviour for dynamic problems, fluid-dynamic models for traffic networks and supply chains. His e-mail address is: dapice@diima.unisa.it.

ALEXANDER V. PECHINKIN was born in Moscow, Russia. He graduated from the Lomonosov Moscow State University in 1968. D. Sc. (Phys.-math.), Professor, author of more then 160 publications in the field of the probability theory and its applications. At present he is Principal Scientist of the Institute of Informatics Problems RAS. His e-mail address is: APechinkin@ipiran.ru.

SERGEY Ya. SHORGIN was born in Kirovograd, Ukraine (former USSR). He graduated from the Lomonosov Moscow State University in 1974. D. Sc. (Phys.-math.), Professor, author of more then 100 publications in the field of the probability theory and its applications. At present he is Deputy Director of the Institute of Informatics Problems RAS. His e-mail address is: SShorgin@ipiran.ru.

SESSION 5

Modelling and Analysis Techniques

MODELING AND ANALYSIS OF SIMULTANEOUS MULTITHREADING

W.M. Zuberek
Department of Computer Science
Memorial University
St.John's, Canada A1B 3X5
wlodek@cs.mun.ca

KEYWORDS
Simultaneous multithreading, instruction issuing, pipelined processors, timed Petri nets, performance analysis, event–driven simulation.

ABSTRACT
In simultaneous multithreading, several threads can issue instructions in each processor cycle. A simple and versatile timed Petri net model of simultaneous multithreading is proposed and is used to compare the performance of architectures with and without simultaneous multithreading. Performance results are obtained by event-driven simulation of net models and are verified by state–space–based analysis using combinations of modeling parameters for which the state space remains reasonably small.

INTRODUCTION

Continuous progress in manufacturing technologies results in the performance of microprocessors that has been steadily improving over the last decades, doubling every 18 months (the so called Moore's law [6]). At the same time, the capacity of memory chips has also been doubling every 18 months, but the performance has been improving less than 10% per year [10]. The latency gap between the processor and its memory doubles approximately every six years, and an increasing part of the processor's time is spent on waiting for the completion of memory operations [11]. Matching the performances of the processor and the memory is an increasingly difficult task. In effect, it is often the case that up to 60 % of execution cycles are spent waiting for the completion of memory accesses [8].

Techniques which tolerate long–latency memory accesses include out–of–order execution of instructions and instruction–level multithreading. The idea of out–of–order execution is to execute, during the waiting for the completion of a long–latency operation, instructions which (logically) follow the long–latency one, but which do not depend upon the result of this long–latency operation. Since out–of–order execution exploits instruction–level concurrency using the existing sequential instruction stream, it conveniently maintains code–base compatibility [7]. In effect, the instruction stream is dynamically decomposed into micro-threads, which are scheduled and synchronized at no cost in terms of executing additional instructions. Although this is desirable, speedups using out–of–order execution on superscalar pipelines are not so impressive, and it is difficult to obtain a speedup greater than 2 using 4 or 8-way superscalar issue [12]. Moreover, memory latencies are so long that out–of–order processors require very large instruction windows to tolerate them.

Although ultra–wide out-of-order superscalar processors were predicted as the architecture of one-billion-transistor chips [9], with a single 16 or 32-wide-issue processing core and huge branch predictors to sustain good instruction level parallelism, at present the industry is not moving toward the wide–issue superscalar model [1]. Design complexity and power efficiency direct the industry toward narrow–issue, high–frequency cores and multithreaded processors. According to [7]: "Clearly something is very wrong with the out–of–order approach to concurrency if this extravagant consumption of on–chip resources is only providing a practical limit on speedup of about 2."

Instruction–level multithreading [3], [4] tolerates long–latency memory accesses by switching to another thread (if it is available for execution) rather than waiting for the completion of the long–latency operation. If different threads are associated with different sets of processor registers, switching from one thread to another (called "context switching") can be done very efficiently [13].

In simultaneous multithreading [5], [7] several threads can issue instructions at the same time. If a processor contains more than one pipeline, or it contains several functional units, the instructions can be issued simultaneously; if there is only one pipeline, only one instruction can be issued in each processor cycle, but the (simultaneous) threads complement each other in the sense that whenever one thread cannot issue an instruc-

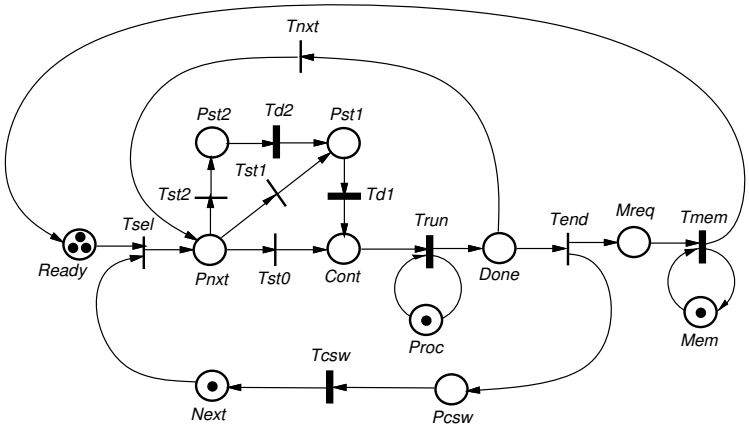


Fig.1. Petri net model of a multithreaded processor

tion (because of pipeline stalls or context switching), an instruction is issued from another thread, eliminating 'empty' instruction slots and increasing the overall performance of the processor.

The main objective of this paper is to study the performance of simultaneously multithreaded processors in order to determine how effective simultaneous multithreading can be. In particular, an indication is sought if simultaneous multithreading can overcome the out–of–order's "barrier" of the speedup (equal to 2). A timed Petri net [14] model of multithreaded processors at the instruction execution level is developed, and performance results for this model are obtained by event–driven simulation of the net model. Since the model is rather simple, simulation results can be verified (with respect to accuracy) by state–space–based performance analysis (for combinations of modeling parameters for which the state spaces remains reasonably small).

SIMULTANEOUS MULTITHREADING

A timed Petri net model of a simple multithreaded processor is shown in Fig.1 (as usually, timed transitions are represented by solid bars, and immediate ones, by thin bars). For simplicity, Fig.1 shows only one level of memory; this simplification is removed further in this section.

Ready is a pool of available threads; it is assumed that the number of of threads is constant and does not change during program execution (this assumption is motivated by steady–state considerations). If the processor is idle (place *Next* is marked), one of available threads is selected for execution (transition *Tsel*). *Pnxt* is a free-choice place with three possible outcomes: *Tst0* (with the choice probability p_{s0}) represents issuing an instruction without any further delay; *Tst1* (with the choice probability p_{s1}) represents a single-cycle pipeline stall (modeled by *Td1*), and *Tst2* (with the choice probability p_{s2}) represents a two–cycle pipeline stall (*Td2* and then *Td1*); other pipeline stalls could be represented in a similar way, if needed. *Cont*, if marked, indicates that an instruction is ready to be issued to the execution pipeline. Instruction execution is modeled by transition *Trun* which represents the first stage of the execution pipeline. It is assumed that once the instruction enters the pipeline, it will progress through the stages and, eventually, leave the pipeline; since these pipeline implementation details are not important for performance analysis of the processor, they are not represented here.

Done is another free-choice place which determines if the current instruction performs a long–latency access to memory or not. If the current instruction is a non–long–latency one, *Tnxt* occurs (with the corresponding probability), and another instruction is fetched for issuing. If long–latency operation is detected in the issued instruction, *Tend* initiates two concurrent actions: (i) context switching performed by enabling an occurrence of *Tcsw*, after which a new thread is selected for execution (if it is available), and (ii) a memory access request is entered into *Mreq*, the memory queue, and after accessing the memory (transition *Tmem*), the thread, suspended for the duration of memory access, becomes "ready" again and joins the pool of threads *Ready*. *Tmem* will typically represent a cache miss (with all its consequences); cache hits (at the first level cache memory) are not considered long-latency operations.

The choice probability associated with *Tend* determines the runlength of a thread, ℓ_t, i.e., the average number of instructions between two consecutive long–latency operations; if this choice probability is equal to 0.1, the runlength is equal to 10, if it is equal to 0.2, the runlength is 5, and so on.

Proc, which is connected to *Trun*, controls the num-

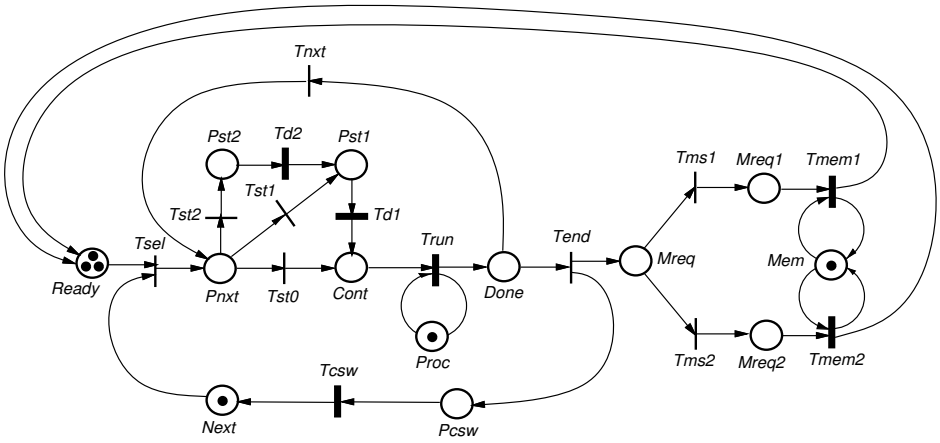


Fig.2. Petri net model of a multithreaded processor with a two–level memory

ber of pipelines. If the processor contains just one instruction execution pipeline, the initial marking assigns a single token to *Proc* as only one instruction can be issued in each processor cycle. In order to model a processor with two (identical) pipelines, two initial tokens are needed in *Proc*, and so on.

The number of memory ports, i.e., the number of simultaneous accesses to memory, is controlled by the initial marking of *Mem*; for a single port memory, the initial marking assigns just a single token to *Mem*, for dual-port memory, two tokens are assigned to *Mem*, and so on.

In a similar way, the number of simultaneous threads (or instruction issue units) is controlled by the initial marking of *Next*.

Memory hierarchy can be incorporated into the model shown in Fig.1 by refining the representation of memory. In particular, levels of memory hierarchy can be introduced by replacing the subnet *Tmem–Mem* by a number of subnets, each subnet for one level of the hierarchy, and adding a free–choice structure which randomly selects the submodel according to probabilities describing the use of the hierarchical memory. Such a refinement, for two levels of memory (in addition to the first-level cache), is shown in Fig.2, where *Mreq* is a free–choice place selecting either level–1 (submodel *Mem–Tmem1*) or level–2 (submodel *Mem–Tmem2*). More levels of memory can be easily added similarly, if needed.

The effects of memory hierarchy can be compared with a uniform, non–hierarchical memory by selecting the parameters in such a way that the average access time of the hierarchical model (Fig.2) is equal to the access time of the non–hierarchical model (Fig.1).

Processors with different numbers of instruction issue units and instruction execution pipelines can be described by a pair of numbers, the first number denoting the number of instruction issue units, and the second – the number of instruction execution pipelines. In this sense a 3-2 processor is a (multithreaded) processor with 3 instruction issue units and 2 instruction execution pipelines.

For convenience, all temporal properties are expressed in processor cycles, so, the occurrence times of *Trun*, *Td1* and *Td2* are all equal to 1 (processor cycle), the occurrence time of *Tcsw* is equal to the number of processor cycles needed for a context switch (which is equal to 1 for many of the following performance analyses), and the occurrence time of *Tmem* is the average number of processor cycles needed for a long–latency access to memory.

The main modeling parameters and their typical values are summarized in Tab.1.

Table 1: Simultaneous multithreading modeling parameters and their typical values

symbol	*parameter*	*value*
n_t	number of available threads	1,...,10
n_p	number of execution pipelines	1,2,...
n_s	number of simultaneous threads	1,2,3,...
ℓ_t	thread runlength	10
t_{cs}	context switching time	1,3
t_m	average memory access time	5
p_{s1}	prob. of one–cycle pipeline stall	0.2
p_{s2}	prob. of two–cycle pipeline stall	0.1

PERFORMANCE RESULTS

The utilization of the processor and memory, as a function of the number of available threads, for a 1-1

processor (i.e., a processor with a single instruction issue unit and a single instruction execution pipeline) is shown in Fig.3.

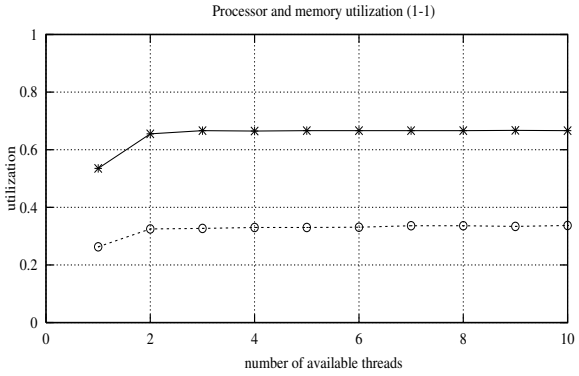


Fig.3. Processor (-x-) and memory (-o-) utilization for a 1-1 processor; $l_t = 10$, $t_m = 5$, $t_{cs} = 1$

The asymptotic value of the utilization can be estimated from the (average) number of empty instruction issuing slots. Since the probability of a single–cycle stall is 0.2, and probability of a two–cycle stall is 0.1, on average 40 % of issuing slots remain empty because of pipeline stalls. Moreover, there is an overhead of $t_{cs} = 1$ slot for context switching. The asymptotic utilization is thus 10/15 = 0.667, which corresponds very well with Fig.3.

The utilization of the processor can be improved by introducing a second (simultaneous) thread which issues its instructions in the unused slots. Fig.4 shows the utilization of the processor and memory for a 2-1 processor, i.e., a processor with two (simultaneous) threads (or two instruction issue units) and a single pipeline.

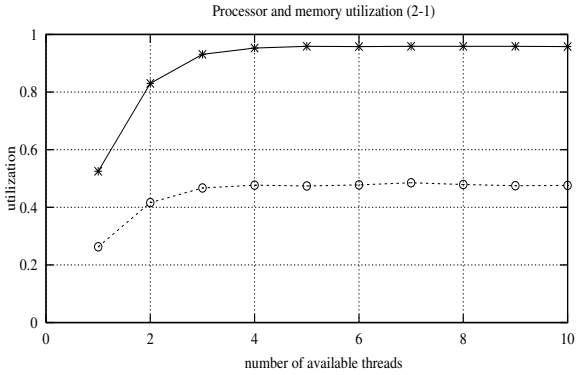


Fig.4. Processor (-x-) and memory (-o-) utilization for a 2-1 processor; $l_t = 10$, $t_m = 5$, $t_{cs} = 1$

The utilization of the processor is improved by almost 50 %, and is within a few percent from its upper bound of 1.00 (or 100 %).

A more realistic model of memory, that captures the idea of a two–level hierarchy, is shown in Fig.2. In order to compare the results of this model with Fig.3 and Fig.4, the parameters of the two–level memory are chosen in such a way that the average memory access is equal to the memory access time in Fig.1 (where $t_m = 5$). Let the two levels of memory have access times equal to 4 and 20, respectively; then the choice probabilities are equal to 15/16 and 1/16 for level–1 and level–2, respectively, and the average access time is:

$$4 * \frac{15}{16} + 20 * \frac{1}{16} = 5.$$

The results for a 1-1 processor with a two–level memory are shown in Fig.5, and for a 2-1 processor in Fig.6.

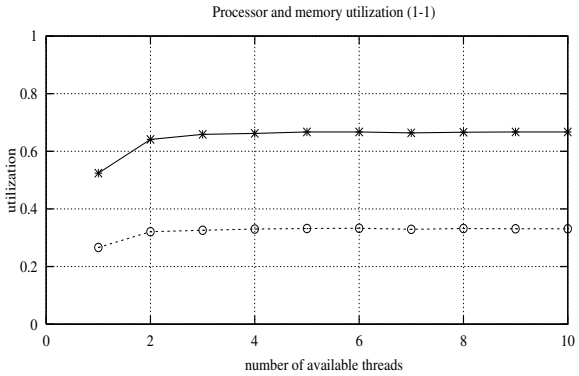


Fig.5. Processor (-x-) and memory (-o-) utilization for a 1-1 processor with 2-level memory; $l_t = 10$, $t_m = 4 + 20$, $t_{cs} = 1$

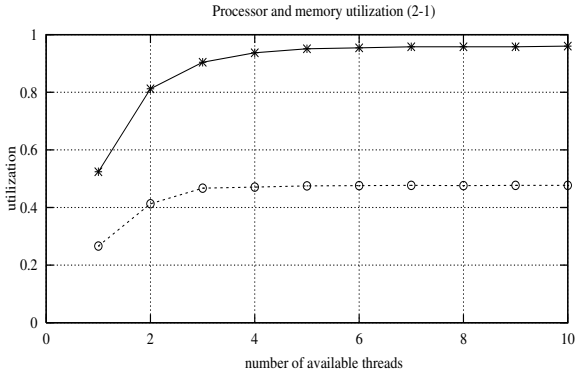


Fig.6. Processor (-x-) and memory (-o-) utilization for a 2-1 processor with 2-level memory; $l_t = 10$, $t_m = 4 + 20$, $t_{cs} = 1$

The results in Fig.5 and Fig.6 are practically the same as in Fig.3 and Fig.4. This is the reason that the remain-

ing results are shown for (equivalent) one-level memory models; the multiple levels of memory hierarchy apparently have no significant effect on the performance results.

The effects of simultaneous multithreading in a more complex processor, e.g., a processor with two instruction issue units and two instruction execution pipelines, i.e., a 2-2 processor, can be obtained in a very similar way. The utilization of the processor (shown as the sum of the utilizations of both pipelines, with the values ranging from 0 to 2), is shown in Fig.7.

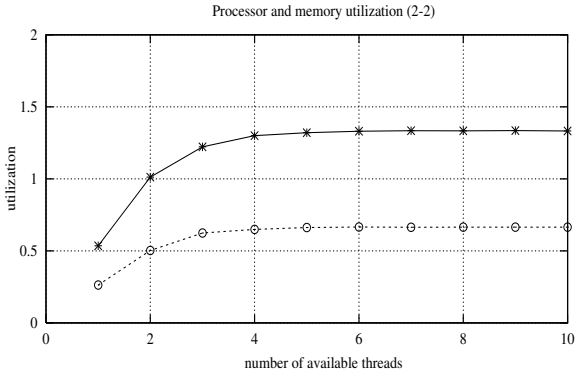


Fig.7. Processor (-x-) and memory (-o-) utilization for a 2-2 processor; $l_t = 10$, $t_m = 5$, $t_{cs} = 1$

When another instruction issue unit is added, the utilization increases by about 40 %, as shown in Fig.8.

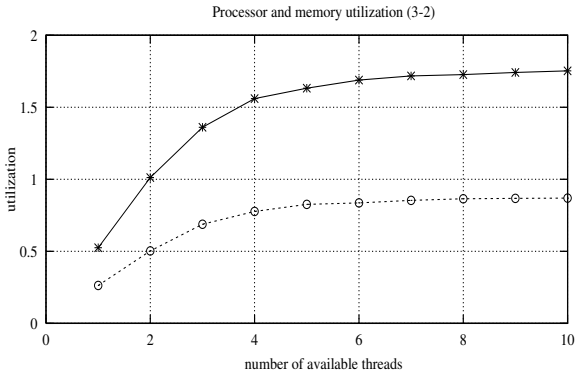


Fig.8. Processor (-x-) and memory (-o-) utilization for a 3-2 processor; $l_t = 10$, $t_m = 5$, $t_{cs} = 1$

Further increase of the number of the simultaneous threads (in a processor with 2 pipelines) can provide only small improvements of the performance because the utilizations of both, the processor and the memory, are quite close to their limits. The performance of the system can be improved by increasing the number of pipelines, but then the memory becomes the system bottleneck, so its performance also needs to be improved, for example, by introducing dual ports (which allow to handle two accesses at the same time). The performance of a 5-3 processor with a dual-port memory is shown in Fig.9 (the utilization of the processor is the sum of utilizations of its 3 pipelines, so it ranges from 0 to 3).

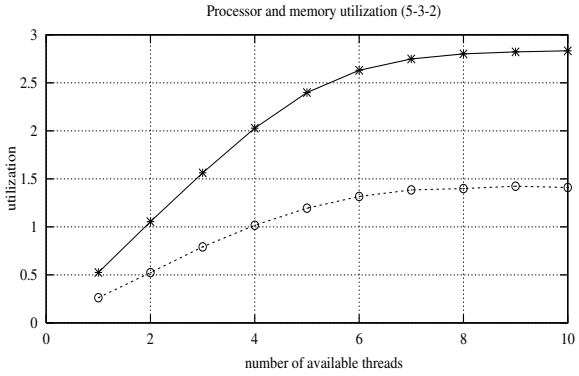


Fig.9. Processor (-x-) and memory (-o-) utilization for a 5-3 processor with dual–port memory; $l_t = 10$, $t_m = 5||2$, $t_{cs} = 1$

Fig.9 shows that for 3 pipelines and 5 simultaneous threads, the number of available threads greater than 6 provides the speedup that is almost equal to 3.

System bottlenecks can be identified by comparing service demands for different components of the system (in this case, the memory and the pipelines); the component with the maximum service demand is the bottleneck because it is the first component to reach its utilization limit and to prevent any increase of the overall performance. For a single runlegth (of all simultaneous threads) the total service demand for memory is equal to $n_s * t_m$, while the service demand for each pipeline (assuming an ideal, uniform distribution of load over the pipelines) is equal to $n_s * \ell_t / n_p$. For a 4-2 processor, the service demands are equal (such a system is usually called "balanced"), so the utilizations of both, the processor and the memory, tend to their limits in a "synchronous" way. For a 5-3 processor with a dual-port memory, the service demand for the pipelines is greater than the service demand for memory, so the number of pipelines could be increased (by one pipeline); for more than 4 pipelines, the memory again becomes the bottleneck.

CONCLUDING REMARKS

Simultaneous multithreading discussed in this paper is a means to increase the performance of processors

by tolerating long–latency operations. Since the long–latency operations seem to be playing increasingly important role in modern microprocessors, so is simultaneous multithreading. Its implementation as well as the required hardware resources are much simpler than in the case of out–of–order approach, and the resulting speedup scales well with the number of simultaneous threads. The main challenge of simultaneous multithreading is to balance the system by maintaining the right relationship between the number of simultaneous threads and the performance of the memory hierarchy.

All presented results indicate that the number of available threads, required for improved performance of the processor, is quite small, and is typically greater by 2 or 3 threads than the number of simultaneous threads. Performance improvement due to a larger number of available threads is rather insignificant.

The presented models of multithreaded processors are quite simple, and for small values of modeling parameters (n_t, n_p, n_s) can be analyzed by the explorations of the state space. The following table compares some results for the 1-1 processor:

n_t	*number of states*	*analytical utilization*	*simulated utilization*
1	11	0.526	0.535
2	57	0.656	0.655
3	107	0.666	0.666
4	157	0.666	0.666
5	207	0.667	0.666

For the 3-2 processor, the comparison is:

n_t	*number of states*	*analytical utilization*	*simulated utilization*
1	11	0.525	0.526
2	86	1.012	1.012
3	309	1.361	1.367
4	660	1.560	1.553
5	1,154	1.632	1.639

The results obtained by simulation of net models are very similar to the analytical results obtained from the analysis of states and state transitions. It should not be surprising that for more complex models the state space can become quite large, and then the event-driven simulation remains the best approach to analysis of net models.

ACKNOWLEDGEMENT

The Natural Sciences and Engineering Research Council of Canada partially supported this research through grant RGPIN-8222.

REFERENCES

[1] Burger, D., Goodman, J.R., “Billion–transistor architectures: there and back again”; IEEE Computer, vol.37, no.3, pp.22-28, 2004.

[2] Burger, D., Goodman, J.R., “Billion–transistor architectures”; IEEE Computer, vol.30, no.9, pp.46-49, 1997.

[3] Byrd, G.T., Holliday, M.A., “Multithreaded processor architecture”; IEEE Spectrum, vol.32, no.8, pp.38-46, 1995.

[4] Dennis, J.B., Gao, G.R., “Multithreaded architectures: principles, projects, and issues”; in “Multithreaded Computer Architecture: a Summary of the State of the Art”, pp.1-72, Kluwer Academic 1994.

[5] Eggers, S.J., Emer, J.S., Levy, H.M., Lo, J.L., Stamm, R.L., Tullsen, D.M., “Simultaneous multithreading: a foundation for next-generation processors”, IEEE Micro, vol.17, no.5, pp.12-19, 1997.

[6] Hamilton, S., “Taking Moore’s law into the next century”; IEEE Computer, vol.32, no.1, pp.43-48, 1999.

[7] Jesshope, C., “Multithreaded microprocessors – evolution or revolution”; in “Advances in Computer Systems Architecture” (LNCS 2823), pp.21-45, 2003.

[8] Mutlu, O., Stark, J., Wilkerson, C., Patt, Y.N., “Runahead execution: an effective alternative to large instruction windows”; IEEE Micro, vol.23, no.6, pp.20-25, 2003.

[9] Patt, Y.N., Patel, A., Friendly, D.H., Stark, J., “One billion transistors, one uniprocessors, one chip”; IEEE Computer, vol.30, no.9, pp.51-58, 1997.

[10] Patterson, D.A., Hennessy, J.L., “Computer architecture – a qualitative approach”, Morgan Kaufman 1996.

[11] Sinharoy B., “Optimized thread creation for processor multithreading”; The Computer Journal, vol.40, no.6, pp.388-400, 1997.

[12] Tseng, J., Asanovic, K., “Banked multiport register files for high–frequency superscalar microprocessor”; Proc. 30-th Int. Annual Symp. on Computer Architecture, pp.62-71, 2003.

[13] Ungerer, T., Robic, G., Silc, J., “Multithreaded processors”; The Computer Journal, vol.43, no.3, pp.320-348, 2002.

[14] Zuberek, W.M., “Timed Petri nets – definitions, properties and applications”; Microelectronics and Reliability (Special Issue on Petri Nets and Related Graph Models), vol.31, no.4, pp.627-644, 1991.

BIOGRAPHICAL NOTE

W.M. ZUBEREK received M.Sc. degree in electronic engineering and Ph.D. and D.Sc. degrees in computer science, all from Warsaw University of Technology. Currently he is a Professor in the Department of Computer Science, Memorial University in St.John’s, Canada.

ON REPRESENTING MULTICLASS M/M/k QUEUES BY GENERALIZED STOCHASTIC PETRI NETS

Simonetta Balsamo
Andrea Marin
Computer Science Department
Università Ca' Foscari di Venezia
Via Torino 155, 30172 Venezia Mestre, Italy
{balsamo,marin}@dsi.unive.it

KEYWORDS

GSPN, multiclass queuing systems.

ABSTRACT

In this paper we study the relations between multiclass BCMP-like service stations and generalized stochastic Petri nets (GSPN). Representing queuing discipline with GSPN models is not easy. We focus on representing multi-class queuing systems with different queuing disciplines by defining appropriate finite GSPN models. Note that queuing discipline in general affects performance measures in multi-class systems. For example, BCMP-like service centers with First Come First Served (FCFS) and with Last Come First Served with Preemptive Resume (LCFSPr) have a (different) product-form solution under different hypotheses. We define structurally finite GSPNs equivalent to the multi-class M/M/k queuing system with FCFS, LCFSPR, Processor Sharing (PS) and Infinite Servers (IS). Equivalence holds in terms of steady state probability function and average performance measure. The main idea is to define a finite GSPN model that simulates the behavior of a given queue discipline with some appropriate random choice. Moreover, we prove that the combination of the introduced equivalent models has a closed-form steady state probability by the $M \Rightarrow M$ property. We consider queuing systems with both a single server with load dependent service rate, and multiple servers with constant service rate.

1 INTRODUCTION

Queuing theory and (Generalized) Stochastic Petri Nets are important classes of stochastic models used to evaluate system performances. Queuing systems have been widely applied to represent resource contention systems where a set of customers competes for resource usage. Queuing networks (QN) extend and combine various queuing systems to represent more complex systems. Generalized Stochastic Petri nets (GSPN) can be naturally used to represent systems with synchronization and concurrency and to perform both qualitative and quantitative analysis. Under some exponential and independence assumptions these models can be studied through the associated continuous-time Markov chain (CTMC). In order to improve the algorithmic analysis efficiency for restricted classes of QN and SPNs, product-form theorems have been introduced. BCMP theorem (Baskett, Chandy, Muntz, & Palacios, 1975) is the main result for QN while the papers of Henderson et al. (Henderson, Lucic, & Taylor, 1989; Coleman, Henderson, & Taylor, 1996) present the main results for SPN and they have been extended to GSPN in the paper (Balbo, Bruell, & Sereno, 2002).

In this paper we investigate the relations between the BCMP queuing centers and GSPN models. The problem is trivial when all the customers in the QN are statistically identical (single class QN) thanks to the insensitivity property, i.e., the performance measures depend on the average of the service time and not on the queuing disciplines. Some difficulties arise when the QN customers are clustered in different classes which have different behaviors for each service center. In (Baskett et al., 1975) it is proved that in this case the queuing disciplines influence the QN performance measures. Thus if we try to represent a QN station by a GSPN we expect to have different models according to the queueing disciplines. Most of the works in this field devote a little attention to this problem. To the best of our knowledge, representing scheduling disciplines in multiclass models with finite GSPNs is still an open problem. In (Vernon, Zahorjan, & Lazowska, 1987) the authors introduce a comparison between QN models and SPN models based on the representation of multiclass features by colored Petri nets. However the differences between different scheduling disciplines are not analyzed. Balbo et al. in (Balbo, Bruell, & Ghanta, 1998) combine GSPN and product-form QN by replacing subsystem in a low-level model with their flow equivalent models. Still little attention is devoted to scheduling disciplines. In (Balbo, Bruell, & Sereno, 2003) the authors observe how they can map each service station of a BCMP QN to a complex GSPN which does not hold the GSPN product-form conditions of (Balbo et al., 2002). The GSPN model depends on the scheduling disciplines but it has an infinite number of places and transitions for the FCFS and LCFSPR stations. Then they give a finite and remarkably compact representation by a GSPN equivalent to the detailed model. The compact representation holds the product-form conditions for GSPN showed in (Balbo et al., 2002) but it does not distinguish different queuing disciplines by mapping everything in the PS discipline.

Print 0-9553018-4-X, CD 0-9553018-3-1

In this paper we present an equivalence result between two types of stochastic models. We propose a finite GSPN representation of a set of queuing systems with various scheduling disciplines. According to the BCMP-type service centers we analyze First Come First Served (FCFS), Last Come First Served with preemptive resume (LCFSPR), Processor Sharing (PS) and Infinite Servers (IS) scheduling disciplines. The main idea behind these results is a probabilistic model of the queue, i.e., all the customers of the same class wait in the same place and when a server becomes free the customer which gets the service is chosen in a probabilistic way similarly to what happens with the random queuing discipline. In the LCFSPR discipline, we also choose probabilistically the customer that looses the server when a new customer arrives to the system.

The advantage of having a finite representation which is different for the various scheduling disciplines is twofold: first it makes the analysis easier and it can be used for practical purposes. Second it does not require the definition of new semantic for the GSPN according to the queuing disciplines. Thus existing analysis or simulation tools can be used with the GSPN nets defined in this work. The proposed results are interesting because they allow the representation of an M/M/k queue with various queuing disciplines by a compact GSPN, which is equivalent to the queuing system in term of steady state queue length distribution. A practical consequence can be that it can extend a GSPN simulator or analyzer for analyzing multiclass queue systems. The only requirement is that the tool is able to model state-dependent firing rates for timed transitions and state-dependent weights for immediate transitions. There is no need to support the colored model extension to represent different classes.

The paper is structured as follows. Section 2 briefly reviews the GSPN models recalling formalism we chose, Section 3 reviews some results of the queuing systems theory used later in the paper. In Sections 4, 5 we introduce the GSPNs respectively equivalent to the FCFS and LCFSPR multiclass M/M/k queue. Section 6 discuss the GSPN models for both PS scheduling and IS systems. Section 7 uses $M \Rightarrow M$ property to state some considerations on the form of the steady state probability for some combinations of the GSPN models. Finally, Section 8 provides some concluding remarks.

2 GENERALIZED STOCHASTIC PETRI NETS

In this section we briefly recall the Generalized Stochastic Petri Nets (GSPN). We consider the notation for GSPN introduced in (Marsan, Balbo, Conte, Donatelli, & Franceschinis, 1995). In order to allow marking dependent probabilities for solving conflicts among immediate transitions we use the techniques discussed in (Chiola, Marsan, Balbo, & Conte, 1993). Let us define a marked Stochastic Petri Net which consists of a 8-tuple as follows:

$$GSPN = (\mathcal{P}, \mathcal{T}, I(\cdot,\cdot), O(\cdot,\cdot), H(\cdot,\cdot), \Pi(\cdot), w(\cdot,\cdot), \mathbf{m_0})$$

where:

- $\mathcal{P} = \{P_1, \ldots, P_M\}$ is the set of M places,
- $\mathcal{T} = \{t_1, \ldots, t_N\}$ is the set of N transitions (both immediate and timed),
- $I(t_i, p_j) : \mathcal{T} \times \mathcal{P} \rightarrow \mathbb{N}$ is the input function, $1 \leq i \leq N$, $1 \leq j \leq M$,
- $O(t_i, p_j) : \mathcal{T} \times \mathcal{P} \rightarrow \mathbb{N}$ is the output function, $1 \leq i \leq N$, $1 \leq j \leq M$,
- $H(t_i, p_j) : \mathcal{T} \times \mathcal{P} \rightarrow \mathbb{N}$ is the inhibition function, $1 \leq i \leq N$, $1 \leq j \leq M$,
- $\Pi(t_i) : \mathcal{T} \rightarrow \mathbf{N}$ is a function that specifies the priority of transition t_i, $1 \leq i \leq N$,
- $\mathbf{m} \in \mathbb{N}^M$ denotes a marking or state of the net, where m_i represents the number of tokens in place P_i, $1 \leq i \leq N$,
- $w(t_i, \mathbf{m}) : \mathcal{T} \times \mathbb{N}^M \rightarrow \mathbb{R}$ is the function which specifies for each timed transition t_i and each marking $\mathbf{m}$ a state dependent firing rate, and for immediate transitions a state dependent weight,
- $\mathbf{m_0} \in \mathbf{N}^M$ represents the initial state of the GSPN, i.e. the number of tokens in each place at the initial state.

We consider ordinary nets, i.e., functions I, O and H take values in $\{0, 1\}$. For each transition t_i let us define the input vector $\mathbf{I}(t_i)$, the output vector $\mathbf{O}(t_i)$ and the inhibition vector $\mathbf{H}(t_i)$ as follows: $\mathbf{I}(t_i) = (i_1, \ldots, i_M)$ where $i_j = I(t_i, P_j)$, $\mathbf{O}(t_i) = (o_1, \ldots, o_M)$ where $o_j = O(t_i, P_j)$ and $\mathbf{H}(t_i) = (h_1, \ldots, h_M)$ where $h_j = H(t_i, P_j)$. Function $\Pi(t_i)$ associates a priority to transition t_i. If $\Pi(t_i) = 0$ then t_i is a timed transition, i.e., it fires after an exponentially distributed firing time with mean $1/w(t_i, \mathbf{m})$, where $\mathbf{m}$ is the marking of the net. If $\Pi(t_i) > 0$ then t_i is an immediate transition and its firing time is zero. We say that transition t_a is enabled by marking $\mathbf{m}$ if $m_i \geq I(t_a, p_i)$ and $m_i < H(t_a, p_i)$ for each $i = 1, \ldots, M$ and no other transition of higher priority is enabled. We consider just two priority levels, 0 and 1. Hence when an immediate transition is enabled all the timed ones are disabled. The firing of transition t_i changes the state of the net from $\mathbf{m}$ to $\mathbf{m} - \mathbf{I}(t_i) + \mathbf{O}(t_i)$. The reachability set $RS(\mathbf{m_0})$ of the net is defined as the set of all markings that can be reached in zero or more firings from $\mathbf{m_0}$. We say that marking $\mathbf{m}$ is tangible if it enables only timed transitions and it is vanishing otherwise. For a vanishing marking $\mathbf{m}$ let $\mathcal{T}_\alpha$ be the set of enabled immediate transitions. Then the firing probability for any transition $t_i \in \mathcal{T}_\alpha$ and any state $\mathbf{m}$ is denoted by $p(t_i, \mathbf{m})$ and it is defined as follows:

$$p(t_i, \mathbf{m}) = \frac{w(t_i, \mathbf{m})}{\sum_{t_j \in \mathcal{T}_\alpha} w(t_j, \mathbf{m})}. \qquad (1)$$

Given a tangible marking $\mathbf{m}$ the transition with the lowest associated stochastic time fires.

A GSPN is represented by a graph with the following conventions: timed transitions are white filled boxes, immediate transitions are black filled boxes, places are circles, if $I(t_i, p_j) > 0$ we draw an arrow from p_j to t_i labelled with $I(t_i, p_j)$, if $O(t_i, p_j) > 0$ we draw an arrow from t_i to p_j labelled with $O(t_i, p_j)$, if $H(t_i, p_j) > 0$ we draw an circle ending line from p_j to t_i labelled with the value of $H(t_i, p_j)$, the marking $\mathbf{m}$ is represented by a set of m_j filled circles representing the tokens in place p_j for each $j = 1, \ldots, M$.

For ordinary nets we do not use labels for the arrows.

GSPN analysis consists in finding the steady state probability for each tangible marking of the reachability set. Some analysis techniques are presented in (Marsan et al., 1995). Under general assumptions, the stochastic process generated by the dynamic behavior of a standard SPN is a CTMC process. Mean state sojourn times are computed from the mean transition delays of the net. For GSPNs the distribution of the sojourn time in any marking can be expressed as a negative exponential and deterministically zero distributions for tangible and vanishing markings, respectively. Thus the marking process can be studied as a semi-Markov random process.

The GSPN models introduced in this paper present marking processes which allow us to easily reduce the semi-Markov process to a CTMC. In fact whenever a vanishing marking is reached, the next marking is tangible. Thus we can simply obtain a CTMC whose states are the tangible states of the original process and the transition rates are computed weighting the transitions rates of the original process with the firing probabilities of the immediate transitions.

Finally let us introduce some other notations: let $\mathbf{e_i}$ be an M-dimensional vector with all zero components but the i-th which is 1. We use the lower case t to name immediate transitions, the upper case T to name timed transitions, $\tilde{t}$ to name a generic timed or immediate transition.

3 SINGLE QUEUING SYSTEMS WITH DIFFERENT CLASSES OF CUSTOMERS

In this section we briefly recall single queuing systems with different classes of customers classifying them on the number of servers and scheduling disciplines. Let us consider an open queuing system with external arrivals, a queue, a set of identical servers and a set of R customer classes. Customers of class r arrive at the system according to a Poisson process with rate λ_r and require an exponentially distributed random service time with parameter μ_r, $r = 1, \ldots, R$. The system has a set of independent servers, possibly infinite.

We consider the following disciplines: First Come First Server (FCFS), Last Come First Server with Preemptive Resume (LCFSPR), Processor Sharing (PS). Let's start by considering a multiclass PS queuing system with a single server with load dependent service rate. Following the BCMP (Baskett et al., 1975) conventions, assume that the service rate can be expressed by a combination of a capacity function $x(n)$ depending on the total number of customers n at the station, and a class dependent capacity function $y_r(n_r)$, where n_r is the number of customers of class r at the station and a constant class dependent service rate μ_r. So the effective service rate for a customer of class r is given by the product $x(n)y_r(n_r)\mu_r$. Note that $x(1) = y_r(1) = 1$. Under stability conditions, the steady state probability of this service center is given by:

$$\pi'(\mathbf{n}) = \pi'_0 \frac{n!}{\prod_{i=1}^{R} n_i!} \prod_{i=1}^{R} \lambda_i^{n_i} \prod_{b=1}^{n} \frac{1}{x(b)} \prod_{r=1}^{R} \Big[\Big(\frac{1}{\mu_r} \Big)^{n_r} \prod_{a=1}^{n_r} \frac{1}{y_r(a)} \Big]. \quad (2)$$

Formula (2) holds also for single server LCFSPR with load dependent service rate. In order to obtain the steady state probabilities for M/M/k mutliclass system it suffices to set appropriate capacity function. An LCFSPR or PS center with k load independent servers requires to set $x(n) = \frac{\min(n,k)}{n}$ and $y_r(n_r) = n_r$.

Formula (2) still holds for FCFS scheduling discipline if the service rate is class independent, i.e., $\mu_r = \mu$ and $y_r(n_r) = 1$ for $1 \leq r \leq R$ and $n_r \geq 1$. In order to study the M/M/k/FCFS queuing system we have to set $x(n) = \min\{n, k\}$, and formula (2) becomes:

$$\pi'(\mathbf{n}) = \pi'_0 \frac{n!}{\prod_{i=1}^{R} n_i!} \prod_{i=1}^{R} \lambda_i^{n_i} \prod_{a=1}^{n} \frac{1}{\mu(a)}, \quad (3)$$

where $\mu(a) = x(a)\mu$.

4 REPRESENTING M/M/k/FCFS QUEUE BY GSPN

In this section we define a GSPN that represents an R-multiclass M/M/k/FCFS queue. Then we prove that the GSPN model is equivalent to the queuing system in terms of the steady state probability. Given the M/M/k/FCFS models defined as in Section 3 let us define the model called *GSPN-1.*

Definition 1 (GSPN-1). *According to GSPN definition given in Section 2:*

- $\mathcal{P} = \mathcal{P}_q \cup \mathcal{P}_s \cup \{P_{2R+1}\}$ *with* $\mathcal{P}_q = \{P_1, \ldots, P_R\}$ *and* $\mathcal{P}_s = \{P_{R+1}, \ldots, P_{2R}\}$,
- $\mathcal{T} = \mathcal{T}_w \cup \mathcal{T}_q$ *where* $\mathcal{T}_q = \{t_1, \ldots, t_R\}$ *and* $\mathcal{T}_w = \{T_{R+1}, \ldots, T_{2R}\}$,

– *function Π defined as follows:*

$$\Pi(\tilde{t_i}) = \begin{cases} 0 & \text{if } R+1 \leq i \leq 2R \\ 1 & \text{if } 1 \leq i \leq R \end{cases},$$

– *input and output vectors for transition t_i, $1 \leq i \leq R$:* $\mathbf{I}(t_i) = \mathbf{e_i} + \mathbf{e_{2R+i}}$ *and* $\mathbf{O}(t_i) = \mathbf{e_{R+i}}$. *Input and output vector for transition T_{R+i}:* $\mathbf{I}(T_{R+i}) = \mathbf{e_{R+i}}$ *and* $\mathbf{O}(T_{R+i}) = \mathbf{e_{2R+1}}$,
– $\mathbf{H}(t_i) = (0, \ldots, 0)$ *for all* $t_i \in \mathcal{T}$,
– $w(T_{R+i}, \mathbf{m}) = m_{R+i}\mu$ *for* $1 \leq i \leq R$ *and* $w(t_i, \mathbf{m}) = m_i$ *for* $1 \leq i \leq R$,
– $\mathbf{m_0} = (0, \ldots, 0, k)$.

Tokens arrive to places P_i, $1 \leq i \leq R$ according to Poisson stochastic processes.

Figure 1 illustrates the graphical representation of GSPN-1 model where $t_1, \ldots, t_R$ are immediate transitions and $T_{R+1}, \ldots, T_{2R}$ are exponential transitions.

Let $\mathbf{m}$ be a valid vanishing state of the GSPN-1, and let $\mathcal{T}_a \subseteq \mathcal{T}_q$ be the set of immediate transitions enabled by $\mathbf{m}$, then the probability of firing of $t_i \in \mathcal{T}_a$ can be written as:

$$p(t_i, \mathbf{m}) = p_i(\mathbf{m}) = \frac{m_i}{\sum_{j \in \{j | t_j \in \mathcal{T}_a\}} m_j} \qquad (4)$$

We shall now derive a closed form solution for the steady state probability of GSPN-1 model by considering the set of reachable markings $\mathbf{m} = (m_1, \ldots, m_{2R+1})$. This is given by Lemma 1. Then we introduce a state aggregation by defining the aggregate state $\mathbf{n} = (n_1, \ldots, n_R)$ where $n_i = m_i + m_{R+i}$, $1 \leq i \leq R$. This state corresponds to the number of customers of class i in the queuing model. Theorem 1 provides the closed form solution for model GSPN-1 in terms of aggregated stationary probability of state $\mathbf{n}$. Finally the GSPN-1 model is shown to be equivalent to the M/M/k FCFS multiclass queuing system in terms of stationary probability.

Lemma 1. *Let $\mathbf{m} = (m_1, \ldots, m_{2R+1})$ be a reachable tangible state of the GSPN-1. Then if the stability condition holds, the stationary state probability can be written as follows:*

$$\pi(\mathbf{m}) = \pi_0 \prod_{i=1}^{R} \lambda_i^{m_i + m_{R+i}} \frac{(\sum_{i=R+1}^{2R} m_i)!}{\prod_{i=R+1}^{2R} m_i!} \cdot \frac{(\sum_{i=1}^{R} m_i)!}{\prod_{i=1}^{R} m_i!} \prod_{j=1}^{\sum_{i=1}^{2R} m_i} \frac{1}{\mu(j)}. \qquad (5)$$

where π_0 is a normalizing constant, $\mu(j) = x(j)\mu$ following the BCMP conventions.

The proof is given in appendix in the technical report (Balsamo & Marin, 2007) and is based on verifying the set of the CTMC global balance equations.

Theorem 1. *Consider model GSPN-1 and let $n_i = m_i + m_{R+i}$, $1 \leq i \leq R$ and $\mathbf{n} = (n_1 \ldots, n_R)$ be an aggregated state. Let $\pi_a(\mathbf{n})$ be the steady state probability of n_i for $i = 1, \ldots, R$. Then we can write:*

$$\pi_a(\mathbf{n}) = \pi_0 \frac{(\sum_{i=1}^{R} n_i)!}{\prod_{i=1}^{R} n_i!} \prod_{i=1}^{R} \lambda_i^{n_i} \prod_{i=1}^{\sum_{i=1}^{R} n_i} \frac{1}{\mu(i)} \quad \forall \mathbf{n} \in \mathbb{N}^R. \qquad (6)$$

The proof is based on a convolution formula for binomial coefficients and can be found in (Balsamo & Marin, 2007).

Corollary 1. *The M/M/k queuing system with FCFS discipline, R customer classes, arrival rates λ_i, $1 \leq i \leq R$, single server rate μ and steady state probability $\pi'(\mathbf{n})$ is equivalent to the GSPN-1 in terms of steady state probability, i.e., $\pi_a(\mathbf{n}) = \pi'(\mathbf{n})$ for all $\mathbf{n} \in \mathbb{N}^R$ where $\pi_a(\mathbf{n})$ is the aggregated probability of GSPN given by formula (6).*

Proof. It follows immediately from equation (3) and Theorem 1. □

Note it can be shown by trivial counterexamples that GSPN-1 does not hold the steady state distribution (2) when the service rate is class dependent.

GSPN-1 can as well simulate a single server FCFS service station with an BCMP-like load dependent service rate as proved in the technical report (Balsamo & Marin, 2007). The net structure complexity is linear on R, the number of customer classes.

5 REPRESENTING M/M/k/LCFSPR QUEUE BY GSPN

In this section we introduce a GSPN which can be considered equivalent, for steady state probability, to a multiclass M/M/k queue with LCFS with preemptive resume scheduling discipline. As we consider just exponentially distributed service times, we do not consider the problem of representing the resume. We provide a model for this queue system whose structure is finite and depends only on the number of classes of customers, i.e., not on the number of servers.

Definition 2 (GSPN-2). *According to GSPN definition given in Section 2:*

– $\mathcal{P} = \mathcal{P}_q \cup \mathcal{P}_w \cup \mathcal{P}_a \cup \{P_{3R+1}\}$ *where* $\mathcal{P}_q = \{P_1, \ldots, P_R\}$ *and* $\mathcal{P}_w = \{P_{R+1}, \ldots, P_{2R}\}$ *and* $\mathcal{P}_a = \{P_{2R+1}, \ldots, P_{3R}\}$,
– $\mathcal{T} = \mathcal{T}_q \cup \mathcal{T}_w \cup \mathcal{T}_f \cup \mathcal{T}_g$ *where* $\mathcal{T}_q = \{t_1, \ldots, t_R\}$ *and* $\mathcal{T}_w = \{T_{R+1}, \ldots, T_{2R}\}$ *and* $\mathcal{T}_f = \{t_{2R+1}, \ldots, t_{3R}\}$ *and* $\mathcal{T}_g = \{t_{ij},\ 1 \leq i, j \leq R\}$,
– *function Π is defined as follows:*

$$\Pi(\tilde{t}) = \begin{cases} 1 & \text{if } \tilde{t} \in \mathcal{T}_q \cup \mathcal{T}_f \cup \mathcal{T}_g \\ 0 & \text{if } \tilde{t} \in \mathcal{T}_w \end{cases},$$

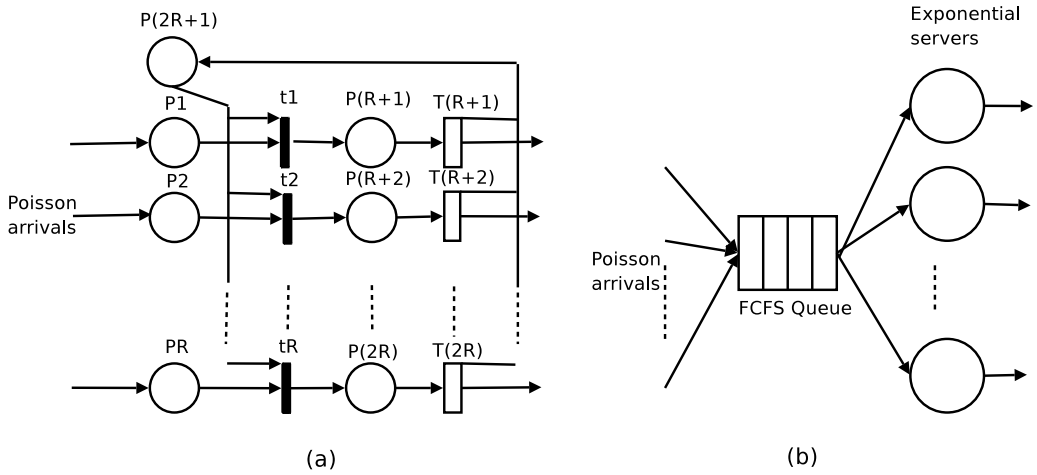


Fig. 1. (a) Graphical representation of model GSPN-1. (b) Queuing station associated

- *Let* $1 \leq i, j \leq R$. *The input and output vectors of* $t_i \in T_q$: $\mathbf{I}(t_i) = \mathbf{e}_i + \mathbf{e}_{3R+1}$ *and* $\mathbf{O}(t_i) = \mathbf{e}_{R+i}$. *The input and output vectors for* $T_{R+i} \in \mathcal{T}_w$: $\mathbf{I}(t_{R+i}) = \mathbf{e}_{R+i}$ *and* $\mathbf{O}(t_{R+i}) = \mathbf{e}_{3R+1}$. *The input and output vectors for* $t_{2R+i} \in \mathcal{T}_f$: $\mathbf{I}(t_{2R+i}) = \mathbf{e}_{2R+i} + \mathbf{e}_{3R+1}$ *and* $\mathbf{O}(t_{2R+i}) = \mathbf{e}_{R+i}$. *The input and output vectors for* $t_{ij} \in \mathcal{T}_g$: $\mathbf{I}(t_{ij}) = \mathbf{e}_{2R+i} + \mathbf{e}_{R+j}$ *and* $\mathbf{e}_j + \mathbf{e}_{R+i}$,
- $\mathbf{H}(t_i) = (0, \ldots, 0)$ *for* $t_i \in \mathcal{T}_q \cup \mathcal{T}_w \cup \mathcal{T}_f$ *and* $\mathbf{H}(t_{ij}) = \mathbf{e}_{3R+1}$ *for* $t_{ij} \in \mathcal{T}_g$,
- *for* $1 \leq i, j \leq R$ *let* $w(T_{R+i}, \mathbf{m}) = m_{R+i}\mu_i$, $w(t_i, \mathbf{m}) = m_i$, $w(t_{2R+i}, \mathbf{m}) = 1$ *and* $w(t_{ij}, \mathbf{m}) = m_{R+j}$,
- $\mathbf{m_0} = (0, \ldots, 0, k)$.

Tokens arrive to places P_{2R+i}, $1 \leq i \leq R$, *according to Poisson stochastic processes.*

Figure 2 shows a graphical model for $R = 2$ classes LCFSPR queue where dotted lines are introduced for the sake of readability and they do not have ant particular meaning. Note that when a token arrives to the place P_{2R+i} it is temporally (i.e. the state is vanishing) stored in P_{2R+i} and we have two cases:

- there is at least one free server, i.e. $m_{3R+1} > 0$, thus the customer goes immediately in service. This is modelled by the immediate transition set $\mathcal{T}_f$
- all the servers are busy, i.e. $m_{3R+1} = 0$, so a customer is preempted and put in queue and the new customer goes in service. This is modelled by R^2 transitions, $\mathcal{T}_g$. The inhibitor arcs are needed to avoid pre-emption when there is at least one free server.

By the structure analysis of the network we can solve the conflicts on immediate transitions introducing just one simple function. When one or more transitions of $\mathcal{T}_q$ are enabled, the probability of firing for the i-th transition is:

$$p(t_i, \mathbf{m}) = p_i(\mathbf{m}) = \frac{m_i}{\sum_{l=1}^{R} m_l}. \tag{7}$$

When one or more transitions of $\mathcal{T}_g$ are enabled, the probability of firing is:

$$p(t_{ij}, \mathbf{m}) = p_{ij}(\mathbf{m}) = \frac{m_{R+j}}{\sum_{l=1}^{R} m_{R+l}}. \tag{8}$$

Now we can state a main lemma for model GSPN-2 representation:

Lemma 2. *Let* $\mathbf{m} = (m_1, \ldots, m_{3R+1})$ *be a reachable tangible marking of GSPN-2 model. Then if the stability condition holds, the stationary state probability can be written as follows:*

$$\pi(\mathbf{m}) = \pi_0 \prod_{i=1}^{R} \lambda_i^{m_i + m_{R+i}} \frac{(\sum_{i=1}^{R} m_i)!}{\prod_{i=1}^{R} m_i!} \frac{(\sum_{i=1}^{R} m_{R+i})!}{\prod_{i=1}^{R} m_{R+i}!} \cdot \prod_{i=1}^{R} \Big(\frac{1}{\mu_i}\Big)^{m_i + m_{R+i}} \prod_{j=1}^{\sum_{i=1}^{2R} m_i} \frac{1}{\min(j, k)}. \tag{9}$$

where μ_i *is the average service rate for one customer of class* i *when there are no other customers in the system,* k *is the number of servers,* π_0 *is a normalizing constant.*

The proof is given in the technical report (Balsamo & Marin, 2007).

Theorem 2. *Consider model GSPN-2 and let* $n_i = m_i + m_{R+i}$, $1 \leq i \leq R$ *and* $\mathbf{n} = (n_1, \ldots, n_R)$ *be an aggregated state. Let* $\pi_a(\mathbf{n})$ *be the steady state probability of* n_i *for* $i = 1, \ldots, R$. *Then we can write:*

$$\pi_a(\mathbf{n}) = \pi_0 \frac{(\sum_{i=1}^{R} n_i)!}{\prod_{i=1}^{R} n_i!} \prod_{i=1}^{R} \lambda_i^{n_i} \prod_{i=1}^{R} \Big(\frac{1}{\mu_i}\Big)^{n_i} \cdot \prod_{i=1}^{\sum_{i=1}^{R} n_i} \frac{1}{\min(k, i)} \quad \forall \mathbf{n} \in \mathbf{N}^R. \tag{10}$$

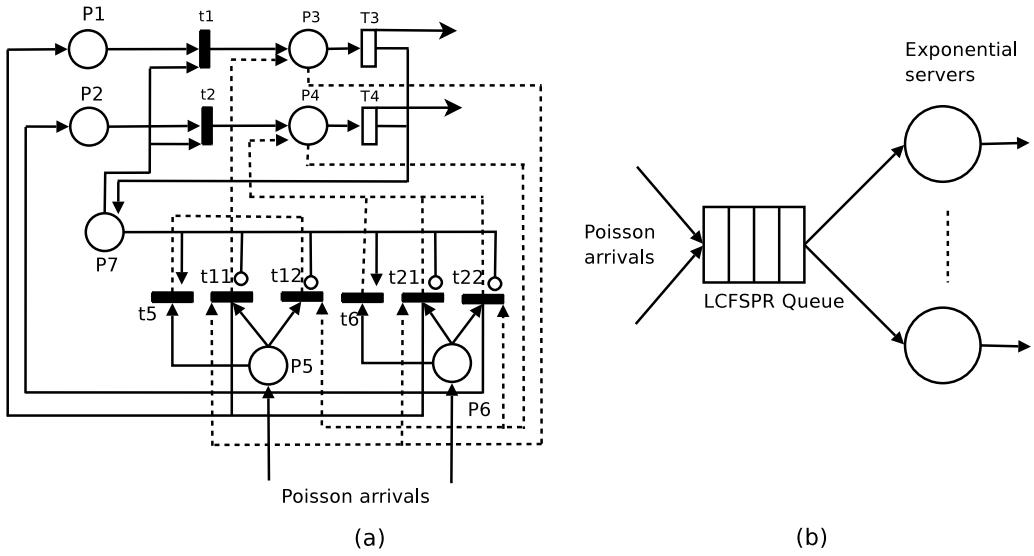


Fig. 2. (a) Graphical representation of model GSPN-2 for two classes of customers. (b) Queuing station associated.

The proof is based on the Vandermonde formula and it is similar the one given for Theorem 1 in (Balsamo & Marin, 2007).

Corollary 2. *The M/M/k queuing system with LCFSPR discipline, R customer classes, arrival rates λ_i, single server rate μ_i for class i customers and steady state probability $\pi'(\mathbf{n})$ is equivalent to model GSPN-2 in terms of steady state probability, i.e., $\pi_a(\mathbf{n}) = \pi'(\mathbf{n})$ for all $\mathbf{n} \in \mathbf{N}^R$, where $\pi_a(\mathbf{n})$ is the aggregated probability of GSPN given by formula (10). The normalizing constant is $\pi_0 = \pi(0,\ldots,0,k) = \pi'(0,\ldots,0,k)$.*

Proof. It follows immediately from queuing theory and Theorem 2. □

The net GSPN-2 can as well simulate a single server LCFSPR service station with a BCMP-like load dependent service rate as proved in technical report (Balsamo & Marin, 2007).

By defining $n_i = m_i + m_{R+i}$ we can aggregate the states and we can prove that the steady state probability π_a of the aggregated CTMC is identical to probability π' defined by equation (2). For what concerns the net structure complexity, the number of places grows as $\mathcal{O}(R)$ and the number of transitions grows as $\mathcal{O}(R^2)$.

6 REPRESENTING M/M/k/PS QUEUE AND M/M/∞ QUEUE BY GSPN

The processor sharing discipline can be easily represented considering that the k processors are shared among the users in the system. Different classes of users can have different average time services, but all modelled by exponentially distributed random variables. We can think that the k servers are shared among the R classes in proportion to the number of customers of the classes.

Definition 3 (GSPN-3). *Let us define the model GSPN-3 as follows:*

- $\mathcal{P} = \{P_1,\ldots,P_R\}$,
- $\mathcal{T} = \{T_1,\ldots,T_R\}$,
- $\Pi(T_i) = 1$ *for each* $T_i \in \mathcal{T}$,
- $\mathbf{I}(T_i) = \mathbf{e}_i$ *and* $\mathbf{O}(T_i) = (0,\ldots,0)$ *for each* $T_i \in \mathcal{T}$,
- $\mathbf{H}(T_i) = (0,\ldots,0)$ *for each* $T_i \in \mathcal{T}$,
- $w(T_i,\mathbf{m}) = \frac{m_i}{m}\min(k,m)$ *where* $m = \sum_{j=1}^{R} m_j$ *for each* $T_i \in \mathcal{T}$,
- $\mathbf{m_0} = (0,\ldots,0)$.

Note that this model is equivalent to a queuing system with PS discipline and one server with load-dependent exponential service time to simulate the multi-server feature. Therefore it immediately follows the theorem:

Theorem 3. *Consider model GSPN-3. Then if stability condition holds the stationary state probability can be written as follows:*

$$\pi(\mathbf{m}) = \pi_0 \frac{(\sum_{i=1}^{R} m_i)!}{\prod_{i=1}^{R} m_i!} \prod_{i=1}^{R} \lambda_i^{m_i} \prod_{i=1}^{R} \left(\frac{1}{\mu_i}\right)^{m_i} \cdot \prod_{i=1}^{\sum_{i=1}^{R} m_i} \frac{1}{\min(k,i)}, \quad (11)$$

where μ_i is the average service rate for one customer of class i when there are no other customers in the system, k is the number of servers, π_0 is a normalizing constant.

This model is similar to the compact model introduced in (Balbo et al., 2003), the only difference is that we allow a whole state dependent firing rate thus we don't need a place whose tokens represent the total number of customers in the system.

Model GSPN-3 can easily represent also the IS center. It suffices to set the firing rates of each transition T_i as $m_i\mu_i$, $1 \leq i \leq R$.

7 M ⇒ M PROPERTY ON THE GSPN REPRESENTATION

Markov implies Markov property is introduced and studied by Muntz (Muntz, 1972). In that paper he shows that if a queuing system with Poisson arrivals presents departures according to a Poisson process (M⇒M property) then a combination of service centers of this type in a queuing network has a product-form solution. As we are considering GSPNs we will prove that a combination of GSPN-1, GSPN-2 and GSPN-3 models still holds a closed-form steady state probability by defining appropriate traffic processes over the CTMC associated to each of the models and using the results given in (Melamed, 1979) which generalize Muntz's work. We now briefly review Melamed's results limited to a CTMC in steady state. Consider an ergodic CTMC with state space Γ and a set of traffic transitions denoted by Θ_1, ..., Θ_R, where $\Theta_i \subseteq \Gamma \times \Gamma$, $\Theta_i \neq \emptyset$. Let us define $K_i(t)$ as the process which counts the number of transitions $(\alpha, \beta) \in \Theta_i$ up to t. Let $m_i = \sum_{\gamma \in \Gamma} \sum_{\eta \in \Theta_i(\cdot,\gamma)} \pi(\eta)\xi(\eta \rightarrow \gamma)$ and for each state $\gamma \in \Gamma$ let $m_i(\gamma) = \sum_{\eta \in \Theta_i(\cdot,\gamma)} \pi(\eta)\xi(\eta \rightarrow \gamma)$ where $\Theta_i(\cdot,\gamma) = \{\beta | (\beta,\gamma) \in \Theta_i\}$ and $\xi(\eta \rightarrow \gamma)$ is the transition rate between states η and γ.
Then we can state that $K_i(t)$ are mutually independent Poisson processes if and only if the following equation holds:

$$\forall \gamma \in \Gamma, \quad \sum_{i=1}^{R} m_i(\gamma) = \pi(\gamma) \sum_{i=1}^{R} m_i \qquad (12)$$

We aim to study the departure traffic processes from our models. Take for example model GSPN-1, we can define R traffic processes as follows:

$$\Theta_i = \{(\mathbf{m}', \mathbf{m}) : |\mathbf{m}'|_i = |\mathbf{m}|_i + 1\}, \ i = 1, \ldots, R, \qquad (13)$$

where $|\mathbf{m}|_i = m_i + m_{R+i}$. In our case, in order to prove that $K_i(t)$ are independent Poisson processes when there are Poisson arrivals, it suffices to prove that:

$$\forall \gamma \in \Gamma, \quad \sum_{\eta \in \Theta_i(\cdot,\gamma)} \pi(\eta)\xi(\eta \rightarrow \gamma) = \lambda_i \pi(\gamma), \qquad (14)$$

In (Balsamo & Marin, 2007) we prove that this condition holds for GSPN-1, GSPN-2 and GSPN-3 models by defining appropriate traffic processes. As observed in (Melamed, 1979) this property of the CTMC is equivalent to the $M \Rightarrow M$ given by Muntz thus it assures that a BCMP-like composition of these GSPN models holds a closed-form steady state probability function. Random switches between the blocks and user class switches can be easily modelled by immediate transitions.

8 FINAL REMARKS

In this paper we have shown how to represent multiclass single queuing systems by structurally finite GSPN for various queuing disciplines. For each of the BCMP center types we have introduced a GSPN model whose steady state probability, aggregating on the number of customers in the system for each class, is equal to the correspondent single queue service center. Hence the two models are equivalent in terms of steady state distribution and average performance indexes. The main advantages of our representation are the following.

- We define a finite GSPN model. The abstraction level of the GSPN model allows the representation of the queuing behavior without introducing an high level of details in the state specification. We distinguish the customers waiting in the queue from those being served without taking in account the arrival order. This leads to a steady state probability which is less detailed than the one proposed in (Balbo et al., 2003) which considers the single station detailed representation with the order of the customers in the queue, similarly to the BCMP paper (Baskett et al., 1975). On the other side the models we introduce are more detailed than those which just consider the total number of customers in a center as the compact models of (Balbo et al., 2003).
- The FCFS and the LCFSPR (or PS) scheduling disciplines have different GSPN representations. The GSPN models simulate the corresponding queuing system even if their semantic is different.

The main idea of the definition of the GSPN models is a probabilistic choice of the customer to serve when a server is avaiable and of the customer to preempt when there is an arrival to a LCFSPR station.

The M⇒M property allows us to state that a combination of GSPN-1, GSPN-2 and GSPN-3 models similar to the service centers combination in BCMP networks, has a simple closed form steady state probability. In (Afshari, Bruell, & Kain, 1982) authors define a queuing center isomorphic to GSPN-1 and show how it can be embedded in a BCMP queuing network so that the steady state probability function of the network does not change. In the GSPN formalism probabilistic routing can be easily simulated by

introducing a block with a place and an immediate transition for each possible route just after the timed transitions of the models.

Further research deals with the extension of the proposed LCFSPR model to Coxian service time distributions and the definition of algorithms to identify GSPN which are compositions of models GSPN-1, GSPN-2, GSPN-3 and in order to obtain efficiently a set of significant performace indexes.

REFERENCES

Afshari, P. V., Bruell, S. C., & Kain, R. Y. (1982). Modeling a new technique for accessing shared buses. In *Proc. of the computer network performance symposium* (pp. 4–13). New York, NY, USA: ACM Press.

Balbo, G., Bruell, S. C., & Ghanta, S. (1998). Combining queueing network and generalized stochastic Petri nets for the solution of complex models of system behavior. *IEEE Trans. on Computers*, *37*, 1251–1268.

Balbo, G., Bruell, S. C., & Sereno, M. (2002). Product form solution for Generalized Stochastic Petri Nets. *IEEE Trans. on Software Eng.*, *28*, 915–932.

Balbo, G., Bruell, S. C., & Sereno, M. (2003). On the relations between BCMP Queueing Networks and Product Form Solution Stochastic Petri Nets. *Proc. of 10th International Workshop on Petri Nets and Performance Models, 2003.*, 103–112.

Balsamo, S., & Marin, A. (2007). *On representing multiclass M/M/k queues by generalized stochastic Petri nets* (Tech. Rep. No. CS-2007-1). via Torino, 155 Venice.

Baskett, F., Chandy, K. M., Muntz, R. R., & Palacios, F. G. (1975). Open, closed, and mixed networks of queues with different classes of customers. *J. ACM*, *22*(2), 248–260.

Chiola, G., Marsan, M. A., Balbo, G., & Conte, G. (1993). Generalized stochastic Petri nets: a definition at the net level and its implications. *IEEE Trans. on Software Eng.*, *19*(2), 89–107.

Coleman, J. L., Henderson, W., & Taylor, P. G. (1996). Product form equilibrium distributions and a convolution algorithm for Stochastic Petri nets. *Performance Evaluation*, *26*, 159–180.

Henderson, W., Lucic, D., & Taylor, P. G. (1989). A net level performance analysis of Stochastic Petri Nets. *J. Austral. Math. Soc. Ser. B*, *31*, 176–187.

Marsan, M. A., Balbo, G., Conte, G., Donatelli, S., & Franceschinis, G. (1995). *Modelling with generalized stochastic Petri nets.* Wiley.

Melamed, B. (1979). On Poisson traffic processes in discrete-state markovian systems with applications to queueing theory. *Advances in Applied Prob.*(11), 218–239.

Muntz, R. R. (1972). *Poisson departure processes and queueing networks* (Tech. Rep. No. IBM Research Report RC4145). Yorktown Heights, New York.

Vernon, M., Zahorjan, J., & Lazowska, E. D. (1987). A comparsion of performance Petri Nets and queueing network models. *Proc. 3rd Intern. Workshop on Modelling Techniques and Performance Evaluation*, 181–192.

AUTHOR BIOGRAPHIES

SIMONETTA BALSAMO is a full professor of Computer Science at the university Ca' Foscari of Venice, Italy. Her research interests include performance and reliability modeling and analysis of computer and communication systems, parallel and distributed processing, distributed simulation, quantitative analysis of software architectures and integration of specification languages and performance models. She has published several papers in international journal, conference proceeding, books and special editions. She has served as general chair, program chair and program committee member for several internatinal conference, is associated editor of Performance Evaluation Journal. She is a member of ACM.

ANDREA MARIN was born in Venice, Italy and went to the Ca' Foscari University of Venice, where he studied computer science and obtained his degree in 2005. He is now a PhD student at Ca' Foscari university and he is interested in stochastic models for performance evaluation, and models for biological systems. His email address is: marin@dsi.unive.it and his Web-page can be found at http://www.dsi.unive.it/ marin/.

A Multilevel Algorithm based on Binary Decision Diagrams

Johann Schuster, Markus Siegle
Universität der Bundeswehr München
Institut für Technische Informatik
{Johann.Schuster, Markus.Siegle}@unibw.de

Abstract: In this paper, a new variant of the multilevel algorithm for computing the steady-state solution of a continuous-time Markov chain is proposed. The method is integrated into a symbolic framework, where the CTMC is represented in a compact way using multi-terminal binary decision diagrams (MTBDD). It is shown how to represent the original CTMC and several aggregated chains within the same decision diagram, where particular attention is devoted to the question of how to deal with unreachable states. Some preliminary empirical results are provided which indicate that the method has the potential to solve very large Markov chains in an efficient manner.
Keywords: Markov chain, numerical analysis, aggregation, multilevel algorithm, Binary Decision Diagram.

I. Introduction

Markov chains are very popular in the area of model-based performance and dependability evaluation of computer and communication systems. Today, several well-understood high-level modelling formalisms are available for specifying such models, and all phases of Markov chain generation and analysis are supported by powerful and user-friendly tools. State space explosion is an adverse phenomenon that occurs during the modelling of complex systems, especially of those consisting of several concurrent subsystems. In view of this situation, researchers have developed novel techniques for representing Markov chains with the help of decision diagram data structures, which make it possible to construct and manipulate Markov chains of immense size. However, even though the generation of billions of states is only a matter of seconds (using techniques such as the ones described in [5], [16]), calculating state probabilities by performing numerical analysis is still a serious bottleneck. In order to alleviate this bottleneck, this paper proposes a new version of the multilevel algorithm for calculating steady-state probabilities, which is based entirely on decision diagrams. Since data structures such as multi-terminal binary decision diagrams (MTBDD) [1], [9] offer a natural structuring of the transition matrix, they seem to be quite ideally suited for block aggregation methods.

A. Related work

Many different numerical methods exist for calculating the steady-state probability vector of a continuous-time Markov chain (CTMC). Apart from standard iterative methods such as Jacobi, Gauss-Seidel or SOR, methods based on the aggregation and disaggregation of the states of the Markov chain (so-called AD methods) were proposed already in the 1970ies. Among these "classical" AD methods are the method of Takahashi [24], the method of Courtois [8] (aimed at nearly completely decomposable Markov chains) and the method of Koury/McAllister/Stewart [13]. An overview of these techniques may be found in [23]. Recently, Bazan et al. [2] proposed a self-correcting aggregation technique, where several approximate first level aggregations are corrected by results from common second level aggregations.

In the 1990ies, Horton and Leutenegger developed a multilevel AD method, inspired by multigrid solvers which had turned out to be very effective for the solution of partial differential equations [12]. Buchholz proposed a multilevel solution method for very large structured Markov models whose generator matrix is represented compactly as a Kronecker expression [3]. This work was carried further in [4] where hierarchical Kronecker structures and different multigrid types were studied.

For Markov chains represented symbolically by binary decision diagrams, Parker was the first to develop numerical analysis techniques whose speed was competitive with those based on explicit data structures [20]. He introduced a hybrid approach, in which the matrix is represented symbolically while the vectors are stored as ordinary arrays, and showed how to best exploit the available memory by replacing parts of the decision diagram by explicit data structures (i.e. sparse matrices). Based on Parker's work, Mehmood developed a symbolic out-of-core method where only a small part of the vector resides in main memory while the remaining part is kept on disk [18]. Lampka et al. were able to speed up Parker's approach by employing a novel type of decision diagram [17].

To the best of our knowledge, multilevel analysis in connection with symbolic decision diagram data structures has not been investigated before.

B. Organisation

The rest of this paper is structured as follows: Sec. II provides the necessary background material on the multilevel method and on the symbolic representation of Markov chains. Sec. III, the core section of the paper, presents new data structures and algorithms which

make it possible to integrate the multilevel aggregation approach smoothly into a symbolic framework. In Sec. IV, some empirical results are presented and discussed, and Sec. V concludes the paper.

II. Preliminaries

A. The multilevel method

Let Q be the generator matrix of an irreducible finite-state CTMC with n states. The unknown vector $\vec{\pi}$ of its steady-state probabilities is the unique solution of the linear system (further called *system*),

$$\vec{\pi}Q = \vec{0} \tag{1}$$

which satisfies the normalising condition $\sum_i \pi_i = 1$. The multilevel method is a method for solving the above equation by recursively aggregating the original system, thereby obtaining systems of smaller dimension. For the moment, only a single aggregation step is considered: Let the current (fine) state space with n states $i, j, \ldots$ be partitioned into N macro states $I, J, \ldots$, where $N < n$. Assuming that $\vec{\pi}^{(l)}$ is the current approximation to the steady-state vector at the fine level l, the transition rates of the aggregated (coarse) Markov chain at level $l-1$ are computed according to the following aggregation equation:

$$q_{IJ}^{(l-1)} = \frac{\sum_{i\in I} \pi_i^{(l)} \sum_{j\in J} q_{ij}^{(l)}}{\sum_{i\in I} \pi_i^{(l)}} \tag{2}$$

and the initial approximation to the steady-state vector of the coarse system is obtained by setting $\pi_I^{(l-1,in)} = \sum_{i\in I} \pi_i^{(l)}$. From $\vec{\pi}^{(l-1,in)}$ an improved approximation $\vec{\pi}^{(l-1,out)}$ is then computed by carrying out some smoothing iteration steps (of the Jacobi, Gauss-Seidel or SOR scheme, for example) at the coarse level and/or by possibly aggregating the Markov chain even further. In the following disaggregation step, the improved approximation on the coarse level is used to correct the previous approximation at the fine level. All micro states i belonging to the same macro state I are corrected by the same factor according to the disaggregation equation

$$\pi_i^{(l,new)} = \frac{\pi_I^{(l-1,out)}}{\pi_I^{(l-1,in)}} \pi_i^{(l,old)} \tag{3}$$

Fig. 1 shows the pseudo code of the basic multilevel algorithm. In line (2), if the lowest level has been reached the system is solved (approximately or exactly) without further recursion. Otherwise, in line (4), some smoothing steps are carried out before the aggregated system is computed in line (5). Line (6) contains the recursive call to the algorithm at the next lower (i.e. coarser) level. Its results are used in lines (7-8) to correct the solution at level l, which is followed by some further smoothing steps in line (9).

B. MTBDD-based Markov chain representation

This subsection introduces the MTBDD-based representation of CTMCs with the help of a simple example. For more detailed information on the use of MTBDDs for compactly encoding CMTCs see [20], [22], [19]. Consider the Markov chain $\mathcal{M}$ whose transition rate matrix $R_{\mathcal{M}}$ is shown in Fig. 2. This CTMC has four states (numbered 0,1,2,3) of which only three are actually reachable, i.e. state 2 is unreachable. Table I shows how the individual transitions of $\mathcal{M}$ can be encoded by bitstrings, where $\vec{\mathsf{s}}$ and $\vec{\mathsf{t}}$ are bit vectors (vectors of Boolean variables) of appropriate length $n_V = 2$. The s-variables encode the source state, and the t-variables encode the target state of a transition. Column headed "path" contains the shuffled concatenation of the s- and t-values, which corresponds to the variable ordering used in the decision diagram. Such an interleaved ordering is a commonly accepted heuristic which yields not necessarily optimal but provable compact decision diagrams [10]. The MTBDD M encoding CTMC $\mathcal{M}$ is shown in Fig. 3 (a). Its non-terminal nodes are labelled with Boolean variables from the ordered set $\{\mathsf{s}_1, \mathsf{t}_1, \mathsf{s}_2, \mathsf{t}_2\}$, and its terminal nodes carry the transition rates. Each path through the MTBDD, starting at the root and ending in a non-zero terminal node encodes a valid transition of $\mathcal{M}$, where dashed edges correspond to the Boolean value 0 and solid edges correspond to the Boolean value 1. Starting from a fixed node, the successor reached by the dashed line will be called the else-successor, the node reached by the solid line will be called the then-successor.

```
(1)  ml(l, Q^(l), π^(l,in))
(2)    if l = 0 then solve π^(l,out) Q^(l) = 0
(3)    else
(4)      π^(l,old) := pre_smoothing(π^(l,in))
(5)      compute π^(l-1,in) and Q^(l-1) acc. to eq. (2)
(6)      ml(l-1, Q^(l-1), π^(l-1,in))
(7)      for all I
(8)        for all i ∈ I compute π_i^(l,new) acc. to eq. (3)
(9)      π^(l,out) := post_smoothing(π^(l,new))
(10)   return
```

Fig. 1. The multilevel algorithm

$$R_{\mathcal{M}} = \begin{pmatrix} 0 & \lambda & 0 & \lambda \\ 0 & 0 & 0 & \nu \\ 0 & 0 & 0 & 0 \\ \mu & \nu & 0 & 0 \end{pmatrix}$$

Fig. 2. An example Markov chain

$\vec{\mathsf{s}}$	$\vec{\mathsf{t}}$	path	rate
00	01	0001	λ
00	11	0101	λ
01	11	0111	ν
11	00	1010	μ
11	01	1011	ν

TABLE I: Binary encoding of CTMC $\mathcal{M}$

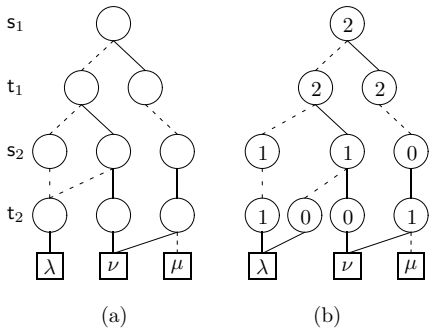


Fig. 3. MTBDD representation of CTMC $\mathcal{M}$ from Fig. 2

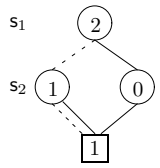


Fig. 4. Offset-labelled BDD *reach* representing the reachability set of $\mathcal{M}$ from Fig. 2

B.1 Offset-labelling

When generating a Markov chain from a high-level modelling formalism, thereby encoding states as bitstrings of length n_V, it is very common that certain such patterns correspond to unreachable states. In fact, symbolic algorithms are often employed in order to determine the set of reachable states, which may be only a very small subset of the so-called potential state space. Tools such as PRISM [15], [21], CASPA [14] and the Moebius pZDD-engine [17] contain such symbolic reachability schemes. During numerical analysis, in order not to waste memory space, the iteration vectors should be stored as arrays whose size corresponds to the number of reachable states n_r (not the number of potential states!). Therefore it is necessary to construct a mapping which maps bitstrings from the potential state space $\{0, 1, \ldots, 2^{n_V} - 1\}$ to the dense set $\{0, 1, \ldots, n_r - 1\}$. An order-preserving such mapping can be achieved by enriching the MTBDD with offsets which may be computed by an algorithm described in [20]. More precisely, first a BDD *reach* encoding the set of reachable states is computed and labelled by offsets. Using those offsets, the MTBDD $\mathcal{M}$ encoding the transitions of the CTMC is then enhanced by offsets in a double fashion, i.e. both for the s- and for the t-variables. Returning to the example, the offset-labelled BDD *reach* representing the reachability set of $\mathcal{M}$ is shown in Fig. 4. From this BDD, it is possible, for example, to calculate the index of state 11 (binary representation) as $2+0 = 2$. The offset-labelled version of MTBDD M is shown in Fig. 3 (b). (The t_2-labelled node at the bottom left has to be duplicated in this example since it can be reached via two different paths with different t_2-offsets.) During traversal of a path of the offset-labelled MTBDD, the index of the source state of the encoded transition may be determined by adding all offsets of s-nodes which are left via a 1-valued (solid) edge. Likewise, the index of the target state may be determined by adding all offsets of t-nodes which are left via a 1-valued edge.

III. Aggregation of the DD

In the following, the data structures and algorithms for the symbolic multilevel algorithm are presented. It will turn out that all the aggregated matrices can be added to the MTBDD without having to change the basic structure of the diagram, which will result in a small time and memory overhead for the aggregation process. Below, the term *fine system* will be used for the given, not aggregated system, while *coarse system* will mean an aggregated system. Furthermore, let the set of reachable states of the Markov chain be stored in a BDD *reach*. Usually in a BDD nodes with identical then- and else-successors are eliminated, but for compatibility with the given algorithms it is assumed that in the paths leading to the terminal 1-node of *reach* all don't care nodes are inserted explicitly. This is also done in [20] in order to store offset information in these nodes.

For an efficient implementation of the aggregation procedure in an MTBDD environment it is necessary to use macro states which combine 2^i neighbouring states of the potential state space. In terms of BDD *reach* this means that for one aggregation step i variables are aggregated, starting at the leaves of the graph.

To be consistent with the ordinary multilevel algorithm we use the following convention: With the notations of section II, fix a number of levels l to get a hierarchy of the fine system and l aggregated systems. Each level $i \in \{l-1, \ldots, 0\}$ in the multilevel algorithm is assigned to a variable level, the level l corresponds to the terminal nodes of BDD *reach*. In order to avoid confusion, in the following, aggregation levels referencing the algorithm in Fig. 1 will be called *aggregation level* or *agg_level*, while the term *aggregation variable level* or *agg_var_level* will be used for aggregation variable levels in BDD *reach*.

The example given in Fig. 5 (a) indicates that the mapping from aggregation levels to aggregation variable levels is well defined: Aggregation level 2 corresponds to the fine system, i.e. the constant level of BDD *reach*. The aggregation level 1 reduces the fine system by removing variable s_3, aggregation level 0 takes the result of aggregation 1 and removes variable s_2.

As the assignment above is in fact a bijection, both aggregation levels and aggregation variable levels could be used to describe the symbolic multilevel algorithm. For simplicity, the algorithms will always be described using the aggregation variable levels.

Aggregating a MTBDD according to its variables is not perfectly symmetric. It depends on how many reachable states are within one partition of the potential state space. Aggregating only unreachable states will lead to an unreachable state in the aggregate. Fig. 5 (b) sketches the aggregation idea using the BDD *reach* from Fig. 5 (a). The arrows indicate how the elements in the probability vectors are summed up in the two aggregation steps. Unreachable states are crossed

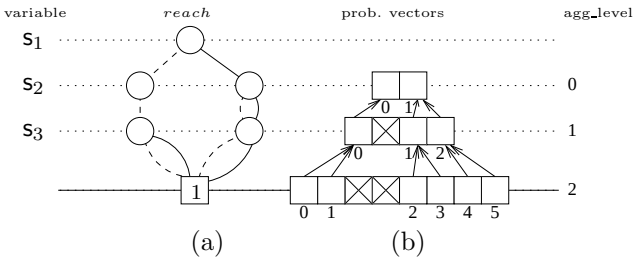


Fig. 5. Aggregation process

```
(1) label(node, s_i)
(2)   if (node=ZERO)
(3)     return 0
(4)   if (s_i=FINAL)
(5)     return 1
(6)   if (node's offsets not calculated)
(7)     node.then_off := label(node.then, s_{i+1})
(8)     node.else_off[system] := label(node.else, s_{i+1})
(9)   return node.then_off+node.else_off[system]
```

Fig. 6. Offset labelling algorithm

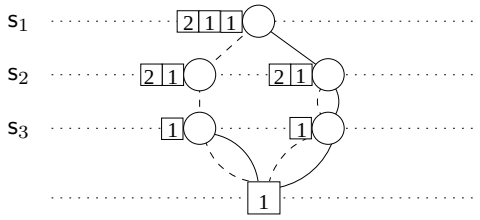


Fig. 7. Multi-offset labelling

out. The numbers under the vectors are the indices of the elements in the vector of reachable states. The aggregation scheme is given by the pattern of reachable and unreachable states.

The following subsections provide the necessary extensions to the MTBDD structure and the corresponding algorithms for performing multilevel cycles.

A. Multi-offset labelling

For the aggregations of the solution vector and the iteration matrix it is necessary to provide BDD *reach* with offset information of the fine and all aggregated systems.

The offset labelling algorithm in Fig. 6 is basically the same as the one given in [20]. The constant ZERO denotes the terminal zero node, the constant FINAL is the variable level where the recursion shall bottom out. The algorithm is identical to the one in [20] when setting FINAL to the level for the constants in the MTBDD. Otherwise, when FINAL is set to an aggregation level, nontrivial subgraphs (i.e. subgraphs finally leading to the terminal 1 node) starting with nodes in the FINAL level play the role of the terminal 1 node.

Note that once the algorithm has finished for a certain FINAL level, for further calculations only the else-offsets and the total number of reachable states are significant. So after a reset the then-offsets may be reused for calculating the offsets of the next system. The offset nodes provide arrays for the else-offsets and one entry for a then-offset.

Depending on the number of systems one MTBDD variable level belongs to, different numbers of offsets have to be stored. For example, the root node always has $l+1$ offsets, while the nodes up to the variable level corresponding to the first aggregation level always carry the offsets of the fine system only.

For the example BDD *reach* in Fig. 5 (a) the multi-offset labelling is shown in Fig. 7. The nodes for s_1 carry from left to right offsets for the fine level, aggregation level 1 and 0, while nodes for s_2 only have offsets for the fine level and aggregation level 1, and so on.

B. Aggregating a vector

Once the BDD *reach* is multi-offset labelled, the aggregation of an iteration vector can be performed with the algorithm shown in Fig. 8. As the given algorithm only sums up the probability masses of single states belonging to the same aggregate, the coarse vector has to be initialised with zeroes.

Let S be the variable level of the system to be aggregated and D the variable level of the aggregated system. Set i_S and i_D to the corresponding indices in the offset arrays. Note that $i_D = i_S + 1$ as only neighbouring levels can be aggregated.

The invocation of agg_vector(root(reach),0,0,s_1) performs the aggregation. The algorithm basically calculates the correct offsets for the summations. Until the D level is reached, in lines (7-10) both the fine and the coarse offsets are modified. Between D and S level in lines (11-13) only the fine offset is modified. When reaching the S level, lines (4-6) perform the summation according to the calculated offsets. In lines (2-3) unreachable subgraphs are skipped, as they either remain unreachable after aggregation or do not contribute to the aggregtion with a reachable state.

C. Aggregated iteration matrix

From the offset labelled BDD *reach* it is possible to derive the multi-offset labelled MTBDD of the iteration matrix from the ordinary MTBDD of the iteration matrix. The basic algorithm is the same as in [20]. The main difference is that in the generated MTBDD not the actual offset values but rather pointers to the multi-offset arrays are stored.

During the construction process for every node two pointers to BDD *reach* are stored: One for the s and one for the t variables. As these pointers represent the whole set of offsets corresponding to this node, the usual comparison may be used to detect offset clashes and will be sufficient: A node to be inserted in the offset labelled MTBDD for the iteration matrix is equal to another node on the same level if and only if the node and its offset pointer coincide. Equal nodes won't be re-inserted when constructing the multi-offset labelled MTBDD while unequal nodes have to be inserted.

```
(1)  agg_vector(node, offset, coarse_offset, s_i)
(2)    if (node = ZERO)
(3)      return
(4)    else if (level = S)
(5)      coarse_vector[coarse_offset]+=vector[offset]
(6)      return
(7)    else if (level < D)
(8)      agg_vector(node.else, offset,
                    coarse_offset, s_{i+1})
(9)      agg_vector(node.then, offset+node.else_off[i_S],
                    coarse_offset+node.else_off[i_D], s_{i+1})
(10)     return
(11)   else if (D ≤ level < S)
(12)     agg_vector(node.else, offset,
                    coarse_offset, s_{i+1})
(13)     agg_vector(node.then, offset+node.else_off[i_S],
                    coarse_offset, s_{i+1})
(13)     return
```

Fig. 8. Vector aggregation algorithm

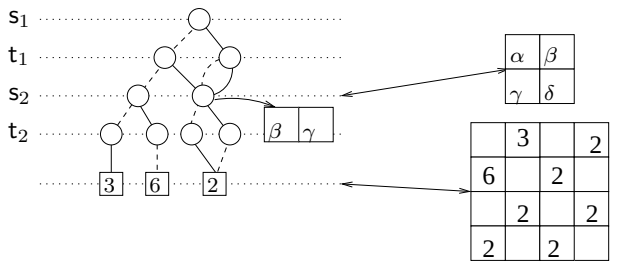


Fig. 9. Aggregation in a MTBDD

An example of the aggregation step is shown in Fig. 9. The multi-offset labelling is not shown in the figure. However, as in this example the reachable state space equals the potential state space, the offset arrays for all the nodes in the s_1 and t_1 variable levels are just [2,1] while the offsets for all nodes in the s_2 and t_2 variables are equal to [1]. The aggregation of the s_2-Variable causes a reduction of the matrix size to one quarter.

To represent the aggregated matrices it remains to store the matrix entries of the aggregated systems in the given MTBDD. It will turn out to be useful to store the diagonal entries of the aggregated systems in separate vectors, so in the MTBDD only non-diagonal elements will have to be stored. Therefore, at the MTBDD nodes in the aggregation variable levels arrays are added storing the aggregated values calculated with eq. (2). The size of such an array is calculated individually for each node by counting the number of paths from the root node leading to this node. In the summation, paths leading to a diagonal value of the aggregated system are skipped. This can be checked by evaluating the offsets corresponding to the aggregated system: The s- and t-offsets have to be different. As long as the BDD is always traversed in a depth first order, it suffices to store for every node in an aggregation variable level the current position of its pointer to the array of aggregated values. Below, this pointer will be called *aggregate_pointer*. Prior to reading or writing values of an aggregated system, the pointers in the corresponding aggregation level have to be reset. Every time a node is visited during BDD traversal its pointer position is incremented. So the pointer keeps the index of the current value in the array as indicated by the unidirectional arrow in Fig. 9.

For the aggregation step, let S be the variable level the aggregation starts from, D the variable level of the aggregated system and i_S and i_D the corresponding indices of the offset arrays. Before calculating the aggregated rates according to eq. (2), the values in the arrays of the D level and the diagonal array have to be reset to 0. The aggregation algorithm is given in Fig. 10. For simplicity, only the case where S is the constant level is shown. In the other case, where S is an aggregation variable level, reading *node.value* in line (5) would change to first incrementing this node's aggregate_pointer and then reading the value given by the current pointer position. To avoid another parameter in the agg_matrix algorithm, the global variable current_ptr is used.

The invocation of agg_matrix(root(MTBDD),0,0,s_1) performs the aggregation. Basically, the algorithm calculates the offsets of the source and destination system. Whenever the level D is reached, the pointer to the current coarse value is set to the next position in the array of aggregated values as seen in lines (12-13). Traversing the nodes from level D to level S, line (8) ensures that only the fine system's offsets are calculated in the next recursive call. At line (5) the S level is reached, so the current weighted fine vector's value is added to the current coarse value. After a node of level D is completely processed, the current coarse value is normalized in line (17) by the coarse vector entry corresponding to the coarse offset of this node, that is the probability mass of all the states in this aggregate, and finally in line (18) this new non-diagonal matrix entry is subtracted from the diagonal value. For the sake of simplicity, the recursion in line (15) on the s-variables, which is given by four different possibilities, is not shown in detail.

With the aggregated system smoothing steps can be performed in the usual way. Prior to a matrix vector multiplication of an aggregated system, the corresponding counters in each node of the aggregation level have to be reset.

D. Correction of the upper level solution

The correction of an upper level solution by an aggregated solution according to eq. (3) is done in a similar way as the aggregation of a vector by a single depth first traversal of the reachability BDD.

E. Sparse matrix approach

The sparse matrix approach, as proposed in [20] is adapted to the multilevel case in the following way: For speeding up the smoothing steps and aggregation method of the fine system, the lower levels of

```
(1)  agg_matrix(node, row_offset, coarse_row_offset, s_i)
(2)    if (node = ZERO)
(3)      return
(4)    if (level = S)
(5)      increment value of current_ptr by
           vector[row_offset]*node.value
(6)      return
(7)    if (level ≥ D)
(8)      coarse_offsets_change := false
(9)    else
(10)     coarse_offsets_change := true
(11)   if (level = D)
(12)     increment node.aggregate_pointer
(13)     current_ptr := node.aggregate_pointer
(14)   if (not diagonal element)
(15)     recurse on s-variables updating required offsets
(16)     if (level = D)
(17)       normalise value of current_ptr by
             coarse_vector[coarse_row_offset]
(18)       subtract value of current_ptr from
             coarse_diag[coarse_row_offset]
```

Fig. 10. Matrix aggregation algorithm

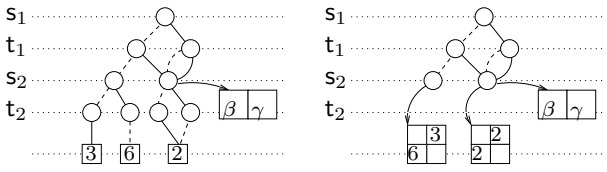


Fig. 11. Adding sparse matrices

the MTBDD are substituted by sparse matrices. In our current implementation of the algorithm, only the nodes up to the first aggregation variable level can be replaced by sparse matrices. C. f. the example in Fig. 11 where the first aggregation level is s_2, so the sparse matrix level cannot exceed this level.

It is important to note that, using our current implementation, the sparse matrix speedup is only achieved for the fine system. All the aggregated systems do not benefit from the sparse matrix speedup, as their levels still have to be processed recursively.

F. Memory considerations

In the current version of the symbolic multilevel algorithm, Jacobi iterations are used for the smoothing steps. So a lower bound for the memory consumption of the multilevel algorithm is the memory consumption of the Jacobi algorithm. The total memory consumption of the multilevel algorithm can be written as

$$matrix + diag + vect + max(vect, agg) + sm \quad (4)$$

where *matrix* is the memory used for the transition matrix of the fine system - without diagonal elements - stored as multi-offset labelled MTBDD, *diag* is the memory for the vector of diagonal elements of the transition matrix of the fine system, *vect* is the memory for one iteration vector of the fine system, *agg* means the memory for additional aggregation information (i.e. $diag_i$ and two times $vect_i$ for each aggregated system i, one temporary iteration vector *temp* and the *rates* for the aggregated transition systems) and the optional sparse matrix memory *sm*. The vector *temp* ist used for the smoothing steps of the aggregated systems. It has the same size as the iteration vector of the finest aggregate. Depending on the number of aggregation levels and their location, the size of *agg* may vary considerably. Aggregation levels next to the root of the graph result in a low memory consumption, aggregation levels next to the terminal nodes result in high memory consumption.

In the current multilevel implementation, the nodes storing the matrix of the transition system are 4 Bytes bigger than the nodes used for Jacobi or pseudo Gauss Seidel iterations (24 Bytes).

IV. Experiments

In the following, experimental results of the MTBDD-based multilevel algorithm are given. The current implementation uses the probabilistic symbolic model checker PRISM as a framework. All measurements were performed on an Intel Xeon 3.0 GHz processor with 2 GByte of main memory.

The following numerical algorithms are compared:

- *JOR*, Jacobi OverRelaxation with relaxation parameter 0.9.
- *PGS* Pseudo Gauss-Seidel [20], a block oriented method which uses the Gauss-Seidel idea on the block level, inside the blocks JOR with overrelaxation parameter 0.9 is used.
- *ML1* Multilevel algorithm using 8 pre- and post-smoothing steps respectively on the fine system and 6 pre- and post-smoothing steps respectively for the aggregated systems.
- *ML2* Multilevel algorithm using 4 pre- and post-smoothing steps respectively on the fine system and 4 pre- and post-smoothing steps respectively for the aggregated systems.

For the multilevel smoothing steps of the fine system and all the aggregated systems, ordinary JOR steps with overrelaxation parameter 0.9 are used.

For every ML experiment the aggregation variable levels for each aggregate are given in parenthesis, starting with the first and ending with the last aggregate. For the aggregation example given in Fig. 5 the algorithm would be called MLx(3,2), $x \in \{1, 2\}$, i.e. the first aggregation reduces the fine system by eliminating s_3, the second aggregation reduces the first aggregate by eliminating s_2. In the MTBDD encoding the transition system, the corresponding t-variables are aggregated as well.

The stopping criterion for all the algorithms is a relative elementwise error smaller than $1.0 \cdot 10^{-6}$. For JOR and PGS this is measured between two consecutive iterations, in the ML-case the post-smoothing step in the fine system is measured.

The tables are organised as follows: In the first

three columns the model characteristics are shown: The *scaling* parameter of the model (e.g. number of tokens for the Kanban system), the number of reachable *states* and the number of *transitions* between the reachable states. Column *algorithm* specifies the numerical method used. In column *ML-cycles* the number of multilevel cycles until convergence is given. This only makes sense for the ML algorithm. Column *steps* gives the number of iteration steps until convergence for JOR and PGS. In the multilevel case the smoothing steps on the fine system are given. In *variable levels* the number of number of s-variables is shown. The next two columns give the number of s- (and t-) variables substituted by sparse matrices beginning from the bottom (*sparse levels*) or from the top (*block levels*) of the MTBDD (the latter is only applicable to the PGS scheme). In column *residual* the maximum norm of the vector $\vec{\pi}Q$ is given. The column *memory (kB)* shows the total memory consumption of the different algorithms in kilobyte, finally the column *time (s)* shows the consumed time until convergence was achieved.

The case studies given in the following subsections are chosen from the standard examples shipped with PRISM. The results are presented in Tables II-IV.

A. Tandem queueing network

This model was originally described in [11]. The remarkable part of the results given in Table II is that the multilevel algorithm converges faster although there are comparably few sparse levels used. Only for the smallest system considered (parameter 200) the standard methods are faster, which is due to the multilevel overhead.

B. Kanban manufacturing system

This model is described in [6]. For this model, initially the aggregation scheme reducing one submodel per aggregation, e.g. aggregation variable levels (37,25,13), was used. However, due to a low sparse level of 12 variables, this scheme did not lead to faster calculation time. So the aggregation levels for the multilevel algorithms were chosen in order to grant the same sparse level for the JOR and ML algorithms. The results are shown in Table III. It can be seen that in all cases the multilevel algorithm is between 1.7-1.9 times slower than the ordinary solvers used. But looking at the total number of iterations until convergence, the ML2 algorithm always takes fewer iterations on the fine system than the ordinary JOR solver. Thus speeding up the aggregation steps and iterations of the aggregated systems, which is part of our future work, the solution times are expected to improve.

C. Flexible manufacturing system

This model is described in [7]. The results are presented in Table IV. Aggregating the model according to the submodel structure with the aggregation variable levels (43,34,20), the number of total smoothing steps of the fine system was reduced for all cases except for parameter 4. However, because of the low sparse level this scheme did not lead to faster calculation time. In order to obtain more competitive results, the aggregation scheme was parametrised such that the same number of sparse levels as with the ordinary JOR algorithm was used. As shown in Table IV, except for the smallest system considered (parameter 4) the ML algorithm leads to fewer iterations of the fine system, but due to the multilevel overhead only for parameter 6 the ML1 algorithm slightly outperforms the standard solvers.

V. Conclusion

In this paper, a symbolic multilevel method has been presented. For that purpose, we introduced the concept of multi-offset labelled MTBDDs. The MTBDD data structure is supplied by additional arrays for storing the aggregated transition matrices in the same MTBDD as the fine system. Symbolic aggregation algorithms for vectors of reachable states and transition matrices are given. First empirical studies indicate that the symbolic multilevel algorithm could be a way to alleviate the state space explosion problem for the numerical solution of Markov chains. The experimental results also indicate that the benefits of the multilevel approach strongly depend on the model used.

References

[1] R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic Decision Diagrams and their Applications. *Form. Meth. in Sys. Design*, 10(2/3):171–206, 1997.

[2] P. Bazan and R. German. Approximate analysis of stochastic models by self-correcting aggregation. In *Proc. of QEST'05*, pages 134–144. IEEE Computer Society Press, 2005.

[3] P. Buchholz. Multilevel solutions for structured Markov chains. *SIAM J. Matrix Anal. Appl.*, 22(2):342–357, 2000.

[4] P. Buchholz and T. Dayar. Comparison of multilevel methods for kronecker-based markovian representations. *Computing*, 73(4):349–371, 2004.

[5] G. Ciardo, G. Luettgen, and R. Siminiceanu. Saturation: An efficient strategy for symbolic state-space generation. In *Proc. TACAS'01*, pages 328–342, Genova, Italy, 2001. Springer, LNCS 2031.

[6] G. Ciardo and M. Tilgner. On the use of Kronecker operators for the solution of generalized stochastic Petri nets. *ICASE Report*, 96(35), 1996.

[7] G. Ciardo and K.S. Trivedi. A decomposition approach for stochastic reward networks. *Performance Evaluation*, 18(1):37–59, 1993.

[8] P.J. Courtois. *Decomposability, queueing and computer system applications*. ACM monograph series, 1977.

[9] M. Fujita, P. McGeer, and J.C.-Y. Yang. Multi-terminal Binary Decision Diagrams: An efficient data structure for matrix representation. *Form. Meth. in Sys. Design*, 10(2/3):149–169, 1997.

[10] H. Hermanns, M. Kwiatkowska, G. Norman, D. Parker, and M. Siegle. On the use of MTBDDs for performability analysis and verification of stochastic systems. *Journal of Logic and Algebraic Programming*, 56(1-2):23–67, 2003.

[11] H. Hermanns, J. Meyer-Kayser, and M. Siegle. Multi terminal binary decision diagrams to represent and analyse continuous-time Markov chains. *Proc. 3rd Int. Workshop on the Num. Sol. of Markov Chains*, pages 188–207, 1999.

[12] G. Horton and S. Leutenegger. A Multi-Level Solution Algorithm for Steady-State Markov Chains. *ACM Performance Evaluation Review*, 22(1):191–200, May 1994. Proceedings of the ACM Sigmetrics and Performance 1994, International Conference on Measurement and Modeling of Computer Systems.

[13] J.R. Koury, D.F. McAllister, and W.J. Stewart. Iterative Methods for Computing Stationary Distributions of Nearly

scaling	states	transistions	algorithm	ML-cycles	steps	variable levels	sparse levels	block levels	residual	memory (kB)	time (s)
200	80601	280599	ML1(16,12,8)	159	2544	17	2	-	$2.2204 \cdot 10^{-16}$	2423.0	92.35
			ML2(16,12,8)	240	1920	17	2	-	$1.1102 \cdot 10^{-16}$	2423.0	91.49
			JOR	-	3670	17	15	-	$2.2204 \cdot 10^{-16}$	2357.3	86.88
			PGS	-	2624	17	11	6	$2.2204 \cdot 10^{-16}$	868.4	75.19
400	321201	1121199	ML1(16,12,8)	341	5456	19	4	-	$2.2204 \cdot 10^{-16}$	5675.0	482.97
			ML2(16,12,8)	524	4192	19	4	-	$2.2204 \cdot 10^{-16}$	5675.0	503.47
			JOR	-	7389	19	14	-	$5.5511 \cdot 10^{-17}$	6101.9	860.11
			PGS	-	5274	19	12	7	$1.1102 \cdot 10^{-16}$	3290.7	682.29
800	1282401	4482399	ML1(16,12,8)	739	11824	21	6	-	$2.2204 \cdot 10^{-16}$	22574.7	3381.51
			ML2(16,12,8)	1133	9064	21	6	-	$2.2204 \cdot 10^{-16}$	22574.7	3659.10
			JOR	-	14880	21	15	-	$1.1102 \cdot 10^{-16}$	23443.7	7429.79
			PGS	-	10614	21	13	8	$2.2204 \cdot 10^{-16}$	22574.7	5686.90

TABLE II: Tandem model

scaling	states	transistions	algorithm	ML-cycles	steps	variable levels	sparse levels	block levels	residual	memory (kB)	time (s)
4	454475	3979850	ML1(13,8)	39	624	48	36	-	$5.4149 \cdot 10^{-10}$	8752.2	67.65
			ML2(13,8)	59	472	48	36	-	$4.8971 \cdot 10^{-10}$	8752.2	70.76
			JOR	-	803	48	36	-	$5.5631 \cdot 10^{-10}$	8721.0	37.28
			PGS	-	402	48	29	19	$5.5785 \cdot 10^{-10}$	5204.0	40.43
5	2546432	24460016	ML1(22,16,8)	54	864	48	27	-	$1.8432 \cdot 10^{-10}$	45701.1	574.47
			ML2(22,16,8)	79	632	48	27	-	$2.1860 \cdot 10^{-10}$	45701.1	582.64
			JOR	-	663	48	27	-	$2.2239 \cdot 10^{-10}$	45660.5	340.59
			PGS	-	573	48	27	19	$2.2321 \cdot 10^{-10}$	25842.1	341.11
6	11261376	115708992	ML1(24,17,9)	72	1152	48	25	-	$9.5764 \cdot 10^{-11}$	198947.8	3536.67
			ML2(24,17,9)	107	856	48	25	-	$9.9801 \cdot 10^{-11}$	198947.8	3692.36
			JOR	-	891	48	25	-	$9.6216 \cdot 10^{-11}$	198901.0	2083.72
			PGS	-	772	48	25	19	$9.4921 \cdot 10^{-11}$	111082.6	2142.28

TABLE III: Kanban model

scaling	states	transistions	algorithm	ML-cycles	steps	variable levels	sparse levels	block levels	residual	memory (kB)	time (s)
4	35910	237120	ML1(15,11,8)	61	976	55	41	-	$9.0486 \cdot 10^{-10}$	2376.9	7.50
			ML2(15,11,8)	90	720	55	41	-	$8.7807 \cdot 10^{-10}$	2376.9	8.17
			JOR	-	803	55	41	-	$8.7390 \cdot 10^{-10}$	2244.6	4.33
			PGS	-	749	55	33	22	$8.6704 \cdot 10^{-10}$	1447.6	5.54
5	152712	1111482	ML1(28,17,8)	56	896	55	28	-	$4.7222 \cdot 10^{-10}$	5184.0	39.31
			ML2(28,17,8)	83	664	55	28	-	$5.9121 \cdot 10^{-10}$	5184.0	41.12
			JOR	-	996	55	28	-	$5.5554 \cdot 10^{-10}$	4959.9	31.53
			PGS	-	936	55	28	22	$5.5231 \cdot 10^{-10}$	3798.6	36.37
6	537768	4205670	ML1(32,16,8)	50	800	55	24	-	$1.5117 \cdot 10^{-10}$	12641.0	132.99
			ML2(32,16,8)	81	648	55	24	-	$2.1612 \cdot 10^{-10}$	12641.0	152.10
			JOR	-	1189	55	24	-	$3.9511 \cdot 10^{-10}$	12304.8	139.23
			PGS	-	1124	55	24	22	$3.8807 \cdot 10^{-10}$	8171.7	152.97

TABLE IV: FMS model

Completely Decomposable Markov Chains. *SIAM Journal on Algebraic and Discrete Methods*, 5(2):164–186, June 1984.

[14] M. Kuntz, M. Siegle, and E. Werner. Symbolic Performance and Dependability Evaluation with the Tool CASPA. In *Europ. Perf. Engineering Workshop*, pages 293–307. LNCS 3236, 2004.

[15] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic model checking in practice: Case studies with PRISM. *ACM Performance Evaluation Review*, 32(4):16–21, 2005.

[16] K. Lampka and M. Siegle. Activity-local Symbolic State Graph Generation for High-Level Stochastic Models. In *Proc. of 13th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB)*, pages 245–263, Nürnberg, Germany, 2006.

[17] K. Lampka and M. Siegle. Analysis of Markov Reward Models unsing Zero-suppressed Multi-terminal BDDs. In *1st. Int. Conf. on Performance Evaluation Methodologies and Tools (Valuetools)*, Pisa, Italy, ACM press, ISBN 1-59593-504-5 (CD edition), 10 pages, 2006.

[18] R. Mehmood. *Disk-Based Techniques for Efficient Solution of Large Markov Chains*. PhD thesis, School of Computer Science, Faculty of Science, University of Birmingham, 2005.

[19] A. Miner and D. Parker. Symbolic representations and analysis of large probabilistic systems. In C. Baier, B. Haverkort, H. Hermanns, J-P. Katoen, and M. Siegle, editors, *Validation of Stochastic Systems: A Guide to Current Research*, volume 2925 of *Lecture Notes in Computer Science (Tutorial Volume)*, pages 296–338. Springer, 2004.

[20] D. Parker. *Implementation of symbolic model checking for probabilistic systems*. PhD thesis, School of Computer Science, Faculty of Science, University of Birmingham, 2002.

[21] PRISM website. `http://www.cs.bham.ac.uk/~dxp/prism/`.

[22] M. Siegle. *Behaviour analysis of communication systems: Compositional modelling, compact representation and analysis of performability properties*. Shaker Verlag, Aachen, 2002.

[23] W.J. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, 1994.

[24] Y. Takahashi. A Lumping Method for Numerical Calculation of Stationary Distributions of Markov Chains. Technical Report B-18, Tokio Institute of Technology, Dpt. of Information Sciences, June 1975.

ABSORBING JOINT MARKOV CHAINS: EXPLOITING COMPOSITIONALITY FOR CUMULATIVE MEASURES

Freimut Brenner
Institute for Computer Science and Business Information Systems (ICB)
Research Group *Systems Modelling*
University of Duisburg-Essen
45127 Essen, Germany
Freimut.Brenner@icb.uni-due.de

ABSTRACT

A novel result on cumulative measures of absorbing joint Markov chains is presented. Provided that the marginal processes are independent continuous time Markov chains, the mean time to absorption and the expected total time in a transient set of the joint Markov chain are computed from the marginal CTMCs in a compositional way. Operations on the state space of the joint Markov chain are never carried out, and hence the problem of state space explosion is avoided. The computational effort of our method rather depends on convergence properties of the joint Markov chain. Numerical examples are given and an application to the solution of Markovian process algebras is briefly sketched.

1 INTRODUCTION

In this paper we consider an absorbing continuous time Markov chain (CTMC) $Y = (Y_1, \ldots, Y_m)$ which is the joint process of independent and absorbing CTMCs $Y_1, \ldots, Y_m$. We develop an algorithm which computes the mean time to absorption and the expected total time spent in some (transient) set in a compositional way, i.e. without operating on the (global) state space of the joint CTMC.

We briefly point out the problem concerning the straight forward way to compute the afore mentioned cumulative measures. Let Q be the generator matrix of Y and assume that T is a sub-matrix which contains all the transition rates between transient states and be α the initial probability distribution of the transient states. Then the mean time to absorption, in the remainder referred to as MTTA, of Y and the expected total time $\mathbb{E}_A$ in a transient set A are given by (see (Neuts 1981))

$$MTTA = -\alpha T^{-1} 1, \quad \mathbb{E}_A = -\alpha T^{-1} 1_A, \qquad (1)$$

where 1 is a column vector consisting of ones only and 1_A is a column vector where entries corresponding to the set A are one, and all other entries are zero. It is clear that the dimension of the sub-matrix T is exponential in the number m of parallel CTMCs, and hence computation of the MTTA and $\mathbb{E}_A$ according to (1) suffers from the state space explosion problem.

Another possibility to compute the MTTA and $\mathbb{E}_A$ is to employ uniformisation. A naive approach, based on directly uniformising Y, also suffers from the state space explosion problem. This becomes apparent later in section 2.2, where this method is briefly discussed. This method is referred to as the direct uniformisation approach in the remainder of this paper. The novel method which we propose is also based on uniformisation but computes certain quantities in a compositional way. The kinship of the two approaches, though, allows to compare the computational effort of our method directly to the effort of the direct uniformisation approach. In (Bohnenkamp 2002) a method which also aims at computing the MTTA in a compositional way was proposed. The computation of $\mathbb{E}_A$ is not possible with that algorithm, and thus it is not discussed here.

2 CUMULATIVE MEASURES

In this section we derive a relation for the mean time to absorption and the mean time spent in some set before absorption of an absorbing CTMC Y which is the joint process of independent absorbing CTMCs $Y_1, \ldots, Y_m$. This relation exhibits a compositional character in the sense that quantities belonging to the marginal chains are combined to yield the desired cumulative measure.

2.1 The Setup

Let $Y_i = (Y_i(t))_{t \in \mathbb{R}_{\geq 0}}$, $i = 1, \ldots, m$, be m independent continuous time homogeneous absorbing Markov chains, where Y_i, $1 \leq i \leq m$, is defined by the finite state space E_i, the starting state $s_i' \in E_i$, the set of absorbing states $S_i \subset E_i$,and the generator matrix $Q_i = (Q_i(j,\ell))_{j,\ell \in E_i}$. With $p_i(t)$ we denote the transient distribution of Y_i at time t.

Now define the Markov chain Y as the joint process $Y = (Y(t))_{t \in \mathbb{R}_{\geq 0}} := (Y_1, \ldots, Y_m)$. Then the state space of Y is given by $E = \times_{i=1}^m E_i$, the starting state is $s' = (s_1', \ldots, s_m')$ and the set of absorbing states is given by $S = \times_{i=1}^m S_i$. That means the joint Markov chain

Print 0-9553018-4-X, CD 0-9553018-3-1

has become absorbed if all of the marginal Markov chains have become absorbed. It is well-known that the generator matrix Q of the joint CTMC Y can be represented by the Kronecker sum $Q = \oplus_{i=1}^{m} Q_i$. Let $p(t)$ denote the transient distribution of Y at time t.

For $A_i \subset E_i$ and $A := \times_{i=1}^{m} A_i$, with $A \cap S = \emptyset$, i. e. A contains only transient states, we are interested in the mean time to absorption of Y (MTTA) and the expected total time Y spends in A before absorption ($\mathbb{E}_A$).

$$\begin{aligned} MTTA &= \int_0^\infty 1 - p(t)(S)dt \\ \text{and} \quad \mathbb{E}_A &= \int_0^\infty p(t)(A)dt. \end{aligned} \tag{2}$$

2.2 Cumulative Measures From Uniformisation

Let $P := I + 1/qQ$, where $q \geq \max_j\{|Q(j,j)|\}$, and $\nu(n) = p(0)P^n$. That means $\nu(n)(\mathcal{A})$ is the probability that Y, if uniformised with rate q, is in $\mathcal{A}$ in the n-th step. Then the transient probability $p(t)(\mathcal{A})$ can be expressed by the well-known uniformisation equation (see (Bolch et al. 1998))

$$p(t)(\mathcal{A}) = \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \nu(n)(\mathcal{A}), \tag{3}$$

where $\nu(0) = p(0)$ and, for $n \geq 1$,

$$\nu(n) = p(0)P^n = \nu(n-1)P. \tag{4}$$

Substitution of $p(t)(S)$ and $p(t)(A)$ in (2) yields

$$\begin{aligned} MTTA &= \frac{1}{q} \sum_{n=0}^{\infty} 1 - \nu(n)(S) \\ \text{and} \quad \mathbb{E}_A &= \frac{1}{q} \sum_{n=0}^{\infty} \nu(n)(A). \end{aligned} \tag{5}$$

Calculating $\nu(n)(S)$ and $\nu(n)(A)$ according to (4) requires a vector-matrix multiplication for every n. Since P is the transition matrix of the joint CTMC $Y = (Y_1, \ldots, Y_m)$, its dimension is exponential in the number m of marginal CTMCs.

The next subsection provides means to compute the values $\nu(n)(S)$ and $\nu(n)(A)$ in a compositional way from certain marginal quantities.

2.3 Main Result

In this subsection the probabilities $p(t)(S)$ and $p(t)(A)$ from (2) are reformulated. In the end this will allow to compute $\nu(n)(S)$ and $\nu(n)(A)$ in (5) without operating on the entire state space of Y. Since both $S = \times_{i=1}^{m} S_i$ and $A = \times_{i=1}^{m} A_i$ are products of marginal sets, it suffices to concentrate on a probability $p(t)(\mathcal{A})$, with $\mathcal{A} = \times_{i=1}^{m} \mathcal{A}_i$, where $\mathcal{A}_i \subseteq E_i$, $i = 1 \ldots m$.

Let $\mathcal{A}_i \subseteq E_i$, $i = 1 \ldots m$, be arbitrary subsets of the marginal state spaces and $\mathcal{A} = \times_{i=1}^{m} \mathcal{A}_i$.

Definition 1. For two discrete functions $f, g : \mathbb{N}_0 \to [0,1] \subset \mathbb{R}$ and constant values $q_f, q_g \in \mathbb{R}_{>0}$ assigned to these functions, define the $\star$-operator by

$$\begin{aligned} (f \star g)(n) &= \sum_{k=0}^{n} \binom{n}{k} \frac{q_f^k q_g^{n-k}}{(q_f + q_g)^n} f(k) g(n-k) \\ \text{and} \quad q_{f \star g} &= q_f + q_g. \end{aligned} \tag{6}$$

Also let $\star_{i=1}^{m} f_i := f_1 \star \cdots \star f_m$.

The following theorem represents the first main result of this paper. For a better readability we postpone the proof of statement (i) until section 3.

Theorem 2. *For $i = 1 \ldots m$, let $P_i := I + 1/q_i Q_i$ and $\nu_i(0) := p_i(0)$, $\nu_i(n) = \nu_i(n-1)P_i$, for $n \geq 1$, where $q_i \geq \max_j\{|Q_i(j,j)|\}$.*

Then, with $\nu_i[\mathcal{A}_i](n) = \nu_i(n)(\mathcal{A}_i)$, $i = 1 \ldots m$, and $q := q_1 + \cdots + q_m$, we have

(i) $p(t)(\mathcal{A}) = \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \left(\star_{i=1}^{m} \nu_i[\mathcal{A}_i]\right)(n)$,

(ii) With $\nu(n)$ from (4), we have

$$\left(\star_{i=1}^{m} \nu_i[\mathcal{A}_i]\right)(n) = \nu(n)(\mathcal{A}), \tag{7}$$

i. e. $\left(\star_{i=1}^{m} \nu_i[\mathcal{A}_i]\right)(n)$ is the probability that Y, if uniformised with rate q, is in $\mathcal{A}$ in the $n-$th step.

Proof. We prove (i) in section 3. To prove (ii) note first that from $q_i \geq \max_j\{|Q_i(j,j)|\}$, $i = 1 \ldots m$, follows $q = \sum_i q_i \geq \max_j\{|Q(j,j)|\}$, which is a consequence of $Q = \oplus_{i=1}^{m} Q_i$. Now, on the one hand $p(t)(\mathcal{A})$ can be expressed by the series in (i) and on the other hand by the uniformisation equation in (3). Equating the right-hand side expressions of (3) and of (i) it follows that, for all $t \geq 0$,

$$\sum_{n=0}^{\infty} \frac{(qt)^n}{n!} \left(\star_{i=1}^{m} \nu_i[\mathcal{A}_i]\right)(n) = \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} \nu(n)(\mathcal{A}) < \infty, \tag{8}$$

where convergence is assured by the fact that $\nu(n)(\mathcal{A}) \in [0,1]$ is a probability. The uniqueness theorem for power series states that if for two series $\sum a_n x^n$ and $\sum b_n x^n$ with positive convergence radii there exists a sequence $(g_k)_{k \geq 0}$, with $0 \neq g_k \to 0$, such that $\sum a_n g_k^n = \sum b_n g_k^n$, for all k, then $a_n = b_n$, for $n \geq 0$. Since (8) holds for all $t \geq 0$, such a sequence obviously exists for both power series in (8), and hence $\frac{\nu(n)(\mathcal{A})}{n!} = \frac{\left(\star_{i=1}^{m} \nu_i[\mathcal{A}_i]\right)(n)}{n!}$. This implies the assertion. □

The definition of the $\star$-operator indicates the possibility of an approximation scheme for the values $\nu(n)(\mathcal{A})$. Note the involvement of the binomial distribution in the sum in (6). In the following we investigate the quality of an approximation to $\nu(n)(\mathcal{A})$ which results from cutting the lower and upper tail of this binomial distribution.

Definition 3. Under the same prerequisites as in definition 1 and for $0 \le \ell < r \le n$ define

$$\begin{aligned}(f\left|\begin{smallmatrix}r\\ \star\\ \ell\end{smallmatrix}\right| g)(n) &= \sum_{k=\ell}^{r}\binom{n}{k}\frac{q_f^k q_g^{n-k}}{(q_f+q_g)^n} f(k)g(n-k)\\ \text{and}\quad q_{f|\star_\ell^r|g} &= q_f+q_g.\end{aligned}$$

We will also use the abbreviation

$$\left|\begin{smallmatrix}\mathbb{r}\\ \star\\ \mathbb{l}\end{smallmatrix}\right|_{i=1}^{m} f_i = f_1\left|\begin{smallmatrix}r_1\\ \star\\ \ell_1\end{smallmatrix}\right| f_2\left|\begin{smallmatrix}r_2\\ \star\\ \ell_2\end{smallmatrix}\right|\cdots\left|\begin{smallmatrix}r_{m-1}\\ \star\\ \ell_{m-1}\end{smallmatrix}\right| f_m,$$

where $\mathbb{l}=(\ell_1,\ldots,\ell_{m-1})$ and $\mathbb{r}=(r_1,\ldots,r_{m-1})$ are multiindices.

Corollary 4. *Consider the same prerequisites as in definition 3. In addition let $\gamma_1,\gamma_2\in\mathbb{R}$, with $0\le\gamma_1,\gamma_2<1$. Let $\ell,r\in\mathbb{N}_0$ be such that*

$$1-\sum_{k=\ell}^{r}\binom{n}{k}\frac{q_f^k q_g^{n-k}}{(q_f+q_g)^n}\le\gamma_1, \tag{9}$$

and let $f':\mathbb{N}_0\to[0,1]\subset\mathbb{R}$ be a discrete function, with

$$0\le f(k)-f'(k)\le\gamma_2,\quad \text{for } 0\le k\le n,$$

Then, the following is true

$$0\le(f\star g)(n)-(f'\left|\begin{smallmatrix}r\\ \star\\ \ell\end{smallmatrix}\right| g)(n)\le\gamma_2(1-\gamma_1)+\gamma_1.$$

Proof. It is clear that $(f\star g)(n)-(f'\left|\star_\ell^r\right| g)(n)\ge 0$. To proof the upper bound, consider

$$\begin{aligned}(f\star g)(n) =& \overbrace{\sum_{k=\ell}^{r}\binom{n}{k}\frac{q_f^k q_g^{n-k}}{(q_f+q_g)^n} f'(k)g(n-k)}^{=(f'\left|\star_\ell^r\right| g)(n)}\\ &+\sum_{k=\ell}^{r}\binom{n}{k}\frac{q_f^k q_g^{n-k}}{(q_f+q_g)^n}(f(k)-f'(k))g(n-k)\\ &+\sum_{k=0}^{\ell-1}\binom{n}{k}\frac{q_f^k q_g^{n-k}}{(q_f+q_g)^n} f(k)g(n-k)\\ &+\sum_{k=r+1}^{n}\binom{n}{k}\frac{q_f^k q_g^{n-k}}{(q_f+q_g)^n} f(k)g(n-k).\end{aligned}$$

The bounds $f,g\le 1$ and $f(k)-f'(k)\le\gamma_2$, for $0\le k\le n$, and the fact that the $\binom{n}{k}\frac{q_f^k q_g^{n-k}}{(q_f+q_g)^n}$ are binomial probabilities, imply

$$\begin{aligned}(f\star g)(n)\le&(f'\left|\begin{smallmatrix}r\\ \star\\ \ell\end{smallmatrix}\right| g)(n)+\gamma_2(1-\gamma_1)\\ &+1-\sum_{k=\ell}^{r}\binom{n}{k}\frac{q_f^k q_g^{n-k}}{(q_f+q_g)^n}.\end{aligned}$$

Together with (9) this concludes the proof. □

In theorem 2 the values $\nu(n)(\mathcal{A})$ were expressed via the $\star$-operator. The following theorem 5 considers the error of approximations $\widetilde{\nu}(n)(\mathcal{A})$ which result by simply replacing the $\star$-operator by the $|\star|$-operator. This is the second main result of this paper.

Theorem 5. *Under the same prerequisites as in theorem 2 Let $\mathbb{l}=(\ell_1,\ldots,\ell_{m-1})$ and $\mathbb{r}=(r_1,\ldots,r_{m-1})$ be multiindices such that, for $i=1\ldots m-1$ and a given $0\le\gamma<1$,*

$$1-\sum_{k=\ell_i}^{r_i}\binom{n}{k}\frac{(\sum_{j=1}^{i}q_j)^k q_{i+1}^{n-k}}{(\sum_{j=1}^{i+1}q_j)^n}\le\gamma,$$

and let

$$\widetilde{\nu}(n)(\mathcal{A}) := (\left|\begin{smallmatrix}\mathbb{r}\\ \star\\ \mathbb{l}\end{smallmatrix}\right|_{i=1}^{m}\nu_i[\mathcal{A}_i])(n).$$

Then the following holds

$$0\le\nu(n)(\mathcal{A})-\widetilde{\nu}(n)(\mathcal{A})\le 1-(1-\gamma)^{m-1}.$$

Proof. We proof this by induction over m. Take $\mathbb{f}=\mathbb{f}':=\nu_1[\mathcal{A}_1]$ and $\mathbb{g}:=\nu_2[\mathcal{A}_2]$. Then corollary 4 implies $0\le(\mathbb{f}\star\mathbb{g})(n)-(\mathbb{f}'\left|\star_{\ell_1}^{r_1}\right|\mathbb{g})\le\gamma$, which is the statement of the current theorem for $m=2$. That means the following induction hypothesis holds:

$$0\le(\star_{i=1}^{m}\nu_i[\mathcal{A}_i])(n)-(\left|\begin{smallmatrix}\mathbb{r}\\ \star\\ \mathbb{l}\end{smallmatrix}\right|_{i=1}^{m}\nu_i[\mathcal{A}_i])(n)\le 1-(1-\gamma)^{m-1}.$$

For the induction step take $f:=(\star_{i=1}^{m}\nu_i[\mathcal{A}_i])$, $g:=\nu_{m+1}[\mathcal{A}_{m+1}]$ and $f':=|\star_{\mathbb{l}}^{\mathbb{r}}|_{i=1}^{m}\nu_i[\mathcal{A}_i]$. That means, it remains to prove

$$0\le(f\star g)(n)-(f'\left|\begin{smallmatrix}r_{m+1}\\ \star\\ \ell_{m+1}\end{smallmatrix}\right| g)(n)\le 1-(1-\gamma)^m.$$

Since by induction hypothesis $0\le f(n)-f'(n)\le 1-(1-\gamma)^{m-1}$, corollary 4 verifies this inequality. □

Remark 1. It is clear that if for a given $0\le\Gamma<1$ we choose $\gamma:=1-\sqrt[m-1]{1-\Gamma}$ in the above theorem, the following bounds are valid

$$0\le\nu(n)(\mathcal{A})-\widetilde{\nu}(n)(\mathcal{A})\le\Gamma.$$

3 PROOF OF THEOREM 2

By independence of the marginal CTMCs

$$p(t)(\mathcal{A}) = p_1(t)(\mathcal{A}_1)\cdots p_m(t)(\mathcal{A}_m). \tag{10}$$

For a given initial distribution $p_i(0)$, some $q_i \geq \max_{j\in E_i}\{|Q_i(j,j)|\}$ and with $P_i := I + 1/q_i Q_i$, the probability $p_i(t)(\mathcal{A}_i)$ can be expressed as the series

$$p_i(t)(\mathcal{A}_i) = \sum_{k=0}^{\infty} \frac{(q_i t)^k}{k!} e^{-q_i t} \nu_i(k)(\mathcal{A}_i), \tag{11}$$

where $\nu_i(0) = p_i(0)$ and $\nu_i(k+1) = \nu_i(k)P_i$, for $k \geq 0$. Throughout this section let

$$\begin{aligned} q &:= q_1 + \cdots + q_m, \\ a_i(n) &:= \nu_i(n)(\mathcal{A}_i), \text{ for } i = 1\ldots m, n \geq 0. \end{aligned}$$

Substituting each $p_i(\mathcal{A}_i)$ in (10) with the corresponding series in (11) yields

$$\begin{aligned} p(t)(\mathcal{A}) &= \prod_{i=1}^{m} \sum_{n_i=0}^{\infty} \frac{(q_i t)^{n_i}}{n_i!} e^{-q_i t} a_i(n_i) \\ &= e^{-qt} \prod_{i=1}^{m} \sum_{n_i=0}^{\infty} \frac{(q_i t)^{n_i}}{n_i!} a_i(n_i) \\ &= e^{-qt} \sum_{n=0}^{\infty} \sum_{\substack{n_i\geq 0,\\ \sum_i n_i = n}} \prod_{i=1}^{m} \frac{(q_i t)^{n_i}}{n_i!} a_i(n_i) \\ &= \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \cdot \sum_{\substack{n_i\geq 0,\\ \sum_i n_i = n}} \left[n! \prod_{i=1}^{m} \frac{q_i^{n_i} a_i(n_i)}{n_i! q^{n_i}} \right]. \end{aligned} \tag{12}$$

The following corollary 6 in combination with (12) proves statement (i) in theorem 2.

Corollary 6. *With the functions $a_i, 1 \leq i \leq m$, and the values $q_{a_i} = q_i$ assigned to them, the following assertion holds*

$$(\star_{i=1}^{m} a_i)(n) = \sum_{\substack{n_i\geq 0,\\ \sum_i n_i = n}} n! \prod_{i=1}^{m} \left[\frac{q_i^{n_i} a_i(n_i)}{n_i! q^{n_i}} \right].$$

Proof. We proof this by induction over m. For $m = 1$ the assertion is trivially true and for $m = 2$ it is easily verified that

$$(\star_{i=1}^{m} a_i)(k) = \sum_{\substack{n_i\geq 0,\\ \sum_{i=1}^m n_i = k}} k! \prod_{i=1}^{m} \left[\frac{q_i^{n_i} a_i(n_i)}{n_i! \left(\sum_1^m q_\ell\right)^{n_i}} \right],$$

which provides us with a valid induction hypothesis. For the induction step we write

$$\begin{aligned} &(a_1 \star a_2 \star \cdots \star a_{m+1})(n) \\ &= \sum_{k=0}^{n} \binom{n}{k} \frac{\left(\sum_1^m q_i\right)^k q_{m+1}^{n-k}}{\left(\sum_i^{m+1} q_i\right)^n} (\star_1^m a_i)(k) a_{m+1}(n-k). \end{aligned} \tag{13}$$

Now substitute $(\star_1^m a_i)(k)$ in (13) with the induction hypothesis and afterwards combine the resulting two sums using the substitution $n - k = n_{m+1}$:

$$\begin{aligned} &(a_1 \star a_2 \star \cdots \star a_{m+1})(n) \\ &\quad = \sum_{\substack{n_i\geq 0,\\ \sum_{i=1}^{m+1} = n}} \binom{n}{n-n_{m+1}} \frac{\left(\sum_1^m q_i\right)^{n-n_{m+1}} q_{m+1}^{n_{m+1}}}{\left(\sum_i^{m+1} q_i\right)^n} \\ &\qquad \cdot (n-n_{m+1})! \prod_{i=1}^{m} \left[\frac{q_i^{n_i} a_i(n_i)}{n_i! \left(\sum_1^m q_\ell\right)^{n_i}} \right] a_{m+1}(n_{m+1}). \end{aligned}$$

Extending the range of the product sign to $m+1$ and proper rearrangement of the terms yields the assertion. □

4 ALGORITHM

In order to compute the MTTA and $\mathbb{E}_A$ in (5), a truncation index N must be introduced, i. e. we actually compute

$$\begin{aligned} MTTA(N) &= \frac{1}{q} \sum_{n=0}^{N} 1 - \nu(n)(S) \\ \text{and} \quad \mathbb{E}_A(N) &= \frac{1}{q} \sum_{n=0}^{N} \nu(n)(A), \end{aligned} \tag{14}$$

The aim of this section is to provide algorithms which compute, approximate respectively, the values $\nu(n)(S)$ and $\nu(n)(A)$ by employing the $\star$-operator, $|\star|$-operator respectively. The application of these operators requires knowledge of certain binomial probabilities. In section 4.1 we show how these probabilities can be computed. Section 4.2 contains the actual algorithms which we refer to as Compositional Uniformisation (CU). Section 4.3 deals with complexity issues.

Remark 2. It is well-known that the distance of the quantities $\nu(n)(A)$ and $\nu(n)(S)$ to the stationary probabilities $\nu(\infty)(A) = 0$ and $\nu(\infty)(S) = 1$ decays exponentially fast (see e.g (Rosenthal 1995)). Hence, from the values $\nu(n)(A)$ and $1 - \nu(n)(S)$, $n = 0\ldots N$, the exponential decay rates could be estimated, such that it would also be possible to estimate or bound the truncated remainders in (14).

4.1 Binomial Probabilities

Let $b(\cdot;n,p)$ be a binomial distribution with

$$b(k;n,p) = \binom{n}{k} p^k (1-p)^{n-k} \quad \text{and} \quad 0 < p < 1.$$

Assume $0 \le k \le n$. Then the following is trivial:

$$\begin{aligned}
b(k;n,p) &= b(k-1;n,p)\frac{(n-k+1)p}{k(1-p)}, \quad \text{for } k \ge 1,\\
b(k;n,p) &= b(k+1;n,p)\frac{(k+1)(1-p)}{(n-k)p}, \quad \text{for } k \le n-1,\\
b(k;n,p) &= b(k;n-1,p)\frac{n(1-p)}{n-k}, \quad \text{for } k \le n-1,\\
b(k;n,p) &= b(k-1;n-1,p)\frac{np}{k}, \quad \text{for } k \ge 1.
\end{aligned} \tag{15}$$

The mode of $b(\cdot;n,p)$ is given by $M := \lfloor (n+1)p \rfloor$. For the modes M and M' of $b(\cdot;n,p)$ and $b(\cdot;n+1,p)$, we have

$$M' - M \in \{0,1\}.$$

When computing binomial probabilities, one might want to avoid to compute extremely low probabilities. Fortunately, for a given $\gamma > 0$, it is possible to determine values ℓ_n and r_n in advance, such that $\sum_{k=\ell_n}^{r_n} b(k;n,p) \ge 1-\gamma$. For example, to find the truncation index ℓ_n for the lower tail, consider a random variable $X \sim b(\cdot;n,p)$ and the Chernoff bound (see (Motwani and Raghavan 1995))

$$\begin{aligned}
P(X < (1-\delta)np) &\le e^{-\delta^2 np/2},\\
\text{or equivalently } P(X < \ell_n) &\le e^{-\frac{(np-\ell_n)^2}{2np}},
\end{aligned} \tag{16}$$

for $\delta > 0$ and $\ell_n < np$.

In order to compute r_n, note that $b(k;n,p) = b(n-k;n,1-p)$. With a random variable $X' \sim b(\cdot;n,1-p)$ we obtain, for $r_n + 1 > np$,

$$P(X > r_n) = P(X' < n - (r_n+1)).$$

Lets say $\gamma = \alpha + \beta$. Then

$$\begin{aligned}
\ell_n &= \max\left\{0, \left\lfloor np - \sqrt{-2np\ln\alpha} \right\rfloor\right\},\\
r_n &= \min\left\{n, \left\lceil np + \sqrt{-2n(1-p)\ln\beta} - 1 \right\rceil\right\}
\end{aligned} \tag{17}$$

satisfy

$$P(\ell_n \le X \le r_n) > 1-\gamma,$$

or equivalently

$$1 - \sum_{k=\ell_n}^{r_n} b(k;n,p) \le \gamma.$$

Other bounds in the style of (16) exist and can be used as well to determine ℓ_n and r_n. E.g. Hoeffding's inequality ((Hoeffding 1963)) yields the bound $P(X \le \ell_n) \le e^{-2\frac{(np-\ell_n)^2}{n}}$, for $\ell_n < np$, which apparently is a tighter bound than (16) if $p > 1/4$.

With these introductory explanations the following algorithm is almost self-explaining. For given numbers ℓ_n and r_n, the n-th pass of the outer loop computes the binomial probabilities $b(k;n,p)$, $k = \ell_n \ldots r_n$, according to (15). In order to reduce the effect of possible underflow, the algorithm starts with the computation of the mode M' and its corresponding binomial probability $b(M';n,p)$, for each n.

Binomial Probabilities:
$b(0;0,p) = 1$;
For $n = 1 \ldots N$ do:
  $M = \lfloor np \rfloor$;
  $M' = \lfloor (n+1)p \rfloor$;
  if$(M' > M)$
    $b(M',n,p) = b(M,n-1,p)\frac{np}{M'}$;
  else
    $b(M',n,p) = b(M,n-1,p)\frac{n(1-p)}{n-M'}$;
  For $k = M+1 \ldots r$ do:
    $b(k;n,p) = b(k-1;n,p)\frac{(n-(k-1))p}{k(1-p)}$;
  For $k = M-1 \ldots \ell$ do:
    $b(k;n,p) = b(k+1;n,p)\frac{(k+1)(1-p)}{(n-k)p}$;
  – use $b(\ell;n,p), \cdots, b(r;n,p)$ in application –
  store $b(M';n,p)$; delete all other $b(\cdot;n,p)$;

From (17) it is seen that, for every positive γ, $r_n - \ell_n \in O(\sqrt{n})$. Hence, the $n-$th pass of the outer loop requires time $O(\sqrt{n})$. The time complexity of the overall algorithm is in

$$O(N^{3/2}).$$

4.2 Compositional Uniformisation

Instead of dealing with the two sets A and S, consider the set $\mathcal{A} = \times_{i=1}^m \mathcal{A}_i$ and the computation of the quantities $\left(\star_{k=1}^m \nu_k[\mathcal{A}_k]\right)(n) = \nu(n)(\mathcal{A})$.

The first algorithm (Exact CU) computes the exact values $\nu(n)(\mathcal{A})$. The second algorithm (Approximate CU) computes approximate values $\widetilde{\nu}(n)(\mathcal{A})$, such that for a given $\gamma > 0$, we have $0 \le \nu(n)(\mathcal{A}) - \widetilde{\nu}(n)(\mathcal{A}) \le 1 - (1-\gamma)^{m-1}$.

Exact CU:
For $n = 0 \ldots N$ do:
(b) $\nu_i[\mathcal{A}_i](n) = \sum_{z \in \mathcal{A}_i} \nu_i(n)(z)$, for $i = 1 \ldots m$
(a) $\nu_i(n+1) = \nu_i(n) P_i$, for $i = 1 \ldots m$
(c) compute $f_m(n) = \left(\star_{k=1}^{m} \nu_k[\mathcal{A}_k]\right)(n)$ iteratively:

$f_1(n) := \nu_1[\mathcal{A}_1](n)$;
For $i = 2 \ldots m$ do:

(i) $1 - p = q_i / \sum_{j=1}^{i} q_j$;
(ii) compute $b(\cdot; n, p)$;
(iii) $f_i(n) = (f_{i-1} \star \nu_i[\mathcal{A}_i])(n)$;
(iv) $q_{f_i} = q_1 + \cdots + q_i$;
(v) store $f_i(n)$;

Approximate CU:
choose an error parameter $\gamma > 0$;

In the exact algorithm replace (ii) and (iii) by:

(ii) find ℓ_n, r_n, with $1 - \sum_{k=\ell_n}^{r_n} b(k; n, p) \leq \gamma$;
compute $b(k; n, p), \ell_n \leq k \leq r_n$;
(iii) $f_i(n) = \left(f_{i-1} \left|{}_{\ell_n}^{r_n}\star\right| \nu_i[\mathcal{A}_i] \right)(n)$;

Then the n-th pass of the outer loop yields:
$\widetilde{\nu}(n)(\mathcal{A}) := f_m(n) = \left(|\star_{\mathbb{I}}^{\mathbb{r}}|_{k=1}^{m} \nu_k[\mathcal{A}_k]\right)(n)$.

4.3 Complexity

Let $d_i = dim(P_i)$, $i = 1 \ldots m$. The storage requirements for the matrices P_i and the vectors $\nu_i(n)$ and $\nu_i(n-1)$, $i = 1 \ldots m$, are $O\left(\sum_{i=1}^{m} d_i^2\right)$ for every $n > 0$. The values $f_i(n)$, $n = 0 \ldots N$, $i = 1 \ldots m$, must be stored permanently, which requires space $O(mN)$. Hence the overall storage requirement of the algorithm lies in $O\left(\sum_{i=1}^{m} d_i^2 + mN\right)$.

For every $n \in \{0, \ldots, N\}$, the number of operations in the above algorithms are:

exact CU:

$$(a)\ O\left(\sum_{i=1}^{m} d_i^2\right),\quad (b)\ O\left(\sum_{i=1}^{m} d_i\right),\quad (c)\ O(mn).$$

approximate CU:

$$(a)\ O\left(\sum_{i=1}^{m} d_i^2\right),\quad (b)\ O\left(\sum_{i=1}^{m} d_i\right),\quad (c)\ O(m\sqrt{n}).$$

After N iteration steps, we arrive at the overall time complexities

exact CU:

$$O\left(N \sum_{i=1}^{m} d_i^2 + mN^2\right) = O\left(N\left(\sum_{i=1}^{m} d_i^2 + mN\right)\right).$$

approximate CU:

$$O\left(N \sum_{i=1}^{m} d_i^2 + mN^{3/2}\right) = O\left(N\left(\sum_{i=1}^{m} d_i^2 + m\sqrt{N}\right)\right). \tag{18}$$

Remark 3. If the quantities $\nu(n)(\mathcal{A})$ are computed employing (4), i. e. following the approach we have called direct uniformisation, then for every $n \in \{1, \ldots, N\}$, the vector matrix multiplication $\nu = \nu(n-1)P$ must be performed, such that the overall time complexity is $O(N \cdot D^2)$, where D is the dimension of the matrix $P = I + 1/qQ$. Since $Q = \oplus_{i=1}^{m} Q_i$, we have

$$O\left(N\left(\prod_{i=1}^{m} d_i\right)^2\right). \tag{19}$$

We now compare (19) to the right hand sides of (18). For $d_i > 1$ and for reasonably large m, we can assume $\sum_{i=1}^{m} d_i^2 \ll \left(\prod_{i=1}^{m} d_i\right)^2$. Hence our novel method becomes significant faster than the approach based on direct uniformisation if in the exact CU method $mN \ll \left(\prod_{i=1}^{m} d_i\right)^2$, and in the approximate CU method $m\sqrt{N} \ll \left(\prod_{i=1}^{m} d_i\right)^2$.

5 SOME NUMERICAL EXAMPLES

Consider an absorbing joint CTMC $Y = (Y_1, \ldots, Y_m)$ in the context of two examples. The generator matrices of the marginal CTMCS Y_i are given by

example 1:
$$Q_i = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -2 & 2 & 0 & 0 \\ 0 & 0 & -3 & 3 & 0 \\ 4 & 0 & 0 & -8 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

example 2:
$$Q_i = \begin{pmatrix} -7 & 1 & 2 & 3 & 1 \\ 2 & -11 & 3 & 4 & 2 \\ 3 & 4 & -15 & 5 & 3 \\ 4 & 5 & 6 & -19 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and in both examples the initial state of each marginal CTMC is the state 1. For both examples set $A_i := \{1, 3\}$ and $A = \times_{i=1}^{m} A_i$. That means $\mathbb{E}_A$ is the expected total time, where all the marginal CTMCs are in state 1 or 3 at the same time. We remark that in all of our computations we did not exploit the fact that the generator matrices are the same for all marginal CTMCs.

5.1 Three Iterative Methods

We computed the MTTA and $\mathbb{E}_A$ using the exact and the approximate version of our novel approach which in both cases we refer to as Compositional Uniformisation (CU). Furthermore, we carried out computa-

tions with the approach based on Direct Uniformisation (DU) and the method of Jacobi Stepping (JS).

DU and JS were implemented in MATLAB 7.1 such that the optimised built-in functions of MATLAB for vector-matrix multiplications were employed. All matrices were implemented as sparse matrices. CU was implemented in JAVA.

Compositional And Direct Uniformisation For CU and DU the truncation index N was chosen as the smallest number where the relative change of two successive estimates for the MTTA and $\mathbb{E}_A$ became smaller than $\varepsilon = 10^{-6}$, i.e.

$$\frac{MTTA(N) - MTTA(N-1)}{MTTA(N)} < \varepsilon,$$
$$\frac{\mathbb{E}_A(N) - \mathbb{E}_A(N-1)}{\mathbb{E}_A(N)} < \varepsilon.$$

For the approximate version of CU we chose the error parameter $\gamma = 1 - \sqrt[m-1]{1-10^6}$ (cp. remark 1), if dealing with m parallel Markov chains. That means, instead of the values $\nu(n)(\mathcal{A})$, $\mathcal{A} \in \{A, S\}$, which are needed to compute $\mathbb{E}_A$ and the MTTA, we computed approximations $\widetilde{\nu}(n)(\mathcal{A})$, with

$$0 \leq \nu(n)(\mathcal{A}) - \widetilde{\nu}(n)(\mathcal{A}) \leq 10^{-6}.$$

Jacobi Stepping In order to apply the Jacobi method note that according to (1) the MTTA and $\mathbb{E}_A$ can be determined by at first solving

$$xT = \alpha$$

and afterwards computing

$$MTTA = -x1 \quad \text{and} \quad \mathbb{E}_A = -x1_A.$$

To compute x the following iteration scheme is used

$$x^{(k+1)} = (\alpha - x^{(k)}(T - T_D))T_D^{-1},$$

where T_D is the diagonal matrix containing the diagonal entries of T. The starting vector $x^{(0)}$ was chosen as a vector consisting of ones only. We stopped the iteration at the smallest index K, where both of the following criterions were satisfied:

$$\frac{|-x^{(K)}1 + x^{(K-1)}1|}{|x^{(K)}1|} < \varepsilon, \quad \frac{|-x^{(K)}1_A + x^{(K-1)}1_A|}{|x^{(K)}1_A|} < \varepsilon, \tag{20}$$

i.e. the relative change of the estimates of the MTTA and $\mathbb{E}_A$ obtained from two successive iteration vectors becomes smaller than $\varepsilon = 10^{-6}$.

To check the quality of this criterion we conducted additional runs and determined K by evaluating the residual error, i.e. here K is the smallest index satisfying $\| \alpha - x^{(K)}T \|_\infty < \varepsilon$. We found that in comparison to (20) the number of iteration steps differed about at most ± 5. However, in order to minimise the processing time of the Jacobi method, for the results in the tables and graphs below the stopping criterion (20) was used.

5.2 Results

All of our computations were carried out on a system equipped with an Intel Pentium M 740 processor (1.73 GHz) and 1 GB main memory.

For the Jacobi method and the method of direct uniformisation computations were carried out for the values $m = 2 \ldots 8$. All required matrices and vectors were kept in the main memory using an explicit representation (as sparse structures). For greater values of m the required space exceeded the available main memory. We note that due to the representation of Q as the Kronecker sum $Q = \oplus_{i=1}^{m} Q_i$ it would be possible to access the entries of Q without explicitly storing the entire matrix, such that storage requirements are not really the bottleneck. However, such a procedure would not reduce the computation times of these methods. For the CU method m ranged from 2 to 20, where the memory consumption was negligible.

Table 1 and table 2 show the numerical results (truncated after the first 4 post decimal positions) for the examples 1 and 2. CU refers to both the exact and the approximate version of compositional uniformisation. In the first example the results of the exact and the approximate CU had at least 6 matching post decimal positions, for all considered values of m and the given error parameter γ. In the second example we counted at least 7 matching post decimal positions. Furthermore, the truncation index N was the same for the exact and the approximate version of the CU method, for all considered m.

Table 1: Numerical Results for Example 1

		m=2	m=4	m=6	m=8	m=10	m=20
JS	MTTA	5.5407	7.4129	8.5882	9.4467		
	$\mathbb{E}_A$	1.1384	0.4473	0.2542	0.1718		
	steps K	100	149	196	243		
DU	same results as CU below						
CU	MTTA	5.5404	7.4121	8.5867	9.4446	10.1205	12.2599
	$\mathbb{E}_A$	1.1384	0.4473	0.2542	0.1718	0.1283	0.0559
	steps N	581	1133	1675	2213	2747	5386

Table 2: Numerical Results for Example 2

		m=2	m=4	m=6	m=8	m=10	m=20
JS	MTTA	0.7370	1.0053	1.1730	1.2953		
	$\mathbb{E}_A$	0.1411	0.0608	0.0381	0.0276		
	steps K	93	134	174	212		
DU	same results as CU below						
CU	MTTA	0.7370	1.0052	1.1729	1.2952	1.3916	1.6967
	$\mathbb{E}_A$	0.1411	0.0608	0.0381	0.0276	0.0217	0.0104
	steps N	201	395	587	776	965	1897

In the figures 1 and 2 the computation times of the three methods CU, DU and JS are compared. Since the dimensions of the matrices P and Q which are being operated on in the methods DU and JS in each iteration step grow exponentially in the degree m of parallelism, so do the computation times (note the log scaled Y-axis). In contrast to that, the computation times of the CU methods grow much slower.

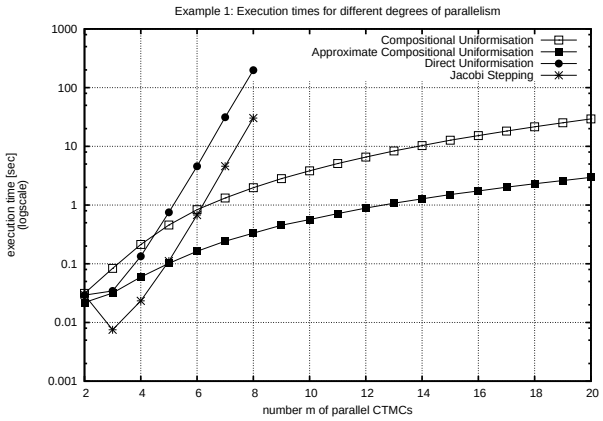


Figure 1: Example 1. Computation Times (in Seconds) for Different Degrees m of Parallelism

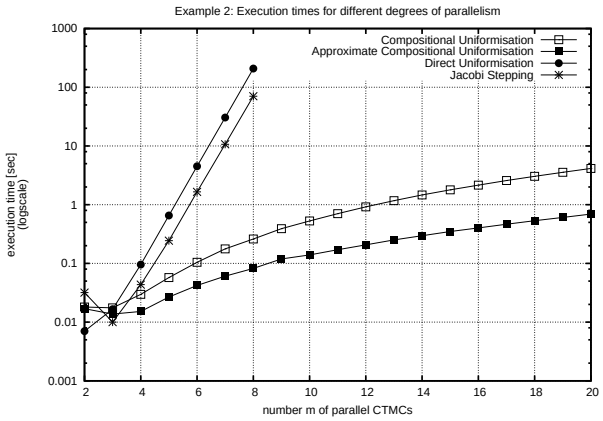


Figure 2: Example 2. Computation Times (in Seconds) for Different Degrees m of Parallelism

6 OUTLOOK: APPLICATION TO MARKOVIAN PROCESS ALGEBRAS

In this section we briefly sketch how we aim at applying our novel method to the field of Markovian process algebras in future work. In a) we recall some basics from Markov renewal theory, b) relates this to an existing approach for solving Markovian process algebras and in c) we indicate how our results could be used to extend this approach.

a) Let Z be a CTMC with state space E and be $S \subset E$ some embedded set in the state space. Let $T_0 = 0, T_1, T_2, \ldots$ denote successive time instants where Z enters the set S. Define

$$\begin{aligned} X(n) &= Z(T_n) \quad \text{embedded DTMC,} \\ \pi &\quad \text{stationary distribution of } X, \\ J(t) &= Z(T_n) \Longleftrightarrow T_n \le t < T_{n+1} \quad \text{semi-Markov chain} \\ &\qquad\qquad\qquad\qquad\qquad \text{embedded in } Z, \\ m_s &= \int_0^\infty \mathbb{P}(T_1 > t | Z(0) = s) dt. \end{aligned}$$

The steady state probability that the semi-Markov chain J is in k is given by ((Çinlar 1975))

$$\mathbb{P}(J = k) = \frac{\pi(k) m_k}{\sum_{s \in S} \pi(s) m_s}. \tag{21}$$

Also from (Çinlar 1975) we cite the following result from the theory of regenerative processes

$$\mathbb{P}(Z \in A) = \frac{\sum_{s \in S} \pi(s) \int_0^\infty \mathbb{P}(Z_t \in A, T_1 > t | Z(0) = s) dt}{\sum_{s \in S} \pi(s) m_s}. \tag{22}$$

For convenience assume that if $Z(0) = s$ the next embedded state that can be reached is different from s. Hence, if for a given state s we derive a Markov chain $Z^{(s)}$ from Z by declaring the states $S \setminus \{s\}$ as absorbing states, then m_s is the mean time to absorption (i.e. MTTA) of $Z^{(s)}$. Furthermore, $\int_0^\infty \mathbb{P}(Z_t \in A, T_1 > t | Z(0) = s) dt$ is the expected total time that $Z^{(s)}$ spends in set A (i.e. $\mathbb{E}_A$).

b) In (Bohnenkamp 2002) Bohnenkamp proposed a method to solve Markovian process algebra (MPA) models in a compositional way. He applied equation (21) to derive steady state information of MPA models. In that context Z is the CTMC underlying the considered MPA model and the embedded states S are states associated with synchronisation. For a more detailed description we refer to that paper. Bohnenkamp observed that under certain circumstances imposed on the MPA model, $Z^{(s)}$ can be expressed as the joint process of several absorbing Markov chains $Z_1^{(s)}, \ldots, Z_m^{(s)}$, i.e. $Z^{(s)} := (Z_1^{(s)}, \ldots, Z_m^{(s)})$. He proposed an algorithm to compute the mean time to absorption of $Z^{(s)}$, i.e. with the above notation m_s. Doing this for every state s and after determi-

nation of the steady state distribution π of the embedded DTMC X, the steady state probabilities of the semi-Markov chain J can be calculated according to (21). The drawback of this method is that only the embedded semi-Markov chain J is solved, but not the CTMC Z which underlies the considered MPA model.

c) If in addition to the quantities required for Bohnenkamp's method we calculate the expected total time $\mathbb{E}_A$ of $Z^{(s)}$, for every $s \in S$, then the steady state probability $\mathbb{P}(Z \in A)$ of the CTMC Z can be determined according to (22). Calculation of $\mathbb{E}_A$ of $Z^{(s)}$ can be achieved with the algorithm presented in this paper, if the set A is the product of suited marginal sets $A_1, \ldots, A_m$ and does not contain absorbing states.

7 CONCLUSION

In this paper a novel compositional method for the computation of the mean time to absorption and the expected total time in some set of an absorbing continuous time joint Markov chain with independent marginal CTMCs is presented. In contrast to traditional techniques the generator of the joint Markov chain needs not to be constructed nor being operated on. Thus, the proposed algorithm is not subject to the state space explosion problem, but rather depends on the number of steps of the joint CTMC until absorption. It is based on at first processing the marginal CTMCs in isolation via a uniformization procedure. Afterwards the marginal results are combined via a convolution-like operator. Numerical examples demonstrate a promising speed-up of the proposed algorithm in comparison with traditional techniques. A possible application to the area of Markovian process algebras is briefly sketched.

FREIMUT BRENNER studied computer science at the University of Trier, where he obtained his diploma in 2005. Since then he has been a member of the systems modelling research group at the University of Duisburg-Essen supervised by Prof. Dr. Müller-Clostermann.

References

H. Bohnenkamp. 2002. *Compositional Solution of Stochastic Process Algebra Models* (PhD thesis, Department of Computer Science, Rheinisch-Westfälische Technische Hochschule Aachen, Germany).

G. Bolch, S. Greiner, H. de Meer, K. S. Trivedi. 1998. *Queueing Networks and Markov Chains – Modeling and Performance Evaluation with Computer Science Applications* (John Wiley & sons, Inc.).

E. Çinlar. 1975. *Introduction to Stochastic Processes* (Prentice-Hall, Inc., Englewood Cliffs, N.J.).

W. Hoeffding. 1963. "Probability inequalities for sums of bounded random variables" (In *Journal of American Statistical Association*, 58, pages 13-30).

R. Motwani, P. Raghavan. 1995. *Randomized Algorithms* (Cambridge University Press).

M. F. Neuts. 1981. *Matrix-Geometric Solutions in Stochastic Models – An Algorithmic Approach* (The Johns Hopkins University Press, Baltimore/London).

J. S. Rosenthal. 1995. "Convergence rates of Markov chains" (In *SIAM Review*, 37, pages 387-405).

SESSION 6

Network Models and Networking Issues

Modeling File Popularity in Peer-to-Peer File Sharing Systems

‡Raffaele Bolla, †Mirko Eickhoff, †Krys Pawlikowski, ‡Michele Sciuto

‡Dept of Communication, Computer and System Sciences, University of Genoa, Italy
{*raffaele.bolla, michele.sciuto*}*@unige.it*
†Dept of Computer Science and Software Engineering, University of Canterbury, New Zealand
m.eickhoff@cosc.canterbury.ac.nz, krys.pawlikowski@canterbury.ac.nz

Abstract— **From the birth of Peer-to-Peer (P2P) protocols, Internet traffic has changed its characteristics; the amount of traffic carried by the web has grown exponentially with strong effects on traffic matrixes, both for the distributed nature of P2P protocols and for their intense bandwitdh request. Many research works studied the features and the behaviors of such P2P protocols, simulating the overlay network topologies at the application layer. We present a scalable P2P model, which is particularly focused on file popularity, considered as the level of interest that a file arouses in the system. The work has the aim of describing the effects that such a quantity generates in the query behavior, assuming that it is strongly linked to marketing procedures. For this reason we make use of the product life cycle in marketing environment to model such level of interest. The model can represent huge overlay networks, up to 500000 peers. The properties of such a model are studied considering the resulting file query distribution, by running multiple independent replications for every simulation. The number of queries performed by every peer and the peer' uptime are also considered, in order to validate the model, comparing simulation output with the measurement results in Gnutella network.**

Keywords— **Distributed systems; File-sharing applications; Peer-to-Peer modeling; Traffic simulation; Internet traffic; Evolution of quantiles.**

I. Introduction

In the last years, Internet technologies and infrastructures have experienced deep advancements and evolutions to meet the increasing user requirements and to support new application requests. It is well known that the major part of traffic carried by Internet is related with a relatively small percentage of classical Internet data applications (i.e., Web, e-mail, FTP, etc.), and that it is basically generated by file-sharing applications. Reference [1] outlines that this kind of applications, based on the P2P technology, is generally characterized by a fast and epidemic spreading and a bandwidth-intensive nature, and generates about 60-80% of the overall Internet traffic. In this scenario it is really important modeling/simulating this kind of applications. One of the difficulties in developing effective P2P simulators/models is the large size of real overlay networks, so the scalability is a fundamental issue. With this respect, we have already presented in [2] a basic framework of a P2P simulator, mainly focused on scalability, which is able to represent huge overlay networks (we have successfully tested it with up to 500000 peers). In this work we are mainly interested in reporting an important improvement of that first proposal obtained by introducing a new and more detailed model for the query generation process , which can better represent the file popularity dynamic, without decreasing system scalability. The query generation process is one of the most significant elements of a P2P model. Its correct representation strongly depends on the precise description of file popularity dynamic, i.e. basically the evolution of the user interest with respect to different files. Many modeling and simulation works, such as [3] - [9], can be found about the Gnutella network and, in general, about P2P overlay networks, but none of them describes the popularity evolution with sufficiently accurate results from simulations or measurements. An interesting way of modeling file popularity can be found in [10], where two different classes of files are taken into account, for generating different file request arrival rates. In a P2P system, such as Gnutella, most of files belong to music or video applications, as reported in [11]. These files arouse a lively interest mainly because they can be obtained for free through the P2P system, even if they have a value on the conventional market. So, the success of P2P systems has to be considered not just in reliability and scalability [12], which improve the sharing of contents, but it has also to be found in economic reasons. Interesting goods are freely available in the network, besides the download processes are faster than in client/server systems and they are based on a reliable service. This motivated us to take marketing issues into account when modeling file popularity in a P2P system. Measurements performed by Reza et al. (see [11]) report a particular and interesting behaviour of the file popularity, which has many similarities with the product life cycle (PLC) behaviour reported in marketing literature [13]. For this reason, we have decided to use the product life cycle adopted in marketing environment to model the level of interest for the files. In this work we describe this new approach and we report some numerical results that show the effectiveness of our proposal.

The paper is organized as follows. Section II describes the proposed P2P simulation model. Section III presents the model results, with a focus on the validity of the model. Section IV explains the statistical analysis we are using for studying results, running multiple parallel replications and considering the time evolution of quantiles. In Section V we show the computation time and the utilization of memory required by the simulator. Finally, in Section VI the conclusions are reported.

II. Model Description

In this section we present the features of the proposed model; the general framework has been already presented in [2]. The application layer has been enriched with some new features, in order to represent P2P traffic in a complete way. The most important improvements regard the birth of new files in the system and the introduction of a new model for file popularity and query mechanism. We took Gnutella 0.6 as a reference system, which relies on a hierarchical unstructured overlay network. We decided to model the Gnutella protocol because it is one of the most used P2P protocols and also because it is well described in literature; moreover there are many measurement results in Gnutella network [11], [14] and [15] we found useful both as input data and for validation. In this Section we provide a general description of the model; more detail can be found in [2].

A. Simulator General Description

The most significant overlay network elements of a file-sharing application are peers, ultra-peers and files. Peers represent the leaves of the overlay network; ultra-peers have the same properties of peers, but they also have to satisfy the queries; files represent the content, which interests peers; see [2]. The developed simulator tries to represent two dynamic processes of a Gnutella P2P network: the overlay network evolution process and the query process. The first one models the ingresses and egresses of peers in and out of the overlay network. The arrival of a new peer causes the creation of new relationships in the overlay network and the potential associations of newly shared file copies to the peer. On the contrary, when a peer exits the overlay network, all its relationships and its shared files have to be erased from the system. The query process is reported in one of the next sections. A new dynamic and more precise model for file popularity than in [2] has been introduced. The structure of the simulation tool can be divided roughly into: the initialization procedure, the overlay network and its dynamics, the birth of new files in the system, the file popularity model, and the query process model.

B. Initialization Procedure

In this Subsection we describe the initialization of the main quantities of the simulator:

- total number of peers and ultra-peers,
- total number of different files in the networks,
- number of file replicas,
- peer and ultra-peer relationships.

The initial numbers of peers and ultra-peers are input parameters. The initial number of replicas is a specific parameter associated with each different file. We assign this parameter to each file, according to the Zipf distribution, as reported in [6]. For instance, starting with 5000 files, the resulting number of total files, considering also replicas is 103387. It is assumed that each peer can maintain no more than one copy of each file. In this way it is clear that the maximum number of replicas for every file is bounded by the total number of peers. We consider an average number of 15 neighbors for every ultra-peer. Each peer has one relationship with an ultra-peer. This ultra-peer is selected randomly with uniform probability among the ones that have less than 150 connected leaves. The overlay network is initialized considering the configuration parameters, and with the strong boundary that every ultra-peer needs at least one neighbor, in order to provide connection for every peer in the network. At this step we initialize the simulator using a specific developed software; the measures performed by Reza et al. in [11], and their conclusions, gave us the input to use real parts of the Gnutella network for initializing the simulator. We actually can acquire real portion of the Gnutella network with a crawler, which we developed for teletraffic classification (see [16]).

C. Overlay Network and its Dynamics

In this sub-section we describe the events provided by the simulation tool to represent the overlay network dynamics, i.e. the login and logout processes related with ultra-peers and leaves. An ultra-peer can enter the system in case of a new ultra-peer's birth or in case of a peer's election to an ultra-peer status. The model adds a new ultra-peer element in the overlay network by performing the creation of the neighbour list for the ultra-peer. The new ultra-peer joins the network by contacting a randomly chosen ultra-peer. This ultra-peer forwards the list of its neighbors to the joining ultra-peer, which contacts immediately the ultra-peers from the list. Every neighbor's list gets an update, derived from the ultra-peer's presence. An ultra-peer logging out causes the assignment of its leaves to a randomly chosen ultra-peer. Also its neighbor list is assigned to another ultra-peer. A leaf logging in provides the assignment of a number of files to share. The peer joins the overlay network by establishing a relationship with an ultra-peer. It is selected randomly with uniform probability from the ones that have less than 150 connected leaves. The number of files shared by such a peer, is determined by following the distribution introduced in [2], which we report here: 25% of the total number of peers share zero files (these peers are called *free-riders*), 70% of peers share up to 100 files and 5% of users share more than 1000 files. A peer logging out causes a decrement in the number of replicas, for all shared files. For this reason the ultra-peer, which was connected to the peer, needs to update its list of leaves. The churning of overlay network is simulated considering the birth of peers and ultra-peers in the system. As in [2], exponential distributions are used to generate the birth and the death of peers and ultra-peers in the system.

D. File Popularity and Product Life Cycle in Marketing Envvironment

File popularity in Gnutella network is measured in [11]. As reported there, file popularity can be defined

as *as the percentage of successfully contacted peers with the file. Given the random distribution of files among peers, the popularity can be interpreted as the probability of having that file at a random peer.* They define *the change in popularity of a given file over interval τ as the difference between its popularity at the beginning and the end of such an interval (in percentage points).* It is clear that this definition involves the fact that popularity can be produced by the system. Considering that file popularity cannot be measured like a physical parameter in the system, the assumption made by Reza et al. is a good way to estimate this parameter in the Gnutella network. Analyzing the measurement results in [11] it is possible to find an interesting similarity between these graphs and the PLC behavior in marketing literature. We could link these apparently disconnetted quantitites taking into consideration that the success of P2P systems has also got economic reasons, as already mentioned before. In [13] a visualization of a product life cycle for an industry is shown. The curve expresses the total market sale that is the *industry product life cycle*. All products have a certain lifetime, during which they pass through certain stages. Their lifes begin with their market introduction. Then they go through a period during which their market grows rapidly. Eventually, they reach market maturity after which their market declines until the end of their life. The exact path traced by the product life cycle varies from case to case. Marketing theory cannot provide a general analytical function for this curve, because of the big differences between different goods. Each product in the market follows its own behavior. Taking account of these properties, and the measurement results reported in [11], we decided to model the popularity of files in the P2P system using the behaviour of product life cycle (PLC) in marketing environment. We are assuming the presence of two different types of files, as suggested by [10]. The files already present in the system are considered as having a constant popularity. They are modeled as old files, belonging to the system for a long time. Every time a peer decides to share a new content, a new file joins the network during the simulation. These files are considered as *new* and they are characterized by a popularity depending on time. The interest they generate in the system is heavily time dependent. As reported in [11] file popularity is assumed to have a life cycle of one year, and it can change with a daily granularity. We assumed that a file could join the network at every instant of its PLC. This means it can appear in the system in the first or in the final stage of its popularity cycle. This temporal instant, describing the position of the file in its PLC at the time of its birth, is defined as T_{pop_0}. Every time a birth of file occurs T_{pop_0} is randomly chosen in a set of 365 integer numbers. These integer numbers represent the number of days in a time period of one year. Another important parameter we consider for every file is its age in the P2P system, which clearly depends on the instant such a file is born in the system, denoted as T_{birth}. If we define T_{sim} as the current simulation time, T_{age} can be expressed like $T_{age} = T_{sim} - T_{birth}$. The analytical function

$$\Psi(\theta) = \begin{cases} (e^{\theta/183} - 1)/(e-1) & \text{if } \theta \in [0, 183) \\ 182/\theta & \text{if } \theta \in [183, 365) \\ 0.5 & \text{if } \theta \in [365, \infty) \end{cases} \tag{1}$$

is a good approximation both for the PLC (shown in [13]), and for the measurements results [11]. They report a growing exponential behavior and, after an average period of 183 days (six months), a smoother decrease (see Fig. 1). It is important to underline that each file follows its own model of popularity behavior. The best way to model popularity should be to consider the different types of files present in a P2P system, measuring the file request arrival rates for each category (video, music, software, documents...) and choosing a correct popularity function. We decided to

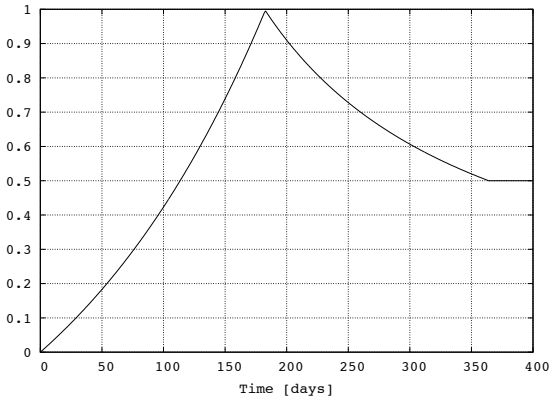


Fig. 1. Popularity function

model the query event with reference to each file, and not to the peers, as it is usually done in P2P simulators [17]. At the beginning of the simulation the first query is scheduled for every file. After the query has been performed, a new query is scheduled, following a cyclic procedure. We chose to model the query process with such a technique in order to manage its behavior with respect to file popularity, as already suggested in [10]. The query interarrival time for every old file is modeled with the exponential distribution. If we define the random variable A_i as the query interarrival time for the old files, its expected value can be expressed like $E[A_i] = 1/\lambda$. The cumulative distribution function is given by $F_{A_i}(x;\lambda) = \text{Exp}(x;\lambda) = 1 - e^{-\lambda x}$. We can define the random variable A'_i as the query interarrival time for the files born during the simulation. We model A'_i with the same exponential distribution, modulated by the popularity function. The cumulative distribution function, for the file request interarrival times is given by

$$F_{A'_i}(x) = \text{Exp}(x; \lambda/[1 - \Psi(\tau)]), \tag{2}$$

the expected value of A'_i is

$$E[A'_i] = \frac{[1 - \Psi(\tau)]}{\lambda}, \tag{3}$$

where Ψ is the popularity function shown in (1) and

$$\tau = T_{pop_0} + T_{age}. \quad (4)$$

So, during the simulation the query inter-arrival time for each new file follows the behavior of the popularity function. τ expresses the popularity time dependency. At the birth of a new file, $\tau = T_{pop_0}$ which describes the position of the file in its PLC at the time of its birth (as reported in II-D). File popularity shifts on time depending on the age of the file, expressed by the term T_{age} in (4). At the maximum of the popularity function, a given file becomes very popular. Thus the inter-arrival time of queries gets minimal to allow a high frequency of requests. We set an average arrival rate λ for old file requests equal to 5 requests/day. The resulting maximum value of the file request arrival rate is 915 requests/day, which fits well with measures, taking into account also the results and the model presented in [10]. Values of popularity after 1 year (over the domain of the popularity function) are considered as constant. The constant value of popularity is set to 0.5 (457.5 requests/day on average), after the end of their PLC the new files will double the popularity of the old files.

During the file query process, the peer processing the query is chosen randomly among the ones joining the overlay network. When the file is queried, two conditions must be satisfied: the file must not be present in the list of shared files, nor in the list of non-shared but owned files, maintained by the peer. If these conditions are satisfied, the query process comes to a good end and the requesting peer can ask its ultra-peer to search the file. During search process the involved ultra-peer receives the requested file ID. It searches the file at one-hop distance, contacting its leaves and the directly connected ultra-peers. The neighbouring ultra-peers receive the file ID and forward it to their leaves. The process terminates when the information is completely received from the peer. The final result is a list of peers at one-hop distance who share the requested file. If no peer is found, the process is repeated also for two-hop distance ultra-peers, which search the file in their attached leaves. When the list of peers sharing the file is received, the best peer must be selected. This choice is based on two important parameters: the amount of bandwidth of the peer and the number of download processes the peer is satisfying. The selected peer is the one having the largest ratio between the available bandwidth and the number of downloads. When the file is received by the peer, the requesting user can choose whether to insert the file in the list of shared files or not to share it. The number of replicas of the file is incremented or not, according to this choice. The decision to share a file is determined with a probability of 50%. This probability is zero in the presence of free-ride peers, which share no files. The peer which decides not to share the file is added to the list of non-shared files. The file is not shared and the peer cannot request it anymore.

III. Simulation Results

Following we present the effects of such a model of popularity on the system. We considered a huge network with 250000 peers, 2500 ultrapeers and 5000 single files. The total number of files at the beginning of the simulation, considering replicas is 103387.

The results we show in this Section are not just limited to popularity. Initially we report the number of queries measured over whole simulated time received by two particular files: the first one ("file 1" in Fig. 2) at the time of its birth stands at the beginning of its PLC ($T_{pop_0} = 0$ days). For this reason the file will reach its peak of popularity in 182 days, after that its popularity will decrease. The second considered file has a T_{pop_0} almost at the end of its PLC; the effect is a linear behavior in the number of received queries during the simulation. In this case the simulation time is 1 year. The same (linear) behavior is followed by files already present in the system at the beginning of the simulation, the rate for these files is described by the exponential distribution.

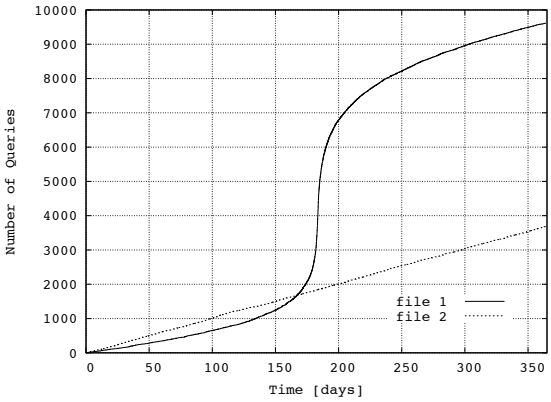
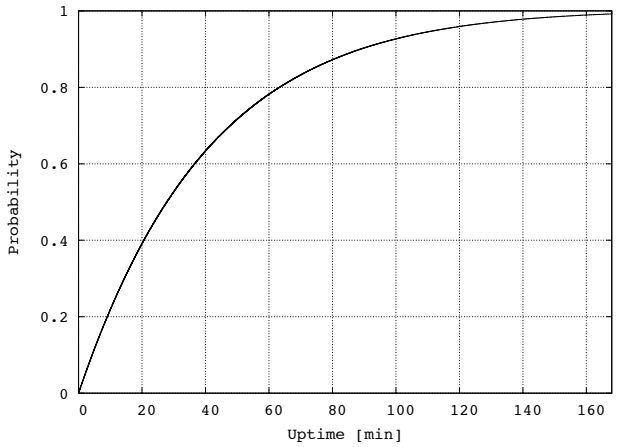


Fig. 2. Number of Queries received by files with different popularities

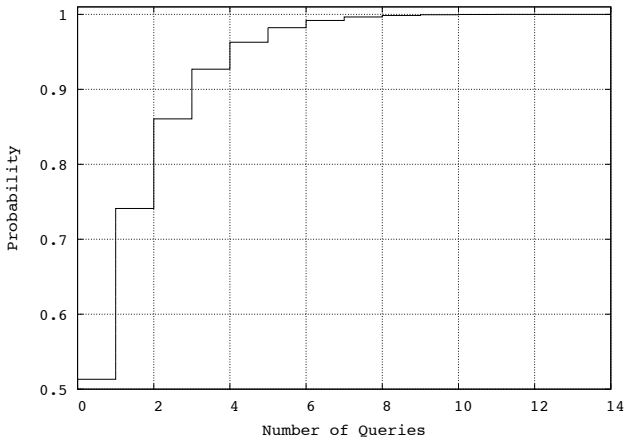
In Figures 3(a) and 3(b) the behavior of queries is considered from the peer's viewpoint. We report the cumulative distribution function for the peer' uptime and for the number of queries done by every peer. The details about the simulation are reported in TABLE I. Using these rates for the events, we can study the system in a realistic way, respecting the high dynamism of a P2P system. In one week of simulation time, the number of peers joining the system is bigger than 10^6. Analyzing the figures, it is possible to see the validity of the model. The number of queries performed by every peer respects the results of measurements published in [15]. The results in Fig. 3(a) fits well with

Query for the Old Files	5 every day
Birth of Files	1 every 40 sec
Birth/Death of Peers	5000 every hour
Birth/Death of Ultrapeers	1 every 150 min
Simulation Time	7 days

TABLE I: Interarrival rates of the events and simulation time.



(a)Uptime of the peer



(b)Number of queries performed by peers over whole simulated time

Fig. 3. Empirical CDF of observed measures

the distribution reported in [14] for the peer' uptime. It is important to underline that the first aim of such a model is to maintain a good level of scalability. We can also claim that these results confirm the measurements done in the Gnutella network. In particular we started from modeling file popularity, and scheduling the requests with a behavior depending on time. It can be validated by looking at the query behavior from the peer' viewpoint.

IV. Time Evolution of Quantiles

In this Section we present the analysis performed using a proprietary software, developed at the Computer Science Department of the University of Canterbury, New Zealand. This software estimates the evolution of quantiles over time with controlled error. We analize the correctness of our model, by considering the time evolution of quantiles of query interarrival times. What we want to verify is the behavior for the file born during the simulation. The software we used can study the time evolution of quantiles, running a number of parallel simulations at the same time. From the final results in Fig. 4(a), 4(b) and 5, we want to verify that the query interarrival time still follows the exponential distribution which we use at the beginning for scheduling processes. Most simulation output analysis is confined to the estimation of mean values. However, the estimation of quantiles provides a deeper insight into the simulated model. Quantile estimation can answer questions like: What is the probability of a query interarrival time longer than t? Questions of this kind are often of more interest than the average system behaviour. A set of several quantiles can be used to approximate a probability distribution function. The analysis of the time evolution of these quantiles as the simulation progresses provides deeper insight into the transient behaviour of the system of interest.

A. Method of Estimation

We use the method of quantile estimation that was proposed in [18] and [19]. This method can show the evolution of a set of quantiles of $F_{A_i'}(x)$ over time, i.e. for increasing i. This enables us to compare the results of our simulator with the expected behaviour given by Equation (2), especially the influence of $\Psi(\tau)$ will be evident. The q-quantile $x_q = F^{-1}_{A_i'}(q)$ can be estimated on the basis of an independent and identically distributed random sample of A_i', which is provided by multiple independent replications of the same simulation run. The half width of a confidence interval of the estimate $\hat{x}_q$ can be described by:

$$\hat{x}_q \in x_{q \pm \epsilon_q}. \tag{5}$$

ϵ_q is an interval in the range of the probability (see [20]). Note, $q \pm \epsilon_q$ should not exceed the bounds 0 and 1.

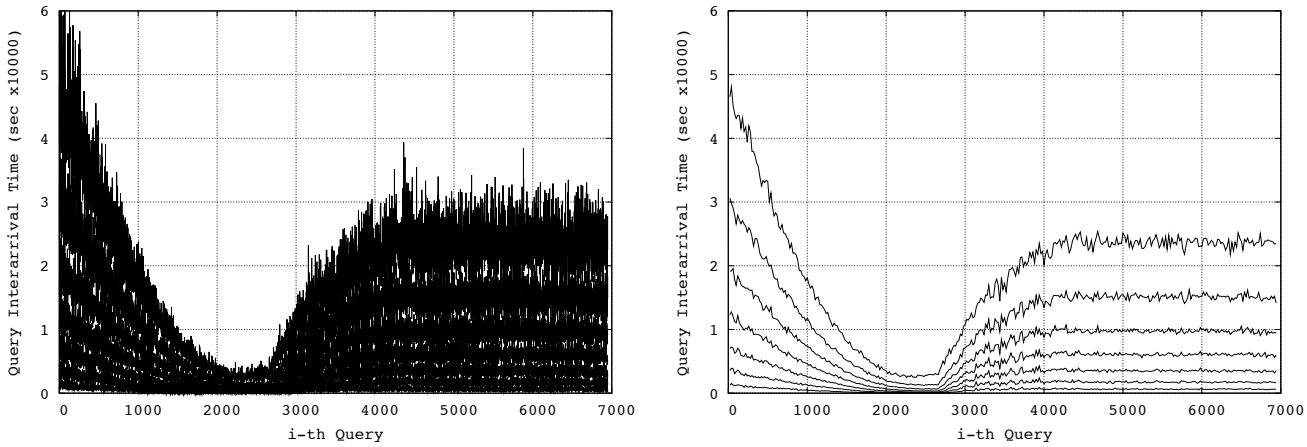
Consecutive quantiles q_k and q_{k+1} are selected automatically so that their confidence intervals do not overlap:

$$\begin{aligned} q_k < 0.5: &\qquad q_k - \epsilon_{q_k} = q_{k+1} + \epsilon_{q_{k+1}}, \\ q_k > 0.5: &\qquad q_k + \epsilon_{q_k} = q_{k+1} - \epsilon_{q_{k+1}}. \end{aligned}$$

This selection starts with the median and searches in both directions, $q_k < 0.5$ and $q_k > 0.5$, for more possible quantiles. The number of selected quantiles depends mainly on the size of the random sample. Non-overlapping confidence intervals avoid high correlation between the estimated quantiles.

B. Query Interarrival Time

The query interarrival times A_i' of the ith query form a sequence of random variables, where $1 \leq i < \infty$. All A_i' are unequally distributed, i.e. $F_{A_i'}(x) \neq F_{A_{i+\Delta}'}(x)$, because they are influenced by $\Psi(\tau)$. To assure a correct behaviour of our simulator we analyse the evolution of $F_{A_i'}(x)$ for increasing i. For small i a long interarrival time is expected, i.e. A_i' should show relatively large values. With increasing i up to the maximum of the file's popularity the interarrival time A_i' should decrease to a low level, implying that a relatively large number of queries take place. After this period A_i' should grow again with increasing values of i because the popularity is shrinking.



(a)Quantile estimates. (b)Smoothed quantile estimates.

Fig. 4. Time evolution of quantiles of $F_{A'_i}(x)$ for a file with $T_{\text{pop}_0} = 0$ days. The query interarrival time depends on file popularity. The end of the PLC is reached in one year of simulation time, roughly around the 4500th query

In Figure 4 we depicted quantiles of $F_{A'_i}(x)$ for increasing i. In this case we considered a small network with 2500 peers, 25 ultrapeers and an initial number of files equal to 50 (594 considering also replicas); the interarrival query time concerns a file with $T_{pop_0} = 0$ *days*. Using $\alpha = 0.1$ and 99 independent replications 7 quantiles could be selected with non-overlapping confidence intervals: 0.067, 0.178, 0.329, 0.5, 0.671, 0.822, 0.933. All results were obtained with a controlled statistical error given by the halfwidth of the confidence interval of the final estimates not greater than 10% in the probability domain at the 0.9 confidence level. Figure 4(a) shows the original quantile estimates, whereas a smoothed curve for every quantile is shown in Figure 4(b). The original estimates are smoothed by averaging 20 consecutive values. As we can see, the behaviour is as expected. The curve of the quantile estimates of A_i start and finish on a relatively high level indicating a low popularity. In between they are on a low level caused by higher popularity. The big variance in the behavior of each single quantile is because interarrival times are independent of each other.

We verifed the exponential distribution of A'_i at the first and the 364th day of the simulation, with $\Psi(1) = 0$ and $\Psi(364) = 0.5$ (see Fig. 5). The smoothed quantiles and their confidence intervals are checked against $F_{A'_i}$. As one can see, the estimates are as expected.

V. Some Comments about Scalability

Finally, we report the computation time and the memory utilization values needed by the simulator. The simulated overlay network is the one we used for the previous results (overlay network with an initial number of 250000 peers, 2500 ultrapeers and 5000 files). The considered simulation time has been fixed to 1 year. We have performed all the simulations using a server equipped with a CPU Dual Intel(R) Xeon(TM) 3.00 GHz and with 4 GBytes of memory. The required computation time is 72 hours, which reveals a good level of scalability also in case of huge overlay networks. The memory utilization is always less than 10%.

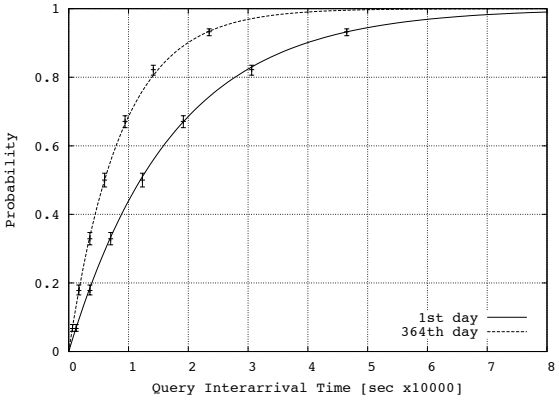


Fig. 5. Interval estimates of quantiles with $\Psi(1)$ and $\Psi(364)$ checked against the expected cumulative distribution function.

VI. Conclusions

We presented a scalable P2P model, particularly focused on file popularity. The work describes the effects that such a quantity generates in the query behavior, assuming that it is strongly linked to marketing procedures. The number of queries performed by every peer and the peer' uptime are considered in order to validate the model, comparing simulation output with the measurement results in Gnutella network. The properties of such a model are studied considering the resulting file query distribution, by running multiple independent replications for every simulation. The model can represent huge overlay networks, up to 500000 peers. Future works will include the use of such a simulative model for studying the impact of P2P traffic on the network, driving our work in the application of this model for network planning reasons.

References

[1] S. Sen, J. Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. ACM Transactions on Networking, Vol. 12, No. 2, pp. 219-232, April 2004.

[2] Raffaele Bolla, Roberto Bruschi, Franco Davoli, Andrea Ranieri, Riccardo Rapuzzi, Michele Sciuto. A Simulative Framework to Estimate P2P Performance Indexes and Traf-

fic Matrixes. In Proceedings of the 2nd European Modeling and Simulation Symposium (EMSS06), Barcelona, Spain, October 2006.
[3] J. Hess, B. Poon. Improving Performance in the Gnutella Protocol. Available online at http://www.cs.berkeley.edu.
[4] M. Karakaya, I. Korpeoglu, . Ulusoy. A General Purpose Simulator for Gnutella and Unstructured P2P Networks. Technical Report BU-CE-0505, Department of Computer Engineering, Bilkent University, 2005.
[5] Q. He, M. Ammar, G. Riley, H. Raj, R. Fujimoto. Mapping Peer Behavior to Packet-level Details: A Framework for Packet-level Simulation of Peer-to-Peer Systems. In Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS 2003), Atlanta, GA, USA, October 2003, pp.71-78.
[6] T. Schlosser, T. E. Condie, S. D. Kamvar. Simulating A File-Sharing P2P Network. In Proceedings of the 1st Workshop on Semantics in P2P and Grid Computing (SemPGRID03), Budapest, Hungary, May 2003.
[7] PeerSim: A P2P Simulator, accessed 01-May-2006. [Online]. Available: http://peersim.sourceforge.net/
[8] PlanetSim: An Overlay Network Simulation Framework, accessed 01-May-2006. [Online]. Available: http://planet.urv.es/planetsim/
[9] NeuroGrid, accessed 01-May-2006. [Online]. Available: http://www.neurogrid.net/php/index.php
[10] T. Hofeld, K. Leibnitz, R. Pries, K. Tutschku, P. Tran-Gia, K. Pawlikowski. Information Diffusion in eDonkey File-sharing Networks. In Proceedings of the 2004 Australian Telecommunication Networks and application Conference (ATNAC 2004), Sydney, Australia, December 2004, pp. 390-397.
[11] Daniel Stutzbach, Shanyu Zhao, Reza Rejaie. Characterizing Files in the Modern Gnutella Network. To appear in the Proceedings of Multimedia Systems Journal, 2007. Technical Report CIS-TR-06-09, University of Oregon, July 2006.
[12] Dongyu Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. In Proceedings of the 2004 Conference on Applications, technologies, architectures, and protocols for computer communications, Portland, Oregon, USA, 2004
[13] E. W. Candiff, R. R. Still. Basic Marketing, Concepts, Decisions and Strategies. Second edition, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1971.
[14] Daniel Stutzbach, Reza Rejaie. Understanding Churn in Peer-to-Peer Networks. Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference, Rio de Janeiro, Brazil, October 2006.
[15] Klemm, A., Lindemann, C., Vernon, M.K., Waldhorst, O.P. Characterizing the query behavior in peer-to-peer file sharing systems. In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, Taormina, Sicily, Italy, October 2004, pp. 55 - 67.
[16] Raffaele Bolla, Riccardo Rapuzzi, Michele Sciuto. Monitoring and Classification of Teletraffic in P2P Environment. In Proceedings of the 2006 Australian Telecommunication Networks and application Conference (ATNAC 2006), Melbourne, Australia, December 2006.
[17] Stephen Naicken, Anirban Basu, Barnaby Livingston, Sethalat Rodhetbhai, Ian Wakeman. Towards Yet Another Peer-to-Peer Simulator. In Proceedings of the Forth International Working Conference in Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs '06), West Yorkshire, U.K., September 2006.
[18] M. Eickhoff, D. McNickle and K. Pawlikowski. Depiction of Transient Performance Measures using Quantile Estimation. Proceedings of the 19th European Conference on Modelling and Simulation, pp. 358-363, 2005
[19] M. Eickhoff, D. McNickle and K. Pawlikowski. Analysis of the Time Evolution of Quantiles in Simulation. International Journal of Simulation, Vol. 7, No. 6, pp. 44-55, 2006
[20] E. J. Chen and W. D. Kelton. Simulation-Based Estimation of Quantiles. Proceedings of the 1999 Winter Simulation Conference, pp. 428-434, 1999

Modelling the effects of a Worm propagation on a BGP router

J.M. Fourneau[1,2], H. Yahiaoui[1]
[1] PRiSM, Université de Versailles Saint-Quentin, CNRS-UMR 8144,
45 Av. des Etats Unis, 78000 Versailles, France
[2] INRIA project MESCAL, Grenoble, France

Abstract—**We model the effects of a Worm propagation in a router queue. Worms like Code RED have used an uniform destination attack on the whole IP space. Such a packet has a very little probability to match the IP addresses in the router cache. Thus the cache miss increases during the worm propagation and real measurements have shown that the BGP routers have experienced availability problem during Code RED attacks. Our model is based on two sub-models of a router and the addresses in the cache. The sub-models are linked by a fixed point equation. We prove an algorithm to compute the solution and we compare numerical results with simulations. We observe that our numerical procedure provides very accurate results.**

Keywords—**Markov model, router, Cache miss, Uniform IP address worm propagation.**

I. Introduction

It is now well known that the Internet inter-domain routing protocol (BGP) suffers from multiple problems of safety: first, reliability because of its reactions to links or routers breakdowns, reliability also when incoherent local policies obstruct the convergence of the routing algorithm or flood the network with obsolete update messages. More recently, BGP routers breakdowns during some worms propagation have also demonstrated a safety problem.

Labovitz and al.report in [3] that the number of exchange messages is several orders of magnitude larger than the number of real faults. These oscillations imply a considerable loss of bandwidth but also larger delays. Of course the TTL, timers and retransmit techniques will insure that packets never stay too long in the network and that they will be sent again. We advocate that this bandwidth lost will have been better used by a more efficient routing protocol and smart routers.

Thus, BGP suffers typically from a reliability problem since an elementary breakdown of router causes serious transient problems. But BGP and the routers also exhibit safety problems which threaten Internet connectivity. The propagation of worms using random IP addresses recently caused important breakdowns of BGP routers [12], [5].

As BGP uses TCP to connect the peers, BGP may suffer the same congestion problem as TCP. When the network is saturated because of usual congestion, worm propagation or distributed denial of service attacks, BGP packets may experience the same problems as data packets. As BGP has its own timer to detect that a link is dead, missing packets due to congestion can create new withdraw message which use the network and increase the congestion. Clearly, we may have some domino effect. Note that the correlation between some worms propagation and BGP instability which has been reported before [5] may also be related to this load and congestion problem. Many papers [12], [5] have reported some correlation in the time series of Code Red propagation and exchange of routing information in BGP routers and even transient disconnection between these routers.

Our main goal is to simulate BGP instable behaviour through the implementation of a realistic BGP instability simulator [16]. We bring realistic BGP behaviour through the use of right delays and realistic comportment of routers during abnormal traffic increase. In order to model the relationship between network load and the stress applied on BGP traffic, we abstract this relationship by modelling the effect of traffic volume increase and traffic shape shift on router's capacity to correctly route it's packets. More precisely, we need a correct estimation of BGP HELLO packets loss during transient traffic increase. This estimation is required to model realistic BGP behaviour during worm attacks, or simply during traffic fluctuations. To do this, we develop a very simple model of a router and its cache for addresses. For the sake of readability we assume IP addresses rather than CIDR address aggregates [2]. The flow of worm has several impact on the routers:

- First, it increases the traffic;
- but due to the limited size of cache in the router and the random IP address generation for propagation, the cache miss ratio in the router increases. This cache miss probability is very high for a packet carrying the worm attack because it is very unlikely that a random IP address is still in the cache. It also increases the cache miss ratio for usual packets as the cache contains addresses added by the worm propagation but that are not typically used by other packets;
- as the cache miss ratio increases, the service time increases and the response time as well. The router may be now congested;
- finally, the routers in the neighborhood do not receive any acknowledgement for their HELLO messages to the congested router and they assume that the router is down. Thus, they change their routes according to the BGP tables and they send update messages to their own neighborhood.

We assume that the router receives two types of packets according to their payload (i.e. worm or data) and we do not distinguish between TCP or UDP packets. In the following, a normal packet has data in its payload while a worm packet contains some malicious code and data to perform the attack. Note that even normal packets may experience cache-misses.

The router has a cache because of the large size of the Internet address space, even if CIDR allows to aggregate addresses into a prefix. We assume that the cache size is

This research is partially supported by SR2I grant from ACI Sécurite Informatique 2005

M. In this cache, the router stores the more recently used addresses and the links packets must use to reach that address according to the BGP routing tables. For the sake of simplicity we assume that the cache replacement is random. We investigate in the last section how we have to modify the model to represent a more realistic replacement strategy such as Last Recently Used.

We model worms which scan uniformly the entire IPv4 space [6], [10]. Such as the Slammer worm , which appeared on January 25th, 2003 and spread throughout the Internet so quickly, that 90% of the vulnerable computers were infected within 10 minutes [6]. We assume that the total IPv4 address space has size K. Thus every address has a probability of $1/K$ to be the destination of a new worm attack. More recent worms use more sophisticated address scanning techniques but this does not have a large impact on our model. Indeed the key idea is that the distribution of the addresses used by the worm and the distribution of the addresses used by the normal TCP or UDP packets are very different.

Many measurements have shown that the distribution of addresses observed in the Internet traffic has a heavy tail [8], [1], [9]. The power law (sometimes denoted as a Zipf-type law) is described such as the number of requests for the i-th most popular item is proportional to $(i)^{-\alpha}$ for $\alpha > 0$. Various studies have reported that several distributions observed in real situations fit well with these distributions. (see, e.g., Breslau and al. [1] and Padmanabhan and Qiu [8]). For instance the authors of [8] reported that the popularity at a busy site shows the Zipf-type law with the value of α larger than 1. Here we assume that two types of distributions for the addresses requested by usual packets : a power load and a slightly modified version of a power low distribution. Note that the theoretical part of the analysis does not depend on the type of distributions used by the worms and by the normal packets. However these distributions have a large influence of the complexity of the algorithm we have developed.

The router has a finite buffer with size B and incoming packets are rejected with the tail drop access mechanism when the router queue is full. We build models to analyze the effect of the worm propagation. In the first model we assume that we know the cache-miss probability and we design a very simple model to analyze the buffer occupancy, the average response time and the loss probabilities. The second model represents the cache and the addresses and provides the cache miss probability which was mandatory in the first model.

The remaining of the paper is as follows. In section II we present a model of the router with the two flows of packets. Section III is devoted to the model of an arbitrary address using an approximate Markov model. In section IV we prove that the fixed point system developed in section III has a positive solution and we give an algorithm to compute this solution. Finally in section VI we study the accuracy of our approximation using simulations and in section VII we show how to extend the model to include a more realistic replacement strategy.

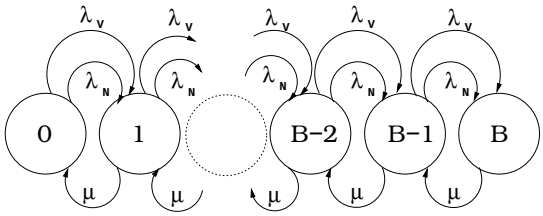


Fig. 1. Model of a router with two types of packets

II. Model of a router

Two types of TCP packets arrive to the router according to the model:

- A flow of usual TCP packets which requires an IP address with the a power law distribution function. Which means, that for an address n the probability of appearance at the router is given by $Y(n) = C \times n^{(-\alpha)}$. We assume that these arrivals follow a Poisson process with rate λ_N.
- A flow of packets which contains a worm. We assume Poisson arrivals with rate λ_V. We also assume independence between these two streams of packets.
- We assume that the services in the router have an average service time of $1/\mu_{CH}$ when the packet experiences a cache-hit and $1/\mu_{CM}$ when we have a cache miss. Let p_N^{miss} be the probability that a typical IP packets provokes a cache miss and let p_V^{miss} be the probability that a packet carrying a worm will provoke a cache miss when it is processed in the router.
- Let us denote by p^{miss} the cache miss probability. Conditioning on the type of packets, we get:

$$p^{miss} = \frac{\lambda_N}{\lambda_N + \lambda_V} p_N^{miss} + \frac{\lambda_V}{\lambda_N + \lambda_V} p_V^{miss}$$

- The services are assumed to be exponential with rate μ such that:

$$\frac{1}{\mu} = p^{miss} \frac{1}{\mu_{CM}} + (1 - p^{miss}) \times \frac{1}{\mu_{CM}}$$

As the worm sends a packet (TCP or UDP) at a destination whose IP address is uniformly selected among the whole IP address space, it is possible to give a simple relation on p_V^{miss}. Assume that the whole IP address spaces has size K and that the cache can contains M IP addresses in its table. The probability that a worm arrivals in a packet provokes a cache miss is simply: $p_V^{miss} = 1 - \frac{M}{K}$

By the law of total probability, the cache miss probability is the sum of the probability of cache miss due to address n times the probability that address n is requested. The probability of cache miss due to address n is the probability that page n is not in the cache when it is requested. Thus:

$$P_N^{miss} = \sum_{n \in K} p(@n\ is\ requested) \times p(@n\ is\ out\ of\ cache)$$

The probability that address n is requested by a normal packet is $Y(n)$. $Y(n)$ is supposed to have a power law distribution function. Let $A(n)$ be the probability that address n is out of the cache, we finally get :

$$P_N^{miss} = \sum_{n \in K} Y(n) \times A(n) \quad (1)$$

We clearly have a M/M/1/B model. All the parameters can be found in the specification of the router (queue size,

cache size, service rate, address space) or by some measurements (arrival rate of normal packets). But the probability of a cache miss is unknown when the router experiences the arrivals of worms. So we must derive a model of an arbitrary address to get the value of $A(n)$ and use equation 1 to get the value of the cache miss probability. We derive in the next section such a model. We use two types of distribution for the addresses requested by the normal packets: a power law distribution and a distribution with groups of addresses with the same probabilities and power low distribution on the groups. The figures we present now are based on a power low distribution for the addresses. Remember that the n-th item (in decreasing order of appearance) has probability$Y(n) = C \times n^{(-\alpha)}$ for a positive value of α. Once this model have been developed, we can analyze the queue and found all the quantities of interest:

- Average response time: to see the effects of the worms on the delays experiences by normal users (see Figure 2).
- Rejection probability due to buffer overflow: This, along with the great increase in messages routing delay, explains why some HELLO messages are not acknowledged (see Figure 3).

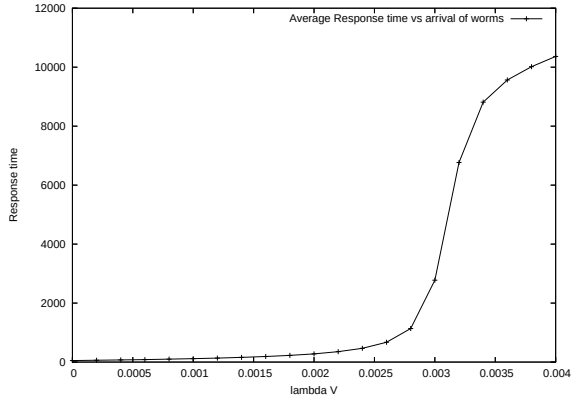


Fig. 2. Average Response Time vs arrival of worms

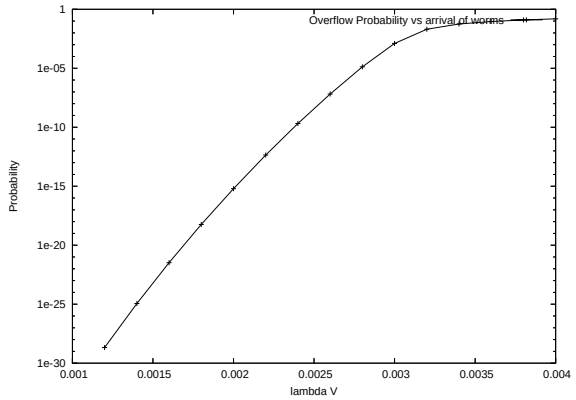


Fig. 3. Overflow Probability vs arrival of worms

The unusual slope of average delay comes from the increasing of the service time when the arrival rate of worms increase. Indeed the cache miss increases and the service time as well.

III. Model of an address in and out of cache

Let n be an arbitrary page index. We want to compute $A(n)$ the probability that address n is in the cache. We

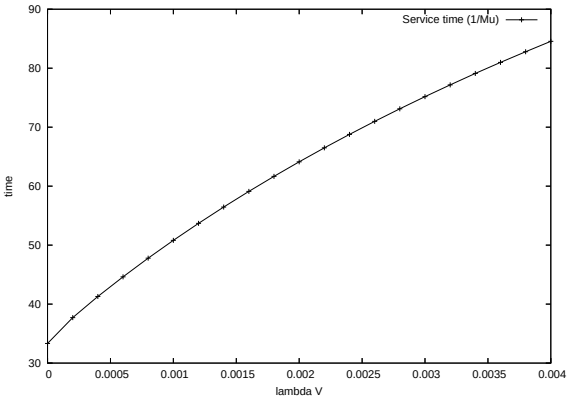


Fig. 4. Service time vs arrival of worms

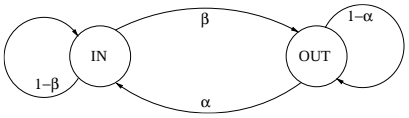


Fig. 5. Markov Chain modelling an address in and out of the cache

model this problem by a two states Discrete-Time Markov Chain. These states are named IN and OUT:

- IN means that address n is in the cache,
- and OUT means that this address is not in the router cache anymore.

Furthermore:

- α_n is the probability of a transition from OUT to IN,
- and β_n is the probability of a transition from IN to OUT.

Of course chain also contains loops when an address does not move in and out of the cache. The probabilities of these loops are obtained from the normalization of the stochastic matrix. The chain is depicted in Figure 5.

α is the probability that address n enters the cache. It is due to a cache miss. Again we use the total probability law and we condition on the type of packets. This address may have been requested by a worm or by a normal packet. If the address is requested by a worm, the cache miss probability is $(K-M)/K^2$. and if the address is requested by a normal packet the probability is $Y(n)$. Finally:

$$\alpha(n) = \frac{\lambda_N}{\lambda_N + \lambda_V} \times Y(n) + \frac{\lambda_V}{\lambda_N + \lambda_V} \times \frac{K-M}{K^2} \qquad (2)$$

$\beta(n)$ represents the probability that an address (say n) leaves the cache. An address leaves the cache because it has been removed by an incoming address requested by a worm or by a normal packet. In both cases we assume a random replacement strategy: any address present in the cache is removed with probability $1/M$. Note that here we just want to give the simplest example to show that we can explain the router performance when the load of worms increases. A random replacement strategy is sufficient for this goal but we will use a LRU model to extend our approach in the future. Remember that the cache miss probability due to a worm is $\frac{K-M}{K}$. Finally we get:

$$\beta(n) = \frac{\lambda_N}{\lambda_N + \lambda_V} \sum_{k \neq n} Y(k)A(k)\frac{1}{M} + \frac{\lambda_V}{\lambda_N + \lambda_V} \frac{K-M}{M\,K} \qquad (3)$$

We want to obtain $A(n)$ which is the probability that address n is out of the cache. If we assume that $\alpha(n)$ and

$\beta(n)$ are known we have a simple Markov chains. Solving the steady state equation gives the probability $A(n)$.

$$A(n) = \frac{\beta(n)}{\alpha(n) + \beta(n)} \tag{4}$$

Rewriting all these equations we obtain the fixed point system we want to analyze:

$$\left\{ \begin{array}{lcl} A(n) & = & \frac{\beta(n)}{\alpha(n)+\beta(n)} \\ \alpha(n) & = & \frac{\lambda_N}{\lambda_N+\lambda_V} \times Y(n) + \frac{\lambda_V}{\lambda_N+\lambda_V} \times \frac{K-M}{K^2} \\ \beta(n) & = & \frac{\lambda_N}{\lambda_N+\lambda_V} \sum_{k \neq n} Y(k)A(k) \frac{1}{M} + \frac{\lambda_V}{\lambda_N+\lambda_V} \frac{K-M}{M\ K} \end{array} \right. \tag{5}$$

In the following we will develop the main theoretical contribution of this paper to prove that this fixed point system has a solution and to provide an algorithm to find efficiently this solution.

IV. About the solution and the algorithm

Before describing the algorithm, we prove for any arbitrary distribution $Y(n)$ that there exists a non zero solution to fixed point system 5.

A. Existence of an admissible solution

We will prove the existence of a solution whose components are all positive. We use the dominated convergence theorem. First we need some technical lemmas to prove that we build an increasing and upper bounded sequence.

Lemma 1: $A(n)$ is an increasing function of $\beta(n)$.
The proof is quite simple when we look at equation III.

Lemma 2: For all distinct address indexes n and k, $\alpha(n)$ is independent of $A(k)$ and $\beta(n)$ is an increasing function of $A(k)$.
Proof : again it is trivially true. Remember that $k \neq n$ and examine equations 2 and 3.

Lemma 3: $\beta(n) \geq \beta_{min} = \frac{\lambda_V}{\lambda_N+\lambda_V} \frac{K-M}{M\ K}$
Proof: Consider the equation used to define $\beta(n)$. The first term is positive by construction. Replace it by 0 to obtain a lower bound.

Lemma 4: $\frac{\beta_{min}}{\alpha(n)+\beta_{min}} \leq A(n) \leq 1$
Proof: Consider equation III and lemma 1. The property is trivially true. Now let us denote in the following S_n this interval and by S the Cartesian product of S_n for all n. Note that Lemma 5 provides a better bound of $A(n)$ to obtain an algorithm in the following of this section.

Theorem 1: Consider the fixed point system of equations 2, 3 and III with unknowns α_n, β_n and $A(n)$. It has a solution such that the unknowns An are in S.

Proof : S is a subset of R^n, it is closed and bounded. Thus S is a compact of R^n and all increasing and bounded sequence of elements of S has a limit in S. The previous lemmas state that if we initialize $A(n)$ with the minimal value given in Lemma 4, then the iterated values of A_n are increasing and bounded. Thus that particular sequence of $A(n)$ have a limit inside S. Note that it does not imply that any sequence of $A(n)$ leads to a solution.

Lemma 5: $\beta_n \leq 1/M$ and $A(n) \leq \frac{1}{M\alpha(n)+1}$.
Proof :

- $\beta_n \leq 1/M$

Indeed $\beta(n)$ is the probability of a cache miss multiplied by the probability that this cache miss provokes the rejection of the address we model. Trivially the first probability is smaller than 1 while the second is equal to $1/M$ due to the random replacement strategy.

- As $A(n)$ is increasing with $\beta(n)$, we substitute the minimal value we have obtained into the definition of $A(n)$ and we get: $A(n) \leq \frac{1/M}{\alpha(n)+1/M}$

B. Algorithm

The exact implementation of the algorithm depends on the number of values for the distribution $Y(n)$. In this section we assume that the distribution has a L-shape but the number of distinct values of Y is much smaller than the number of pages. We suppose that the addresses are gathered into G groups $C_1, \ldots, C_G$. Addresses within the same group have the same probability $Y()$ to be requested by a normal packet. Let Y_{C_i} be this probability. Let n_i be the size of group C_i. We clearly have: $\sum_{i=1}^{G} n_i = K$ and $\sum_{i=1}^{G} n_i Y(C_i) = 1$

These equations state that the number of addresses is still K and that the total probability for the whole set of addresses is one. Using this distribution we may consider a huge number of addresses (K) with a small complexity. Indeed we will show that the complexity of the algorithm is related to the number of groups and not on the size of the groups. It is worthy to remark that the theoretical proof of convergence is not based on this property and the theorem is proved for any distribution Y.

So consider two addresses $n1$ and $n2$ in the same group C_i. They have the same probability to be requested by a worm because of the uniform distribution of the worm scanning on IP space. They also have the same probability to be requested by a normal packet. Because of the Random replacement strategy they also have the same probability to be flushed out of the cache. Thus we have $\alpha(n1) = \alpha(n2)$, $\beta(n1) = \beta(n2)$ and finally $A(n1) = A(n2)$. Thus it is not necessary to solve the fixed point systems for K values. It is sufficient to consider one point in every group. The first two equations of the fixed point system are not changed; we just introduce a new notation $Y(C_i)$ to denote the distribution $Y(n)$ for any n in C_i. Furthermore, assume that n is in group C_i we have:

$$\sum_{k \neq n} Y(k)A(k) = \sum_{k \in C_i and k \neq n} Y(k)A(k) + \sum_{k \notin C_i} Y(k)A(k)$$

As all the states in a group have the same value for Y and A, we can factorize and the summations are replaced by the size of the groups.

$$\sum_{k \in C_i and k \neq n} Y(k)A(k) = (n_i - 1)Y(C_i)A(C_i)$$

And

$$\sum_{k \notin C_i} Y(k)A(k) = \sum_{j \neq i} n_j Y(C_j)A(C_j)$$

Finally,

$$\begin{cases} A(C_i) &= \frac{\beta(C_i)}{\alpha(C_i+\beta(C_i} \\ \alpha(C_i) &= \frac{\lambda_N}{\lambda_N+\lambda_V} Y(C_i) + \frac{\lambda_V}{\lambda_N+\lambda_V} \times \frac{K-M}{K^2} \\ \beta(C_i) &= \frac{\lambda_N}{M(\lambda_N+\lambda_V)}(n_i-1)Y(C_i)A(C_i) \\ &+ \frac{\lambda_N}{M(\lambda_N+\lambda_V)} \sum_{j\neq i} n_j Y(C_j)A(C_j)) \\ &+ \frac{\lambda_V}{\lambda_N+\lambda_V} \frac{K-M}{M\ K} \end{cases} \quad (6)$$

C. Convergence and complexity

Note that the convergence proof given before also applies here. Indeed we have just simplifies distribution Y to have a smaller number of operations. The algorithm is as follows:

Algorithm 1 Computing $A(n)$ by a fixed point iteration:

```
l1 = λ_V/(λ_V + λ_N)
l2 = λ_N/(λ_V + λ_N)
BETAMIN = l1 * (K − M)/(M * K)
for i = 1 to g do
α(i) = l2 * Y(1) + l1 * (K − M)/K
A(i) = BETAMIN/(BETAMIN + α(i))
end for
repeat
for i = 1 to g do
β(i) = (n_i − 1) * l2 * Y(i) * A(i)/M + l1 * (K − M)/(K * M)
for J = 1 to g do
IF j ≠ i β(i) = β(i) + l1 * n_j * Y(j) * A(j)/M
end for
end for
for i = 1 to g do
OA(i) = A(i)
A(i) = β(i)/(β(i) + α(i))
end for
DIFF =0
for i = 1 to g do
DIFF = DIFF + Abs(OA(i)-A(i))
end for
until DIFF < ε
return A()
```

Clearly, we have an unknown number of iterations until convergence. But the inner loop is composed of two deterministic loops the index of which has size g. Therefore the inner loop has a quadratic complexity on g.The number of operations is, then, based on the number of groups but the sizes of the various groups are not important.

We do not have theoretical results on the speed of convergence to the solution but for all experiments we have made the convergence is very fast.

V. Simulation and Validation of the Approach

The model is built to analyze very large system (IP state space is quite large: IPv4 with CIDR aggregation has 2^{16} addresses). But we need to validate the approximations we made in the fixed point iteration model against simulation on moderate size simulations. Thus we have developed a simulator in Omnet++ [15] to check the accuracy of the address model. Thanks to the modularity and the flexibility of OMNet's environment, this simulator, whose primary purpose was to validate our algorithm's computations, allowed experimentation on different effects of worm propagation on routers in general.

The simulator's architecture, shown in Figure 6 is composed from the following components:

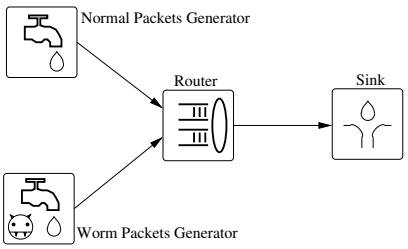


Fig. 6. Simulation model of a router with two types of packets

- The normal packets generator: produces a flow of "normal" packets using a *Poisson* law with rate λ_N.
- The worm packets generator: produces a flow of "worms" packets using a *Poisson* law with rate λ_V.
- The router: is the core component for simulations and performance evaluation. It is an abstraction of real router, so it does not do any routing job. Its role is to simulate routing treatment time, then, forward the concerned packet to the sink. The router receives packets from the generators described above. To ensure the accuracy of simulations, we implemented a real cache digging procedure into the router.
- The sink: is an abstraction for the remaining life of each packet, once out of the router. It is mainly used for statistics computation.

A. Normal packet addresses generation

Trying to copy the nature of flows passing through a router, we used two kinds of generators for normal flows:

1. A power law probabilistic generator: ensured that each address n generated had a probability $C \times n^{-\alpha}$ to come out.
2. A power low per group generator: given G groups of addresses $C_1, \ldots, C_G$ with probabilities $P_1, \ldots, P_G$ and size $n_1, \ldots, n_G$. The probabilities P_i follow a power law distribution while the quantities n_i are exponentially increasing (Figure 7).

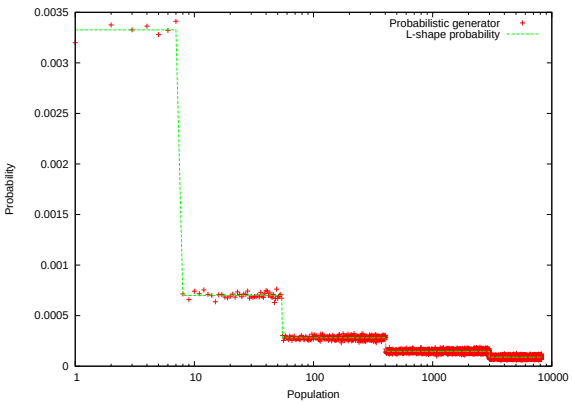


Fig. 7. Power law distributed probabilities on exponentially sized groups: shows a comparison between the L-shape probability distribution of normal addresses arriving at a router, and the probabilities using our generator

B. Validation

First, we model a small system to compare simulation and analytical models. The first validation we can do is computing $\sum_n A(n)$. It must be equal to $K - M$. The numerical results we get satisfies this relation. Now let us consider some simulations, in Figure 8 and 9. We assume that address space contains 8192 addresses and cache size is 128.

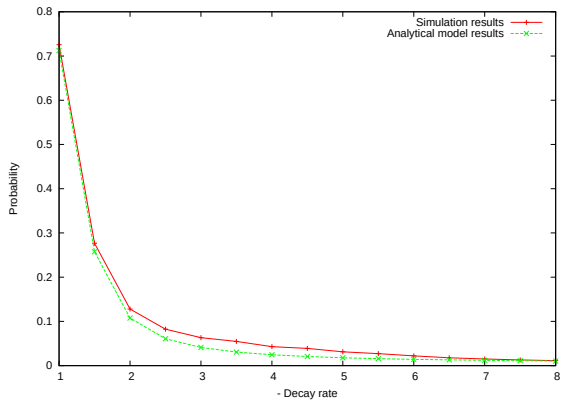


Fig. 8. Cache miss probability for normal packets vs decay rate: relative difference is slight, which confirms the algorithm correctness

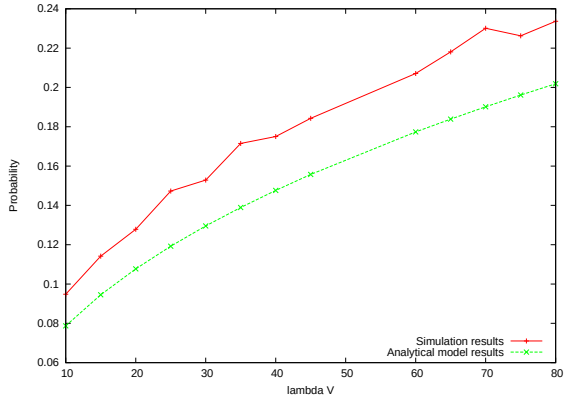


Fig. 9. Cache miss probability for normal packets vs λ_V

C. Numerical Results for Large Address Space

In Figure 10 and 11 we present results that cannot be obtained by simulation in a fast way. We consider a large state space (2^{21} addresses) and we use our numerical algorithms to compute the average queue size and the buffer overflow versus the arrival rate of worms.

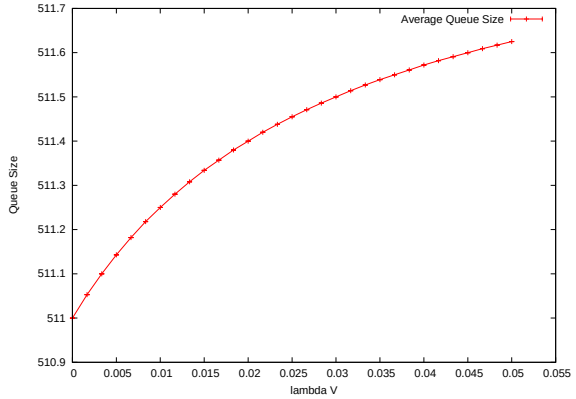


Fig. 10. Effect of λ_V on average queue size: This figure, obtained quickly through our algorithm, has the shape foreseen for this relationship

VI. Conclusion

The model may be extended in several directions. We have made various simplifications which can be removed with the burden of a more complex presentation or more sophisticated numerical techniques.

Modern worms have used more sophisticated scanning of the address space. Code Red II for instance, has a more local attack strategy. This worm has a higher probability of scanning an IP address within the same class B or class A network than an uniform random address [7]. Blaster scans sequentially from a local IP address with a non zero probability [13]. But these strategies do not change the effect on a border router because attacks on the same network do not route through the BGP router. Thus a model of Blaster could simply use a smaller arrival rate of worms on the router.

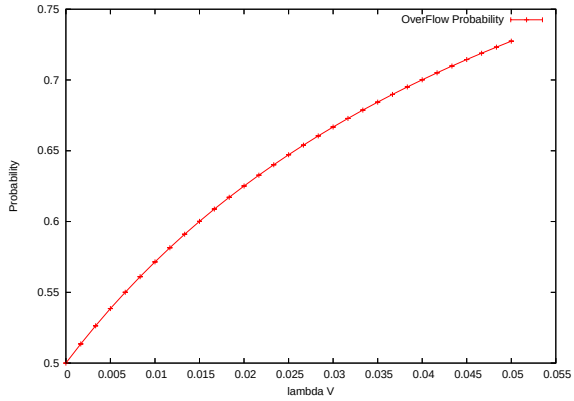


Fig. 11. Effect of λ_V on Buffer Overflow

We have also made an approximation to model the router. We have assumed that the distribution is exponential with a rate which depends on the cache miss probability. We may use a more complex GI distribution. We still use the same model to get the cache miss probability for normal packets and we use numerical techniques to analyze a M/GI/1/B queue.

References

[1] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. "Web caching and Zipf-like distributions: Evidence and implications". IEEE INFOCOM 99, pages 126-134, New York, 1999.

[2] S. Halabi, "Internet Routing Architectures", CiscoPress, 2001.

[3] C. Labovitz, G. R. Malan, and F. Jahanian, "Internet Routing Instability", ACM SIGCOM 97, France, pp 115-126

[4] C. Labovitz, G. R. Malan, and F. Jahanian, "Origins of Internet Routing Instability", IEEE INFOCOM 99, USA

[5] M. Liljenstam, Y. Yuan, B.J. Premore, D. Nicol, "A mixed abstraction level simulation model of large-scale Internet worm infestations", IEEE MASCOTS 2002, USA, pp 109–116.

[6] D.Moore, V.Paxon, S.Savage, C.Shannon, S.Staniford, and N. Weaver. "Inside the slammer worm", IEEE Security and Privacy Magazine, 1(4), July 2003.

[7] D. Moore, C. Shannon, and K. Claffy. "Code-Red: a case study on the spread and victims of an Internet worm". In Proceedings of the second ACM workshop on Internet Measurement, Nov 2002.

[8] V. N. Padmanabhan and L. Qiu. "The content and access dynamics of a busy web site: Findings and implications", ACM SIGCOMM 2000, pages 111-123, Stockholm, Sweden, 2000.

[9] H. Reittu, I. Norros "On the power-law random graph model of massive data networks", Performance Evaluation, V55, pp 3-23, 2004.

[10] S. Staniford, "Code Red analysis pages", http://www.silicondefense.com/cr

[11] K. Trivedi, "Probability and Statistics with Reliability, Queueing and Computer Science Applications (second edition)", Wiley, 2002.

[12] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S.F. Wu, and L. Zhang, "Observation and Analysis of BGP behavior under stress", ACM SIGCOMM USENIX IMW, pp 183-195, 2002

[13] eEye Digital Security, Blaster Worm Analysis, http://www.eeye.com/htmml/Researc/Advisories/AL20030811.html

[14] eEye Digital Security, Code Red Worm, http://www.eeye.com/htmml/Researc/Advisories/AL20010717.html

[15] Omnet++, Discrete event Simulation System, http://www.omnetpp.org/

[16] J-M. Fourneau and H. Yahiaoui "Internet Topology Generation for Large Scale BGP Simulation", In Proceedings of the third International Workshop IPS-MoMe, Mar 2005.

Analytical Modeling of Flit-Based Partial Cut-Through Switching

Dietmar Tutsch, Sebastian Russin, and Daniel Lüdtke
Technische Universität Berlin
D–10587 Berlin, Germany
{dietmart,srussin,dluedtke}@cs.tu-berlin.de

Keywords— **stochastic modeling, interconnection network, cut-through switching, flit, parallel computer, multicore processor, Markov chain**

Abstract—**An important design issue in establishing a parallel computer system or a multicore processor with many cores is a powerful communication network between the processors or cores, respectively. When developing such a network, many parameters, like topology, buffer sizes, switch sizes and so on can be varied. To fix them best for a given system, a performance evaluation of different settings helps. This paper presents an analytical model for performance evaluation of regular dynamic networks that operate with the switching technique of cut-through switching. In contrast to other models, the presented one describes the flow control at the flit level to achieve more accurate results.**

I. Introduction

To establish a parallel computer system or a multicore processor with many cores, an adequate communication network between the processors or cores, respectively, must be provided. Such a network, in case of a multicore processor on a single chip, it is called network-on-chip [1], provides many parameters, like topology, buffer sizes, switch sizes and so on. To fix them best for a given system, a performance evaluation of different settings helps.

This paper presents an analytical model for performance evaluation of parallel computer and multicore processor networks. It is concentrated on modeling the switching technique of cut-through switching because this method is very often proposed and applied. Due to the short model calculation times, other model parameters can exhaustively be investigated. The results are very accurate because message transfer is not only modeled on the packet level. Packets are further on divided into flits (flow control digits), which is much closer to hardware realization.

Only few publications about analytical models for cut-through switching exist. For instance, Szymanski and Fang [2] introduce a model that describes cut-through switching at the packet level. Due to the missing lower levels, the different cut-through switching techniques cannot be distinguished. Entire packets are transferred during a network clock cycle, which is unrealistic.

Anderson and Abraham [3] investigated networks operating in cut-through switching technique. They named the smallest transferred units flits, giving the impression that the networks are modeled at flit level. But a closer look at their paper shows that they divide messages into flits. Usually, messages are divided into packets. Furthermore, it is not distinguished between header flits and payload flits, which behave completely different concerning blocking. Thus, in the context of this paper, flits are actually packets and a packet-based model is presented. Aside from this, only static network topologies are considered.

In contrast to such models at packet level (see also [4], [5]), particularly to develop networks-on-chip (NoCs) for multicore processors, it is important to establish realistic models at the flit level [6]. In the following, an analytical model for flit-based partial cut-through switching is presented. Compared to a packet-based modeling, more accurate performance measures of networks and the message flows within can be obtained by this model. It can be applied to many regular dynamic network topologies, particularly multistage interconnection networks.

The paper is organized as follows. Section II introduces the architectures of networks proposed for parallel computers or for multicore processors. Particularly multistage interconnection networks will deal as an example. The same section also illustrates the modeled switching technique of partial cut-through switching. The analytical model is developed in Section III. It describes the message flow on the flit level. Performance results, which were obtained by applying the model, are presented in Section IV. In Section V, a summary and conclusions are given.

II. Network Architecture and Switching

Various network architectures are known. In this paper, multistage interconnection networks (MIN) will deal as an example.

A. Multistage Interconnection Network

Multistage interconnection networks [7] are dynamic networks which are based on switching elements (SE). SEs are arranged in stages and connected by interstage links. The link structure and amount of SEs characterizes the MIN. MINs with the banyan property, which are exemplarily dealt with in the following, are networks where a unique path exists for any input-output pair. Such MINs of size $N \times N$ consist of $c \times c$ switching elements with $N = c^n$ and $n, c, N \in \mathbb{N}$. The number of stages is given by n. Figure 1 depicts a three-stage MIN. To achieve synchronously operating switches, the network is internally clocked. At each stage k ($k \in \mathbb{N}$ and $0 \leq k \leq n-1$), buffers may be inserted at each switch input or output. The information flow through the network is controlled by the switching technique.

B. Partial Cut-Through Switching

Partial cut-through switching [8] belongs to the packet switching techniques. In packet switching, messages to be transferred are divided into packets. Using packets of limited size allows a buffering of the transferred information units and thus, can avoid information loss and provides a more effective network utilization.

For flow control, packets are further on divided into smaller units called flits (flow control digits). The first flit contains the routing information of the packets header. It is named header flit. The following flits carry the actual information of the packet and are called payload flits. Finally, the

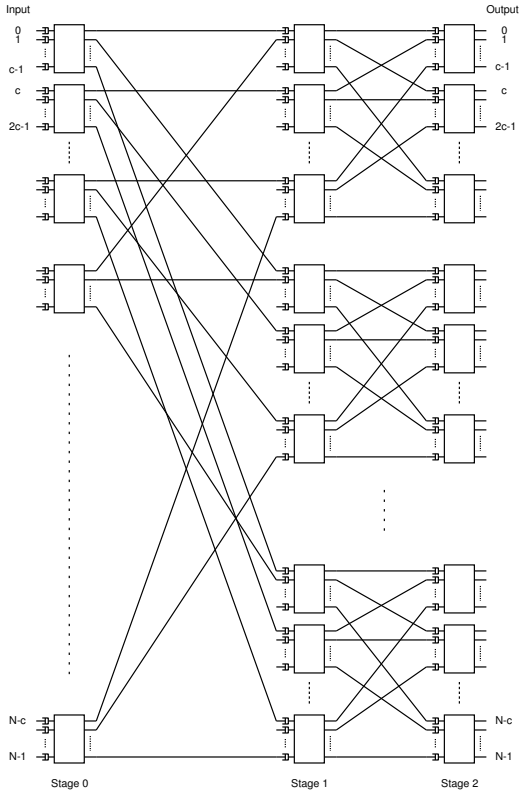


Fig. 1: Three-stage MIN consisting of $c \times c$ SEs

last flit is named tail flit and may contain some additional information like error correction. All flits are of equal size to allow buffer sizes of a multiplier of the flit size and thus, a simple buffer management.

It is assumed that the size of a flit equals the size of a phit (physical unit). A phit represents the size of the information unit that a network is able to forward from a buffer to the succeeding one during a network clock cycle. This means that a packet is transferred by forwarding the header flit to the first buffer on its path through the network in the first clock cycle. In the second clock cycle, the second flit is transferred to this buffer, and so on. In case of partial cut-through switching, the header flit can be forwarded to the second buffer in the same (second) clock cycle, in which the second flit reaches the first buffer. As a result, the flits of a packet move in a pipelined manner through the network.

But if the header flit is blocked somewhere in the network because it looses a contention for a network resource, then it is stored in the buffer at its current position. In the following clock cycles, all remaining flits of this packet are also stored in this buffer. As soon as the blocking of the header flit is released, the flits, starting with the header flit, resume their way through the network.

The backpressure mechanism avoids packet loss: If the buffer of a network stage is completely occupied, the header flit and the related payload flits of a packet destined to this buffer are stored in the buffer of the preceeding stage till the full buffer becomes available again. This is called local backpressure. In global backpressure, a header flit (and afterwards, the related payload flits) are also forwarded to a full buffer, if a header flit leaves this buffer at the same clock cycle. A leaving header flit ensures that clock cycle by clock cycle, buffer space becomes available for the flits of the newly received packet.

III. Markov Model

As in most other attempts in modeling complex systems, establishing a Markov chain of the entire network would result in a state space explosion. Profiting from some network features and traffic assumptions helps to reduce the state space.

A regular (symmetric) network architecture combined with an assumed symmetric (uniform) network traffic leads to a similar behavior of network subsystems. In the following, a multistage interconnection network deals as an example. Other regular networks can be modeled in the same way by choosing appropriate subsystems and by adapting the parameters to it. The most critical issue will be to describe the dependences between the subsystems.

In case of MINs, a representation of the entire network can be replaced by a single switching element row of input-output pairs. However, this subsystem cyclically depends on itself because inputs (mean values of throughput, delay, and queue length) to this row that originate from another row are also represented by this single row.

The uniform traffic precondition is guaranteed by some assumptions, similarly to those in [4] and other publications in this area:

- The traffic load to all inputs of the network is equal.
- Packet destinations are uniformly distributed. This means that every output of the network is with equal probability one of the destinations of a packet.
- Conflicts between packets / header flits for network resources are randomly solved with equal probabilities.
- The switching of all elements is synchronously performed with an internal network clock cycle.
- The flit flow occurs in every stage and in every row of the network in parallel.

Furthermore, some additional assumptions simplify the dependences and also reduce the state space:

- All packets have the same size, which means that they consist of the same number pl of flits.
- Flits are immediately removed from their final destinations after arrival.
- The destinations of succeeding packets are independent of each other.

These assumptions allow establishing a mathematical model to determine the performance of $N \times N$ MINs consisting of $c \times c$ SEs with $n = \log_c N$ stages. At each stage k ($0 \leq k \leq n-1$), there is a FIFO buffer of size bl_k to accommodate bl_k flits. To simplify the following equations, it is assumed that the buffer size is a multiple of the packet size: $bl_k = j \cdot pl$ with $j \in \mathbb{N}$. The global backpressure mechanism is applied.

Four basic types of equations define the model: equations describing the state probabilities, the buffer behavior, the probabilities of receiving flits, and the probabilities of sending flits.

A. State Probabilities

In case of modeling a flit-based switching technique, it is important to distinguish two kinds of flits: flits that belong to the first packet in a buffer (in the following called E-flits) and flits that belong to packets located after this first one (called A-flits). Then, the state (e, a, i) of a buffer is characterized by the number e of E-flits and the number a of A-flits. Furthermore, some information of the first flit in the buffer must be provided: Its position i within the corresponding packet also characterizes the state space. As a result, the buffer state probability of stage k at clock cycle t is denoted by $\pi_{e,a,i}(k,t)$. Table I gives some examples for a buffer of size $bl = 6$ flits and a packet length of $pl = 3$

TABLE I: Some states for a buffer of size $bl = 6$

State Prob.	Buffer Occupancy (6 positions) 1	2	3	4	5	6	Description
$\pi_{0,0,0}$	–	–	–	–	–	–	buffer empty
$\pi_{3,0,0}$	E0	E1	E2	–	–	–	single entire packet
$\pi_{2,0,0}$	E0	E1	–	–	–	–	still missing Flit 2
$\pi_{2,0,1}$	E1	E2	–	–	–	–	Flit 0 already sent
$\pi_{2,1,1}$	E1	E2	A0	–	–	–	ditto, Flit 0 of Packet 2
$\pi_{2,3,1}$	E1	E2	A0	A1	A2	–	ditto, entire 2nd packet

flits. Of course, much more states than those ones presented in the table exist for this buffer.

Note that if the buffer is in the state (2,3,1) (last line of Table I), it can accept the header flit of a new packet because the payload flits E1 and E2 will leave the buffer clock cycle by clock cycle: payload flits will always follow the header flit and will never be blocked. Thus, the payload flits of the new packet will find available buffer space when they arrive. Due to global backpressure, even in state (3,3,0), which means that two entire packets are in the buffer, a new header flit can be accepted, but only if the header flit E0 is sent.

The overall number of states for this model can be determined by considering two scenarios: If only E-flits are in the buffer, the first E-flit's possible positions i in the packet (of length pl) depend on the number e of E-flits: $0 \le i \le pl-e$. The number of such states results in

$$\left|\left\{\pi_{e,0,i} \,|\, 1 \le e \le pl; 0 \le i \le pl - e\right\}\right| = \sum_{e=1}^{pl} (pl - e + 1) = \frac{pl \cdot (pl+1)}{2} \quad (1)$$

If there are additional A-flits in the buffer, $bl - e$ buffer places remain for them. For A-flits in the buffer, a number of

$$\left|\left\{\pi_{e,a,i} \,|\, 1 \le e \le pl; 1 \le a \le bl - e;\ i = pl - e\right\}\right| = \sum_{e=1}^{pl} (bl - e) = pl \cdot bl - \frac{pl \cdot (pl+1)}{2} \quad (2)$$

states is yield. The overall number of states results from Eq. (1) and (2) plus the additional empty state in

$$\left|\left\{\pi_{e,a,i}\right\}\right| = pl \cdot bl + 1 \quad (3)$$

For all these states, the state transition equations must be established. The state transition equations describe, how states $\pi_{e,a,i}(k, t+1)$ can be achieved from previous clock cycle states $\pi_{e,a,i}(k,t)$. Table II gives the events that initiate state transitions. The events correspond to flit actions. The payload flits represent all flits of a packet (including the tail) except the header flit. $p_{send}(k,t)$ denotes the probability that a header flit is sent out of the buffer at stage k at clock cycle t. The probability that a buffer at stage k receives a header flit at clock cycle t is called $p_{receive}(k,t)$. But not all state transitions are valid for all states. For instance, it is obvious that a header flit can only be sent if there is a header flit in the first buffer position. This means that for all states at clock cycle t, it has to be checked whether each of the state transitions of the first column of the table are valid. If so, the state probability has to be multiplied by the corresponding transition probability of the second column. This product contributes to the state probability of the resulting state at clock cycle $t+1$. For instance, to the state probability of an empty buffer at clock cycle $t+1$, it is contributed by a formerly empty buffer that did not receive a new flit and by a buffer that has accommodated a single (tail) flit and also did not receive a new flit:

$$\begin{aligned} \pi_{0,0,0}(k, t+1) &= \pi_{0,0,0}(k,t) \cdot (1 - p_{receive}) \\ &+ \pi_{1,0,pl-1}(k,t) \cdot (1 - p_{receive}) \end{aligned} \quad (4)$$

Note that payload flits (including tail flits) will always be sent (and thus, always be received) because only header flits may suffer from occupied resources.

In most cases, the model will reveal a huge state space, for which the state transition equations can only be established by an automatic generation as presented in [7]. Due to the space limitations for this paper, details how to automatically generate the equations for partial cut-through switching are omitted here. They can be found in [9].

B. Buffer Behavior

The probabilities of sending and receiving a header flit depend on the buffer behavior of the succeeding network stage. Thus, this behavior must be determined first. Behaviors α and $\bar{\alpha}$ are distinguished. Behavior α is defined to describe the situation that the succeeding buffer is ready to accept a new packet (header flit). Correspondingly, behavior $\bar{\alpha}$ is defined to represent the situation that the succeeding buffer is not ready to accept a new packet because it is full or because it is currently receiving payload flits. The corresponding probabilities of these buffer behaviors at stage k at clock cycle t are called $\alpha(k,t)$ and $\bar{\alpha}(k,t)$, respectively, with $\bar{\alpha}(k,t) = 1 - \alpha(k,t)$.

A new packet can only be received by a buffer if the previous packet has entirely been received. That means behavior α is only possible if the previous buffer state belongs to the set

$$A_k = \left\{(0,0,0)\right\} \cup \left\{(e, z \cdot pl, i) \,\middle|\, (e, z \cdot pl, i) \in \Pi_k, z \in \mathbb{N} \cup \{0\}, e + i = pl\right\} \quad (5)$$

where the set Π_k includes all states (e, a, i) that exist for this buffer.

TABLE II: State transitions

Event	Probability
header sending / header reception	$p_{send}(k,t) \cdot p_{receive}(k,t)$
header sending / no header reception	$p_{send}(k,t) \cdot (1 - p_{receive}(k,t))$
header sending / payload reception	$p_{send}(k,t)$
no header sending / header reception	$(1 - p_{send}(k,t)) \cdot p_{receive}(k,t)$
no header sending / no header reception	$(1 - p_{send}(k,t)) \cdot (1 - p_{receive}(k,t))$
no header sending / no header reception, buffer full	$1 - p_{send}(k,t)$
no header sending / payload reception	$1 - p_{send}(k,t)$
payload sending / header reception	$p_{receive}(k,t)$
payload sending / no header reception	$1 - p_{receive}(k,t)$
payload sending / payload reception	1
empty buffer, no sending / header reception	$p_{receive}(k,t)$
empty buffer, no sending / no header reception	$1 - p_{receive}(k,t)$

Furthermore, a new packet can only be received by a buffer if either there is at least space for a single flit available, given by the state set

$$B_k = \{(e,a,i) \,|(e,a,i) \in A_k, \;\; e + a < \; bl_k\} \qquad (6)$$

or if the buffer is full (filled with entire packets) but the header flit at the first buffer position leaves the buffer at the same clock cycle: global backpressure comes into effect. As a result, behavior α occurs with probability

$$\begin{aligned} \alpha\,(k,t) \;\; = \;\; & \sum_{(e,a,i) \in B_k} \pi_{e,a,i}\,(k,t-1) \\ & + \pi_{pl,bl_k - pl,0}\,(k,t-1) \cdot p_{send}\,(k,t) \end{aligned} \qquad (7)$$

Besides the behaviors α and $\bar{\alpha}$, it is necessary to know the probability that, if the E-flits and all A-flits in a buffer represent entire packets (no further payload flit is expected), the buffer is full and cannot accept a new packet because the header in the first position is not sent:

$$p_{bufferfull}\,(k,t) = \frac{\pi_{pl,bl_k - pl,0}\,(k,t-1) \cdot (1 - p_{send}\,(k,t))}{\sum\limits_{(e,a,i) \in A_k} \pi_{e,a,i}\,(k,t-1)} \qquad (8)$$

C. Probabilities for Sending Flits

As previously mentioned, payload flits including tail flits are always sent and their sending probability equals 1 because once a header flit has moved to the succeeding stage, this path is reserved for this packet and all remaining flits are following the header flit. This means that only the probability $p_{send}(k,t)$ of sending a header flit out of the buffer at stage k at clock cycle t has to be determined. For higher clarity in the equations, the parameters k and t will be omitted in the sequel. They will only be set if their value is not clearly indicated.

A header flit is destined to a particular output of the switching element at stage k. To this output, some other header flits may also be directed. If a number *hConflicts* of other header flits also try to reach this output, the conflict is solved by random according to the model assumptions and the header flit under investigation wins with probability

$$p_{Win} = \frac{1}{1 + hConflicts} \qquad (9)$$

Conflicts with payload flits are always lost because they are never blocked.

The probability that exactly *hConflicts* conflicts with other header flits occur and no conflict with a payload flit is called p_{Conf}. This probability can be determined as follows: Header flits are destined to the investigated output (among all c SE outputs) with probability $1/c$ and to any other output with probability $(c-1)/c$. Thus, if a number *hFlits* of header flits are in the first buffer positions, exactly *hConflicts* with other header flits occur with probability

$$\begin{aligned} p_{ConfNoPayl} = & \left(\frac{1}{c}\right)^{hConflicts} \cdot \left(\frac{c-1}{c}\right)^{hFlits - hConflicts} \\ & \cdot \binom{hFlits}{hConflicts} \end{aligned} \qquad (10)$$

where the last term describes the different possibilities which of the header flits are in conflict. None of the *pFlits* payload flits is destined to the investigated output if the first payload flit is transferred to one of the other $c-1$ of all c outputs, the second payload flit to one of the remaining $c-2$ of all $c-1$ remaining outputs, and so on: No payload flit is directed to the investigated output with probability

$$p_{NoPayl} = \frac{c-1}{c} \cdot \frac{c-2}{c-1} \cdots \frac{c - pFlits}{c - pFlits + 1} = \frac{c - pFlits}{c} \qquad (11)$$

Then, the probability p_{Conf} is yield by the product

$$p_{Conf} = p_{ConfNoPayl} \cdot p_{NoPayl} \qquad (12)$$

If no payload flit is destined to the output in question, then the last flit in the following buffer of stage $k+1$ must be a tail flit. This means that the investigated header flit with the given number of conflicts can be sent to this buffer if it wins and if this buffer is not full: $p_{Conf} \cdot p_{Win} \cdot (1 - p_{bufferfull}(k+1,t))$. The sum of these probabilities for all possible number of conflicts results in the combined probability of sending

$$\begin{aligned} P_{CS} \;\; = \;\; & \left(\sum_{hConflicts=0}^{hFlits} p_{Conf} \cdot p_{Win} \right) \\ & \cdot (1 - p_{bufferfull}\,(k+1,t)) \end{aligned} \qquad (13)$$

Equation (13) holds for SE input states, where exactly *hFlits* header flits and *pFlits* payload flits are in the first buffer po-

sitions of the buffers. Now, the probabilities for such an SE input combination must be determined.

In general, the probability of a particular SE input combination at clock cycle $t-1$ is given by

$$p_{InpComb} = \prod_{\forall(e,a,i)\in\Pi_k} \left(\pi_{e,a,i}\,(k,t-1)\right)^{ip_{e,a,i}} \qquad (14)$$

where $ip_{e,a,i}$ gives the number of SE inputs being in state (e,a,i). Because the input combination includes all SE inputs except the investigated one, for which the header flit is to be sent, all $ip_{e,a,i}$ must sum up to $c-1$:

$$\sum_{\forall(e,a,i)\in\Pi_k} ip_{e,a,i} = c-1 \qquad (15)$$

For each input combination, as used in Equation (14), multiple permutations usually exist. For instance, if the SE consists of $c=8$ inputs and three inputs are in state $\pi_{0,0,0}$, two in state $\pi_{1,0,0}$, and two in state $\pi_{1,0,1}$ (and the last one is the investigated one), there exist 7! permutations for the states to be distributed among the $c-1=7$ inputs. But similar states cannot be distinguished. In consequence, the number of permutations that differ can be determined by the multinomial coefficient

$$\binom{7}{3,2,2} = \frac{7!}{3!\cdot 2!\cdot 2!} = 210 \qquad (16)$$

In general, the number of permutations is given by the multinomial coefficient that considers all $ip_{e,a,i}$:

$$I_{perm} = \binom{c-1}{\left\{ip_{e,a,i}\,\middle|\,\forall\left((e,a,i)\in\prod_k \wedge\, ip_{e,a,i}\neq 0\right)\right\}} \qquad (17)$$

To determine the probability $p_{send}(k,t)$ that a particular header flit is sent at the newly beginning clock cycle t, all state combinations of the other SE inputs have to be considered and for each combination, the probability of sending must be calculated. Probability $p_{send}(k,t)$ is yield by

$$p_{send}\,(k,t) = \sum_{\forall combinations} I_{perm}\cdot p_{InpComb}\cdot P_{CS} \qquad (18)$$

This probability for sending a header flit neglects some information and is, thus, more inaccurate than necessary. For instance, [7] shows that a packet that was blocked at the previous clock cycle, moves at the current clock cycle with different probabilities than a non-blocked packet: besides behavior α, a new behavior β can be introduced to consider buffers with previously blocked packets. Incorporating this and other neglected information leads to a more detailed model and much more complex equations. Due to space limitations of this paper, the improvement of above equations is omitted here. The interested reader is referred to [9].

D. Probabilities for Receiving Flits

Besides the probabilities for sending header flits, the probabilities $p_{receive}(k,t)$ for receiving a header flit in stage k at clock cycle t (provided that the buffer is able to accept a new packet, which means buffer behavior α) is needed for the presented model. The probability of receiving a header flit at the first stage $k=0$ of the network defines the network traffic load and is an input parameter to the model. This parameter is called *flitload*, which yields

$$p_{receive}(0,t) = flitload \qquad (19)$$

For all other stages ($1\le k\le n-1$), a header flit can only be received if there is a header flit in the first position of the preceeding buffer at stage $k-1$. The probability $p_H(k-1,t)$ of a header flit being in the first buffer position at stage $k-1$ at clock cycle t is given by

$$p_H\,(k-1,t) \;=\; \sum_{\forall((e,a,0)\in\prod_{k-1}\,\wedge\, e>0)} \pi_{e,a,0}\,(k-1,t) \qquad (20)$$

This header flit will be send with probability $p_{send}(k-1,t)$ to one of the c SE outputs of stage $k-1$. Thus, the investigated buffer at the succeeding stage will receive this header flit with probability $p_H(k-1,t)\cdot p_{send}(k-1,t)/c$. But there are c input buffers at the SE at stage $k-1$, eliminating the term c again. As a result and by considering that it is known that the buffer at stage k shows behavior α (thus, the result must be normalized to $\alpha(k,t)$), probability $p_{receive}(k,t)$ is yield by

$$p_{receive}\,(k,t) = \frac{p_H\,(k-1,t)\cdot p_{send}\,(k-1,t)}{\alpha\,(k,t)} \qquad (21)$$

for $1\le k\le n-1$. This concludes the system of equations to be solved to determine all state probabilities.

E. Solving the Equations

Previous equations represent a homogeneous Markov chain. Due to the high number of equations, an equation solver would fail to determine the steady state. But fixed point iteration can be applied. The iteration variable will be t, and it is started with an initial state describing the empty network:

$$\begin{aligned} \pi_{0,0,0}\,(k,0) &= 1 \quad (0\le k<n) \\ \pi_{e,a,i}\;(k,0) &= 0 \quad \big(0\le k<n \\ &\qquad \wedge\;(e,a,i)\in\textstyle\prod_k\setminus\{(0,0,0)\}\big) \end{aligned} \qquad (22)$$

Some constant values result for $\alpha(n,t)$ and $p_{bufferfull}(n,t)$ from the assumption that flits are immediately removed from their final destinations after arrival at the network outputs:

$$\begin{aligned} \alpha\,(n,t) &= 1 \\ p_{bufferfull}\,(n,t) &= 0 \end{aligned} \qquad (23)$$

With these initial settings, the iteration is processed till a steady state is reached. A short sketch of the algorithm is given in Figure 2.

F. Performance Measures

The performance of a network is usually characterized by its throughput (of header flits), delay time, and buffer utilization. These measures can be determined from the steady state of the presented model as follows: The throughput S_i at a network input is given by the probability that a header

```
get network size (number of stages n, SE size c),
    buffer sizes bl_k, packet length pl flit load flitload;
t := 0;
for k = 0 to n–1 do
        initialize all π_{e,a,i}(k,0);
end for;
repeat
        t := t + 1;
        α(n,t):= 1;
        p_bufferfull(n,t) := 0;
        for k = n–1 to 0 step –1 do
                calculate p_send(k,t);
                calculate α(k,t);
                calculate p_bufferfull(k,t);
        end for;
        for k = 1 to n–1 do
                calculate p_receive(k,t);
        end for;
        for k = 0 to n–1 do
                calculate π_{e,a,i}(k,t);
        end for;
until steady state reached;
calculate and print performance measures;
```

Fig. 2: Algorithm to solve equations

flit is offered to the input and the probability that the buffer of this input is able to accept it:

$$S_i = \mathit{flitload} \cdot \alpha\left(0, t_{steady}\right) \tag{24}$$

In the steady state, S_i is equal to the throughput S_o at a network output. But S_o can also be determined by the probability that a header flit leaves the last network stage:

$$S_o = p_H\left(n-1, t_{steady}\right) \cdot p_{send}\left(n-1, t_{steady}\right) \tag{25}$$

The queue length $\tilde{m}(k)$ at stage k is yield by the number of flits in the buffer for each state:

$$\tilde{m}\left(k\right) = \sum_{\forall (e,a,i) \in \Pi_k} \left(e+a\right) \cdot \pi_{e,a,i}\left(k, t_{steady}\right) \tag{26}$$

The delay $d(k)$ of stage k can be determined by Little's Law: The average number of flits in a stage is the product of the throughput and the delay. The number of flits in a stage is given by $\tilde{m}(k)$. The throughput is now to be related to all flits (not only the header flits). It is called $S_{all}(k)$ and is yield by the probability that a new header flit is received (probability $p_{receive}(k, t_{steady}) \cdot \alpha(k, t_{steady})$) or a payload flit (probability that the last packet in the buffer is not complete). The delay results in

$$d\left(k\right) = \frac{\tilde{m}\left(k\right)}{S_{all}(k)} \tag{27}$$

with

$$\begin{aligned} S_{all}(k) &= p_{receive}\left(k, t_{steady}\right) \cdot \alpha\left(k, t_{steady}\right) \\ &+ \sum_{\forall (e,a,i) \in \Pi_k \setminus A_k} \pi_{e,a,i}\left(k, t_{steady}\right) \end{aligned} \tag{28}$$

The delay time of the entire network is then yield by

$$d_{total} = \sum_{k=0}^{n-1} d\left(k\right) \tag{29}$$

IV. Results

The previous section presented a Markov model of flit-based partial cut-through switching in a multistage interconnection network. A more accurate model can be found by an improved modeling of blocked header flits as shortly sketched in the model description. It is presented in detail in [9]. There, a comparison of the analytical model to a simulation can also be found for model validation. The simulation results are obtained by using the simulator *CINSim* [10]. Networks with various different parameters are simulated and compared to analysis. Figure 3 shows a comparison exemplarily. An 8×8 MIN with 2×2 SEs is chosen.

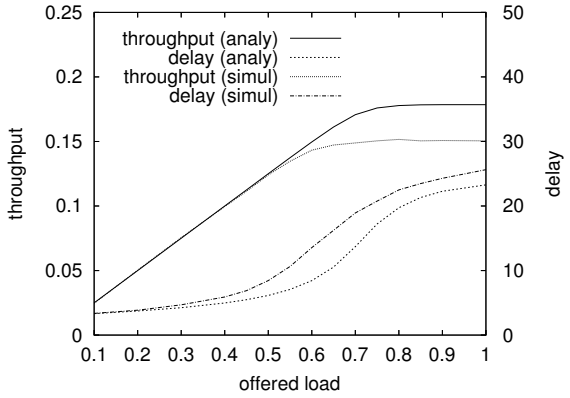


Fig. 3: 8×8 MIN with $pl = 4$ and $bl_k = 8$

Buffers to store $bl_k = 8$ flits at each SE input and at each stage k are used. The packets consist of $pl = 4$ flits. They are fed into the network with their destinations uniformly distributed among the network outputs. Throughput and delay are given depending on the offered load. The throughput is related to packets, which means that due to a packet size of 4 flits, a packet can leave the network at most every fourth clock cycle: The highest theoretical packet throughput is 0.25. The offered load gives the average probability with which a flit is offered to the network. It relates to all types of flits.

Concerning the throughput, the maximum occurring result difference between the two modeling methods is found for a saturated network (offered load greater than 0.7). The difference is about 15%, which is an in literature often found error size for throughput in analytical models of this modeling area. As well as in other analytical models found in literature, the delay in our analysis is much more sensitive to neglected dependences compared to throughput. Delay times of the analysis may differ to simulation up to 30%. Particularly traffic close to network saturation is critical (see Figure 3 at an offered load around 0.6).

To show the modeling power of the analysis, Figures 4 and 5 depict the network performance obtained by investigating some of the network parameters. In Figure 4, net-

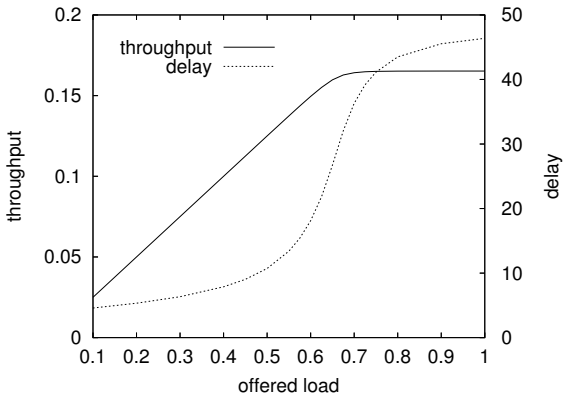


Fig. 4: 81×81 MIN with $pl = 4$ and $bl_k = 12$

work throughput and delay times are presented depending on the offered load to the network. The network is of size 81×81 consisting of 3×3 SEs with a buffer size to store $bl_k = 12$ flits. Packets are divided into $pl = 4$ flits. The figure shows that at an offered load of approximately 0.6, the network becomes saturated and throughput stops growing and delay grows heavily. Thus, the given parameters for the investigated network are only a good solution if the offered load keeps below 0.6.

Figure 5 compares two packet lengths, $pl = 2$ flits and $pl = 4$ flits, dependent on the buffer size, which is increased from $bl_k = 2$ to $bl_k = 12$ and equal at each stage k. The

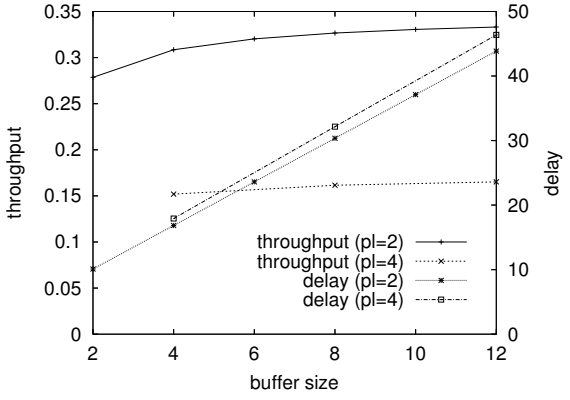


Fig. 5: 81×81 MIN with different buffer sizes and packet lengths

offered load is set to 1, which means that new packets are offered to the network as soon as a network input is available. All other parameters are not changed compared to the previous investigation. Not surprisingly, it can be seen that doubling the number of packet flits halves the throughput. The delay times are of similar size because the number of buffer places to be passed by a packet on its path through the network does not change.

Concerning the buffer size, larger buffers only increase the throughput slightly. Delay time grows in a linear shape due to the higher number of buffers to be passed. Thus, choosing a buffer with double the size of a packet seems to be sufficient. Of course, this will change if bursty traffic is involved.

V. Conclusion

The paper presented an analytical model of partial cut-through switching on the flit level for regular dynamic networks, which are used in parallel computer systems and in multicore processors. Such a flit-based model is much more realistic and accurate than a packet-based modeling as found in most previous publications.

Because analytical models usually extremely outperform simulations in run time, a higher amount and, thus, more deep investigations of network architectures can be achieved with the new model. Architectures can be dimensioned faster and better adapted to a given network traffic. On the other hand, one should be aware that a higher accuracy of results is achieved by simulation.

The given equations analytically model partial cut-through switching for buffer sizes that are a multiplier of the packet length. In this case, partial cut-through switching operates identically to wormhole switching. But wormhole switching was invented to deal with buffer sizes less than a packet length. Our future work is dedicated to extend our model in this direction.

References

[1] Benini, L.; De Micheli, G. 2002, "Networks on Chips: A New SoC Paradigm." IEEE Computer, 35, no. 1: 70–80.

[2] Szymanski, T.; Fang, C. 1990, "Design and Analysis of Buffered Crossbars and Banyans with Cut-through Switching." In *Proceedings of the 1990 conference on Supercomputing*, New York, 264–273.

[3] Anderson, J. R.; Abraham, S. 2000, "Performance-Based Constraints for Multidimensional Networks." IEEE Transactions on Parallel and Distributed Systems, 11, no. 1: 21–35.

[4] Zhou, B.; Atiquzzaman, M. 2002, "A Performance Comparison of Four Buffering Schemes for Multistage Interconnection Networks." International Journal of Parallel and Distributed Systems and Networks, 5, no. 1: 17–25.

[5] Yang, Y.; Wang, J. 2004, "A Class of Multistage Conference Switching Networks for Group Communication." IEEE Transactions on Parallel and Distributed Systems, 15, no. 3: 228–243.

[6] Ogras, U. Y.; Hu, J.; Marculescu, R. 2005, "Key Research Problems in NoC Design: A Holistic Perspective." In *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, IEEE, 69–74.

[7] Tutsch, D. 2006, *Performance Analysis of Network Architectures*. Springer Verlag, Berlin, 1st edn.

[8] Liszka, K. J.; Antonio, J. K.; Siegel, H. J. 1997, "Is an Alligator better than an Armadillo? [Interconnection networks]." IEEE Concurrency, 5, no. 4: 18–28.

[9] Russin, S. 2006, "Mathematische Modellierung von Virtual-Cut-Through-Switching in mehrstufigen Verbindungsnetzen bei Mehrfachsendung (in German)." Master's thesis, Technische Universität Berlin.

[10] Tutsch, D.; Lüdtke, D.; Walter, A.; Kühm, M. 2005, "CINSim – A Component-Based Interconnection Network Simulator for Modeling Dynamic Reconfiguration." In *Proceedings of the 12th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA 2005); Riga*, IEEE/SCS, 132–137.

SESSION 7

Advanced Markovian Models

A quantitative decision model for bottleneck link upgrades in packet switched networks

Martin Qvist
Department of Mathematical Sciences
Aalborg University
9220 Aalborg, Denmark
E-mail: qvist@math.aau.dk

Hans-Peter Schwefel
Center for TeleInfrastructure
Aalborg University
9220 Aalborg, Denmark
E-mail: hps@kom.aau.dk

Martin Bøgsted Hansen
Department of Mathematical Sciences
Aalborg University
9220 Aalborg, Denmark
E-mail: mbh@math.aau.dk

Abstract— **Core network links are frequently based on leased lines, whose capacity can be upgraded or downgraded on time scales of a few days. In this paper, a method for deciding when to upgrade links in an IP network is presented. The method calculates a threshold in the utilization of the link, and an upgrade should be performed when this threshold is crossed. The threshold is calculated using a Markov decision process which includes the traffic model and the costs and rewards involved in running the link. The threshold for Poisson traffic and bursty ON/OFF traffic is compared, and we quantify to what extent having bursty traffic lowers the upgrade threshold. It is also shown that the specific forecasting model for the long-term traffic fluctuations only has a marginal impact on the upgrade thresholds.**

I. Introduction

With converging networks, ensuring adequate link bandwidths in the core network is one prerequisite for achieving proper Quality of Service (QoS). This is especially important when the network usage includes delay sensitive applications such as VoIP. Core network link capacities can typically be upgraded (and downgraded) on medium time scales (ranging from few hours to several days/weeks) by adding additional leased lines, however, with associated costs. Therefore, the goal is to: (1) adapt link capacities only very rarely to reduce reconfiguration effort and costs; (2) work with link capacities that are as small as possible to reduce the leasing costs, while at the same time (3) avoiding violations of service-level agreements caused by congestion and resulting large delays/loss. Since there is a clear trade-off in meeting these different goals, in practice frequently heuristic approaches are applied, e.g. "upgrade when 60% utilization is exceeded during the daily busy hours". The choice of such a utilization threshold can be made either by past experience (trial-and-error) or substantiated by a quantitative model, as developed in this paper. The formalization of the latter requires to determine a cost function which should be optimized by the upgrade (and downgrade) decision strategy.

Short-term traffic fluctuations on time scales of a few seconds further complicate such upgrading decisions. Bursty traffic, as e.g. ON/OFF traffic, (Heffes, 1980), typically leads to higher bandwidth requirements to achieve the same QoS level. These bandwidth demands can even be further aggravated for certain types of long-range dependent traffic, as e.g. shown for aggregated ON/OFF traffic with long-range dependent properties in (Schwefel and Lipsky, 2001). A quantitative decision model for link-capacity upgrades therefore has to allow for different input traffic models, including in particular bursty traffic, as e.g. resulting from ON/OFF type of models. These traffic models have direct impact on QoS violations, which must be penalized in the cost function. Another element in the cost function are the costs for operating the link, which include the equipment costs, the cost of leasing the link, and also the costs due to the upgrade and downgrade procedures. Revenue on the other hand is created by successfully serving traffic within the QoS bounds as stated by a service-level agreement.

A model which supports optimizations of the above outlined type is the Markov decision process (MDP). A MDP is a Markov process whose transition probabilities can be influenced by a set of actions, and to each transition there is an associated reward or cost depending on this action. The key is now in each state to select the action which gives the highest reward, and algorithms for achieving this, on the average, exist, see e.g. (Puterman, 1994) for an introduction. MDPs have been applied previously to bandwidth allocation problems, however, these models are based on monitoring the queue length and to select at each queue length the service rate with highest reward. Such models stem back to (Heyman, 1968; Sobel, 1974; Lippman, 1973) and were extended to networks of queues in (Weber and Jr., 1987). More recent methods of service rate control apply fuzzy logic, (Phillis and Zhang, 1999). The latter reference also contains a good overview of the service rate control problem.

However, making service rate decisions based on queue length is not practicable, since it would result

in bandwidth changes on time scales of queue length fluctuations, so typically within fractions of a second. Furthermore, the above service rate control models assume a Poisson arrival process, hence require extension to allow for bursty traffic.

This paper develops a quantitative model to determine a utilization threshold for upgrade decisions based on MDPs and the aforementioned elements in the cost function. Such a threshold based approach is related to the approach in (Menth, 2001), and other methods include (Hoey et al., 2001) and (Lim et al., 2005), which do not consider costs or ON/OFF type traffic, and (Jagannathan et al., 2003) which considers costs, but leaves the thresholds to be determined by the user. Our model includes the traffic model via queue lengths so as to penalize poor QoS, but the calculated threshold in the utilization of the link does not depend on the queue length. The achieved QoS level is thereby obtained by a discrete-time version of an M/M/1 or MMPP/M/1 queue. Special instances of the latter can be used to model bursty ON/OFF traffic, and, using the methodology of (Schwefel and Lipsky, 2001) and (Greiner et al., 1999) also ON/OFF traffic with long-range dependent properties.

The MDP approach and its mapping to utilization thresholds is introduced in Sections II and III. Subsequently, the MDP model is applied in Section IV to investigate the impact of different parameters of the model on the threshold. Furthermore, the impact of bursty traffic as described by discrete versions of exponential ON/OFF traffic as well as by ON/OFF traffic with (truncated) power-tail distributions for the ON duration (and thus showing long-range dependent properties) is analyzed. Finally, the numerical analysis confirms that the utilization threshold with adequate quantitative choice of the threshold as achieved by the model in this paper, is rather robust towards certain model variations.

II. Derivation of utilization thresholds using MDPs

Instead of basing upgrade decisions on the queue length, it is more suitable to base them on a time averaged property of the traffic, e.g. utilization during busy hours. The utilization of a link is available in most network diagnostics tools and we will not consider the estimation of it further. The long-term change in traffic intensity, or equivalently in utilization, is what we want to match when adjusting the bandwidth of the link. Observing the past and forecasting the future evolution of the utilization during the daily busy hours, the upgrade and downgrade decisions should aim at keeping the resulting utilization between certain thresholds. Although the model can be used to calculate the lower threshold, i.e. for downgrades, we will focus on the upper threshold.

To determine the threshold, the state space of the service rate control MDP is expanded by a dimension $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ which models the long-term evolution of the arrival probability. The transitions between the

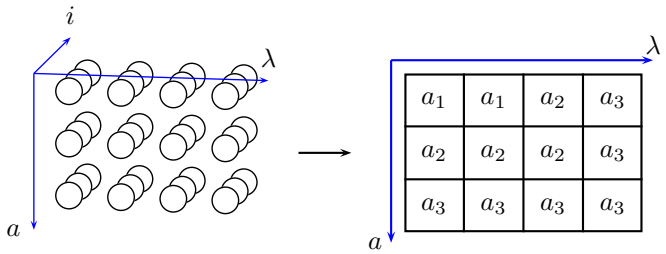


Fig. 1. To the left the state space of the model is shown without transitions. The i, a, and λ axes describe the queue length, bandwidth, and arrival probability, respectively. The model is constrained such that the policy (right) does not include the queue length. The policy consists of a_i denoting which bandwidth to choose in each setting, and this is then searched for the lowest λ such that the policy selects a higher bandwidth than the current bandwidth. The ratio between this λ and a the current bandwidth is the utilization threshold.

λ_i's are assumed in Section IV to have birth-death-structure, although also more general models can be used. From this model a policy for the choice of the bandwidth can be generated. However, in our setting such a policy must not depend on the queue length.

To solve this we employ *constraining* on the policy. The state space is partitioned into groups of states, and instead of finding optimal actions for the individual states, optimal actions are found for each group of states. That is, a constrained policy is required to select the same action for two states belonging to the same group. This leads to a MDP with partial information, cf. (Hordijk and Loeve, 1994).

Applying this to our problem the states are grouped so that only the current bandwidth a and the arrival probability λ are input to the policy, see Figure 1. This gives a policy, which for each bandwidth a and arrival probability λ outputs an optimal bandwidth $d(a, \lambda)$. For each a we can now search for the lowest λ such that $d(\lambda, a) > a$, i.e. the lowest arrival probability at which a higher bandwidth is chosen. This corresponds to the arrival probability where an upgrade should be performed. Normalizing by a we obtain the utilization threshold

$$\rho_a = \frac{\min\{\lambda \,|\, d(\lambda, a) > a\}}{a}.$$

III. Detailed structure of the MDP

An MDP consists of a state space S, an action set A, a matrix of transition probabilities $P_{ij}(a)$ dependent on $a \in A$, and a reward function $r\colon S \times A \to \mathbb{R}$. A policy is a function $d\colon S \to A$ telling which action, i.e. in our setting which bandwidth, to take in a given state. The MDPs considered in this paper are discrete time and as optimization goal infinite horizon discounted costs are used: costs are considered infinitely far into the future, but they are subject to a discount rate $0 < \phi < 1$ which is multiplied per time period, meaning that a cost C experienced n periods in the future is evaluated as $\phi^n C$.

Using discrete MDPs, we consider the Geo/Geo/1 and MMBP/Geo/1 (Markov modulated Bernoulli process) queues, cf. (Tsuchiya and Takahashi, 1993), respectively. The Geo/Geo/1 queue has Bernoulli dis-

tributed arrivals, i.e. for each time period there is either one or no arrivals, and similarly for services. The MMBP is a modulated Bernoulli process, where the modulating states influence the arrival probability, allowing the description of traffic fluctuations on short time scales as e.g. caused by bursty traffic. An MMBP is described by a transition probability matrix Q and an arrival probability matrix L. The latter contains the arrival probabilities of each state in its diagonal.

The link model assumes finite buffer size k, which also results in a finite state-space. To account for losses a penalty cost of having a full queue is added in the reward function, see Section III-C.

A. Geo/Geo/1/k model

The standard service rate control model is extended by the state space $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ of possible arrival probabilities. For simplicity, we assume here that the change of arrival probability shows a birth-death-structure, with birth probability p and death probability q, but more general models can easily be plugged in.

Letting P' denote the transition matrix of the birth-death-process we have for $\lambda_{j-1}, \lambda_j, \lambda_{j+1} \in \Lambda$ that

$$\begin{aligned} P'_{\lambda_j,\lambda_{j+1}} &= p(1-q) \\ P'_{\lambda_j,\lambda_{j-1}} &= q(1-p) \\ P'_{\lambda_j,\lambda_j} &= 1 - p(1-q) - q(1-p), \end{aligned}$$

with the usual adaptations at the boundary of the state space. Letting i denote the queue length, and $a \in [0,1]$ the action (selected bandwidth), the transition matrix for the Geo/Geo/1/k model is thus

$$\begin{aligned} P_{(\lambda',i,a')(\lambda,i+1,a)}(a) &= \lambda'(1-a)P'_{\lambda',\lambda} \\ P_{(\lambda',i,a')(\lambda,i-1,a)}(a) &= a(1-\lambda')P'_{\lambda',\lambda} \\ P_{(\lambda',i,a')(\lambda,i,a)}(a) &= (1-\lambda'(1-a) - a(1-\lambda'))P'_{\lambda',\lambda}, \end{aligned}$$

where $\lambda, \lambda' \in \Lambda$. Again, the usual adaptations of these equations apply for the boundaries of the state space at queue lengths $i = 0$ and $i = k$, where k is the buffer size.

The constraint on the policy is for each arrival probability λ and action a

$$d((\lambda, i', a)) = d((\lambda, i, a)),$$

for all queue lengths i' and i. Hence, the policy only depends on the arrival probability and the currently selected service probability, both of which will only change on longer time scales. Short time traffic fluctuations (burstiness) is included in the model in the next section.

B. MMBP/Geo/1/k model

The MMBP/Geo/1/k model is modified in order to have the same average arrival probability as the Geo/Geo/1/k model. In the used example, the transition probabilities of the modulating process are unchanged and only the arrival probabilities L^λ are modified to achieve a certain average probability λ. Keeping P' as in the previous section we have:

$$\begin{aligned} P_{(m',\lambda',i,a')(m,\lambda,i+1,a)}(a) &= L^{\lambda'}_{m'm'}(1-a)Q_{m'm}P'_{\lambda',\lambda} \\ P_{(m',\lambda',i,a')(m,\lambda,i-1,a)}(a) &= a(1-L^{\lambda'}_{m'm'})Q_{m'm}P'_{\lambda',\lambda} \\ P_{(m',\lambda',i,a')(m,\lambda,i,a)}(a) & \\ = (1 - L^{\lambda'}_{m'm'}(1-a) - a(1-L^{\lambda'}_{m'm'}))&Q_{m'm}P'_{\lambda',\lambda}, \end{aligned}$$

The arrival probability is modulated in small time scales by the state m', and in large time scales by the different average arrival probabilities of the MMBPs. As with the Geo/Geo/1/k the probabilities are multiplied by the transition probability of the modulating states $Q_{m'm}$ and the transition probability $P'_{\lambda',\lambda}$ of the states λ controlling the arrival probability.

The model is constrained so that the queue length and the modulating states M of the MMBP are eliminated in the policy, i.e. for each $\lambda \in \Lambda$ and $a \in A$

$$d(m, \lambda, i, a) = d(m', \lambda, i', a)$$

for every $m, m' \in M$ and queue lengths $i, i' \in I$. This results in a policy depending only on λ and a, i.e. neither the current queue length, nor the state of the modulating process of the MMBP can influence the link capacity selection.

C. Reward function

The method of how to determine an upgrade threshold quantitatively can in principle be applied using any reward function, but we will use a special class of reward functions for the parametric studies in this paper. This class of reward functions contains many of the elements which typically are expected to contribute to a cost or reward criterion: There is a reward r when a packet is served ($i = i' - 1$), and a penalty for poor QoS, here modeled by holding costs that are proportional to queue-length, $h \times i$. This means that if there are many packets in the queue the costs per time unit will be large. Furthermore, an additional penalty g is added when the queue is full (i.e. when $i' = i = k$). This corresponds to the cost of buffer overrun and subsequent packet loss. The costs for deploying different link capacities are expressed by running costs b_a for action (bandwidth selection) a, and an additional cost c occurs, whenever an upgrade (or downgrade) is executed. Altogether, the exemplary reward function used in this paper is:

$$\begin{aligned} & r((m', \lambda', i', a'), (m, \lambda, i, a), a) \\ &= \begin{cases} -hi - b_a - c\bar{\delta}(a, a') & \text{if } i = i' \text{ or } i = i' + 1 \\ -hi - b_a - c\bar{\delta}(a, a') - g & \text{if } i = i' = k \\ r - hi - b_a - c\bar{\delta}(a, a') & \text{if } i = i' - 1 \end{cases}, \end{aligned}$$

where $\bar{\delta}(a, a') = 0$ if $a' = a$ and $\bar{\delta}(a, a') = 1$ otherwise. Notice that the only positive element is the reward r for successfully servicing a packet, and that the function is independent of m, m', λ, and λ'.

D. Finding an optimal policy

Having completed the specification of the MDPs for calculating the utilization threshold, the optimal policies need to be found. If r is the reward function, P

is the transition matrix, $0 < \phi < 1$ is the discounting rate, and S is the state space then a policy (without constraints) can be found by solving the following equation for v, cf. (Puterman, 1994),

$$v(s) = \max_{a \in A} \left\{ r(s,a) + \sum_{j \in S} \phi P_{sj}(a) v(j) \right\}, \quad s \in S, \tag{1}$$

e.g. by iteration, where $r(s,a) = \sum_{j \in S} r(s,j,a) P_{sj}(a)$, see Puterman (1994). With this v the policy can be found by simply replacing $\max_{a \in A}$ with $\arg\max_{a \in A}$ in the right hand side of (1). Algorithms for finding an optimal policy in the presence of constraints are presented in e.g. (Hordijk and Loeve, 1994; Smith, 1971; Schneeberger, 1992). Except for the algorithm in (Hordijk and Loeve, 1994), which does not guarantee optimality, in the worst case these algorithms have to calculate the value of all possible policies. For large state spaces this makes the calculation of the threshold infeasible.

Faster computations of the optimal policy result for the following case: Assume the model includes only two bandwidths, i.e. the initial bandwidth and a bandwidth to which the link can be upgraded, and that the policy is monotonic, i.e. that the action is non-decreasing as the arrival probability increases. This means that once the policy selects a higher bandwidth, it will not select a lower bandwidth for higher arrival probabilities, and since there are only two bandwidths the policy is characterized by the threshold state where the higher bandwidth is selected. Intuitively, an increased arrival probability should not cause a lower bandwidth to be optimal, and since the policies of queue length based MDPs for service rate control are monotonic, cf. (Puterman, 1994), the assumption seems reasonable. There are now $|\Lambda| + 1$ possible positions of the threshold for each action in the policy, where $|\Lambda|$ is the number of states modeling the arrival probability, so iterating through all $(|\Lambda| + 1)^2$ combinations of thresholds and calculating the value of each policy the optimal policy can be found.

IV. Numerical results

Although the model can be used to decide for downgrades as well, we will focus on the upgrade scenarios for the subsequent parametric study.

As a default parameter set for the numerical examples of the application of the constrained MDP to obtain a utilization threshold, we use the following: For computational reasons the dimension of the state space is limited to $|\Lambda| = 15$ states controlling the arrival probability, and $k = 20$ queue states. For the Geo/Geo/1/k model with two actions (bandwidths) to choose from this gives a state space with 600 states.

The possible choices of link bandwidth, expressed by the service probabilities, are set to $a_1 = 0.1$ and $a_2 = 0.2$, and the arrival probabilities range from $\lambda_{\text{low}} = 0.01$ to $\lambda_{\text{high}} = 0.11$, such that utilizations from 0.1 up to values above 1 are possible. The probabilities $p = 10^{-3}$ and $q = 10^{-4}$ of an increase and decrease of average arrival probability are kept very low, as these express the long-term changes of the average arrival probabilities as observed during busy hours.

The remaining parameters, the holding cost h, the cost of the bandwidths b_1 and b_2, the cost of switching bandwidth c, and the penalty cost g for having a full buffer are given in table I. For the given parameter set and the MDP based on the Geo/Geo/1/k model, a utilization threshold of 0.61 results.

A. Impact of traffic model

First the impact of the traffic model is assessed. The threshold obtained for a Bernoulli arrival process is compared to burstier arrival processes, namely ON/OFF type traffic. This will determine if the traffic model of the arriving traffic has an impact on the threshold for upgrading.

The threshold is calculated for a Bernoulli arrival process, an ON/OFF arrival process with geometric ON and OFF periods, and an ON/OFF process with TPT ON periods, called 1-Burst according to (Schwefel and Lipsky, 1999, 2001). An important parameter of the ON/OFF traffic models is the *burstiness* of the arrival process, defined in (Antonious et al., 2002) as

$$b := \frac{\overline{\text{OFF}}}{\overline{\text{ON}} + \overline{\text{OFF}}},$$

where $\overline{\text{ON}}$ and $\overline{\text{OFF}}$ are the average ON and OFF times, respectively. If $b = 0$ the ON/OFF process is actually a Bernoulli process. This burstiness parameter is varied by setting $P_{\text{ON,ON}} = 0.96$ and varying $P_{\text{OFF,OFF}}$. This gives an average ON period of 24 which means that the buffer can be filled in a single ON period.

The upper graph in Figure 2 shows the resulting utilization threshold as calculated from the MDP plotted against the holding cost h, for Bernoulli ($b = 0$) and ON/OFF arrival processes, the latter having burstiness $b = 0.8$. Additionally the thresholds are plotted for different full buffer penalty costs $g = 20, 200$. The thresholds are overall lower for the ON/OFF process as compared to the Bernoulli process. With g low and low holding cost the threshold for the ON/OFF process coincides with the threshold for the Bernoulli process.

The lower graph in Figure 2 shows the same comparison of arrival processes with burstiness $b = 0, 0.8$ plotted against the reward per packet, and plotted for different costs of changing bandwidth $c = 20, 200$. Overall, the threshold is far less sensitive to this reward parameter than the holding cost, and again the threshold for the ON/OFF process is mostly lower than the threshold for the Bernoulli process. Similar results have been observed for other parametrizations of the reward function, not shown here.

The ON/OFF process above has geometrically distributed ON times, which is a discrete analog of the exponential distribution. For certain types of network traffic (e.g. resulting from http or ftp), the distribution of the ON periods may show power-tails, at least over some orders of magnitude, which will cause long-range dependent properties, or asymptotic second order self-similarity, as detected in the 90ies by (Leland

Parameter	Description	Value		Parameter	Description	Value
$\|\Lambda\|$	Evolution states	15		c	Upgrade cost	20
k	Queue states	20		g	Full buffer penalty	20
a_1	Current bw.	0.1		λ_{low}	Lowest arrival prob.	0.01
a_2	Upgraded bw.	0.2		λ_{high}	Highest arrival prob.	0.11
r	Reward	30		p	Inc. in arr. prob.	0.001
h	Holding cost	1		q	Dec. in arr. prob.	0.0001
b_1	Cost of current bw.	4		ϕ	Discount rate	0.99
b_2	Cost of upgraded bw.	8				

TABLE I: Parameters used in numerical experiments

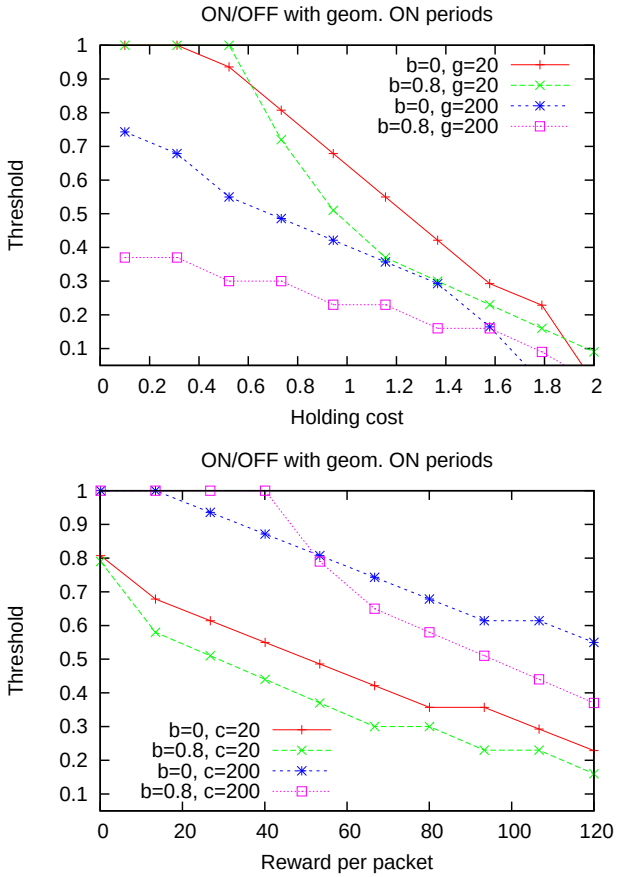


Fig. 2. Utilization threshold as a function of the holding cost and the reward per packet, respectively. The thresholds are shown both with a Bernoulli arrival process ($b = 0$) and ON/OFF arrival process with burstiness $b = 0.8$. Additionally the thresholds are shown for different full buffer penalty costs g and costs of changing bandwidth c.

et al., 1994) and others. Such phenomena may have significant impact on performance when it is caused by heavy-tailed ON times, (Schwefel and Lipsky, 2001). Since including true heavy-tailed ON times is not possible in the MDP model, and may also not be practically meaningful (no infinite tails will ever be observed in finite life-times of networks), a discrete analog of the truncated power-tailed (TPT), (Greiner et al., 1999), is used for the ON times, see Appendix A. This model is also called a 1-Burst process.

The parameters used for the 1-Burst process are shown in Appendix A, note only that T controls the length of the power-tail and α its slope. As an example on how these parameters influence the queue length distribution it can be noted that with burstiness of 0.75 and utilization 0.72 the probability of obtaining a queue length larger than 20 is around 10^{-1}, while it is around 10^{-3} for the geometric ON/OFF model.

The TPT ON periods do affect the threshold, as can be seen in Figure 3. The TPT ON periods results in lower threshold for lower holding costs, and the longer the power-tail (controlled by T), the lower the thresholds become. However, the slope of the power-tail (controlled by α) does not seem to influence the thresholds, see the right part of Figure 3. The threshold remains low for the scenario with TPT distributed ON period, even for very low holding costs. The threshold is kept low because of the increased probability of hitting the full buffer penalty. This means that even with low QoS demands, an upgrade should be performed at significantly lower utilization than with a Bernoulli arrival process.

In conclusion, burstiness and the distribution of the ON periods of the traffic appear to be important parameters for the utilization threshold. The holding cost ('costs of poor QoS') also showed a notable influence on the threshold, in combination with the distribution of the ON periods. Having geometrically distributed ON times however resulted in lower thresholds regardless of the holding cost, but with TPT ON periods the threshold is only different from the threshold of the Bernoulli process for lower holding costs.

B. Influence of available bandwidths

Focusing on a model with two bandwidths the influence of the bandwidths and their costs on the threshold is examined. The two bandwidths are the current bandwidth, $a_1 = 0.1$, and the bandwidth to which we can upgrade, $a_2 = 0.2$.

It is natural to expect that the threshold might depend on how large an upgrade is performed. Varying the *upgrade ratio* a_2/a_1, whilst keeping $a_1 = 0.1$ shows that the threshold decreases as the ratio increases. Remember, however, that the costs remain fixed while increasing the ratio. If the cost of a_2 is increased proportional to a_2 the reverse conclusion arises; the threshold goes to 1 as the ratio increases. This means that if the upgrade ratio is large then we let the current setup run

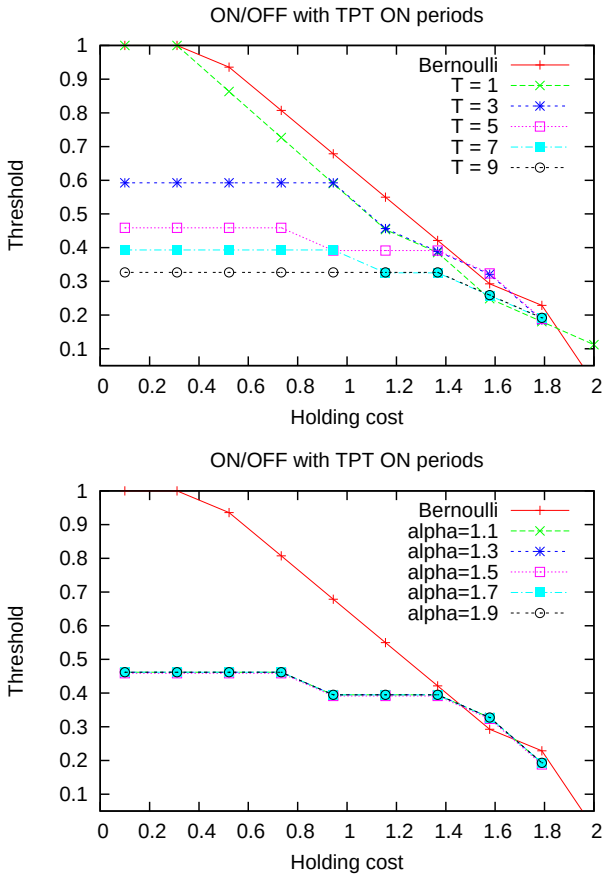


Fig. 3. Utilization threshold as a function of the holding cost varying the length (T) and slope (α) of the power-tail. If nothing else is noted $T = 5$ and $\alpha = 1.5$.

at higher utilization before upgrading, i.e. we are compromising somewhat on the QoS of the current setup of the network link.

Keeping the upgrade ratio constant (equal to 2) and increasing the current bandwidth a_1 also causes a change in the threshold. The threshold increases nearly linearly ending near 1 as a_1 goes towards 0.5, corresponding to a_2 near 1. An increase in a_1, and consequently a_2, will cause an increase in rewards (e.g. reward per packet) in comparison to the fixed costs (e.g. bandwidth change costs). We suspect that this is the cause of the dependence of the threshold on a_1, and have found empirically that adjusting the discount factor as

$$\phi = 0.1^{\frac{a_1}{40(1-a_1)}},$$

stabilizes the threshold, making the utilization threshold independent of a_1 as long as the upgrade ratio is kept constantly equal to 2.

C. Influence of prediction model for long-term traffic variations

The probabilities for an increase and decrease in arrival probability were $p = 0.001$ and $q = 0.0001$, respectively. The purpose of this section is to determine if these parameters, which represent a prediction model of the long-term fluctuations of the arrival process, have an impact on the utilization threshold.

Recall that we have assumed that the average arrival probability evolves as a birth-death process with parameters p for a birth, and q for a death, so the traffic prediction is essentially the parameters p and q. Varying the probability p shows no influence on the utilization threshold, until about 0.3 is reached. From 0.3 to 0.4 the utilization threshold results in values from 0.61 to 0. However, in this range of p, the time scales of the changes of the average arrival probability are now so short that they are in the order or below the time scales of short-term traffic fluctuations; hence outside the range of the scenarios that the model was originally designed for. This means, excluding extreme values, that the predicted time scales at which the arrival probability increases does not have an effect on the utilization threshold

The threshold is similarly unaffected for q below $p = 0.001$. From 0.01 to 0.07 the utilization threshold increases nearly linearly from 0.6 to 0.95. Again q is in the order of the arrival probability, which lies outside the scope of the assumptions of the model. Thus if the traffic grows sufficiently slow the parameters of the prediction model do not influence the utilization threshold.

V. Conclusion

This paper develops and analyzes an approach to quantitatively obtain a utilization threshold for bandwidth upgrades (or downgrades) of links consisting of leased lines. The approach is based on constraint policies in Markov decision processes; the constraining allows to obtain policies that only depend on long-term average arrival probabilities and not on queue-length or short term arrival probability fluctuations. The model is formulated both for Poisson traffic as well as for bursty ON/OFF type traffic.

In practical network operation, the quantitative model would be applied offline. The operator would use traffic measurements during individual busy hours to calibrate the short-term traffic model (in the example of the paper ON/OFF or Poisson) and then uses his long-term average traffic measurements to calibrate the prediction model. The reward function is mainly determined by the business model and Service Level Agreements of the operator. These three elements are used to setup the MDP model which then results in a quantification of an upgrade threshold, which subsequently can be used over a period of several months for upgrade decisions. Hence, the computational part can be done offline and only the simple utilization based approach is used for the online upgrade decisions. Consequently, no additional operational complexity results for an operator.

The resulting upgrade thresholds for a specific class of reward functions was subsequently examined numerically. Increased burstiness generally lowered the threshold and having (truncated) power-tailed ON periods, and thus long-range dependent properties, also showed a notable influence on the threshold, with longer tails giving lower thresholds. The MDP ap-

proach quantifies this decrease of the threshold. The available bandwidths and their costs also influence the model, most notably on the ratio between the bandwidths. The prediction model of changes of the long-term average arrival probability showed only marginal influence on the utilization threshold.

VI. Acknowledgments

This work has been supported by the Danish Research Council via the Centre for Network and Service Convergence (CNTK). The authors would like to thank the CNTK partners for their helpful comments, in particular Jan Bøgh and Peter N. Andersen from Sonofon, Aalborg. Furthermore, the authors would like to thank Imad Antonious, Southern Connecticut State University, for the inspiring discussions.

References

I. Antonious, L. Lispky, H.-P. Schwefel, M. Greiner, and M. Jobmann. Comparison of the analytic N-Burst model with other self-similar ON-OFF approximations to telecommunications traffic. In *Proceedings of the IEEE International Symposium on Network Computing and Applications*, Boston, February 2002.

M. Greiner, M. Jobmann, and L. Lipsky. The importance of power-tail distributions for telecommunication traffic models. *Operations Research*, 47(2):313–326, 1999.

H. Heffes. A class of data traffic processes - covariance function characterization and related queueing results. *Bell Systems Technical Journal*, 59(6):897–929, 1980.

D. P. Heyman. Optimal operating policies for M/G/1 queueing systems. *Operations Research*, 16(2):362–382, 1968.

G. V. Hoey, S. V. den Bosch, P. de La Vallee-Poussin, H. D. Neve, and G. Petit. Capacity planning strategies for voice-over-IP traffic in the core network. In *High Performance Switching and Routing, 2001 IEEE Workshop on*, pages 109–112, 2001.

A. Hordijk and J. A. Loeve. Undiscounted markov decision chains with partial information; an algorithm for computing a locally optimal periodic policy. *Mathematical Methods of Operations Research*, 40(2):163–181, 1994.

S. Jagannathan, J. Altmann, and L. Rhodes. A revenue-based model for making resource investment decisions in IP networks. In *Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on*, pages 185–197, 2003.

W. Leland, M. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, February 1994.

K. Y. Lim, S. Soh, and S. Rai. Computer communication network upgrade for optimal capacity related reliability. In *Communications, 2005 Asia-Pacific Conference on*, pages 1102–1106, 2005.

S. A. Lippman. Semi-Markov decision process with unbounded rewards. *Management Science*, 19(7):717–731, 1973.

M. Menth. A scalable protocol architecture for end-to-end signaling and resource reservation in IP networks. In *Proceedings of the 17th International Teletraffic Congress*, pages 211–222, 2001.

Y. A. Phillis and R. Zhang. Fuzzy service control of queueing systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 29(4):503–517, 1999.

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

S. Schneeberger. Markov-entscheidungs-prozesse mit abhängigen aktionen für optimale reparaturmaßnahmen bei unvollständiger information. *OR Spektrum*, 14(2):71–78, 1992.

H.-P. Schwefel and L. Lipsky. Impact of aggregated, self-similar ON/OFF traffic on delay in stationary queueing models (extended version). *Performance Evaluation*, 43(4):203–221, 2001.

H.-P. Schwefel and L. Lipsky. Performance results for analytic models of traffic in telecommunication systems, based on multiple ON-OFF sources with self-similar behavior. In *Teletraffic Engineering in a Competitive World*, volume 3A. Elsevier, 1999.

J. L. Smith. Markov decisions on a partitioned state space. *IEEE Transactions on Systems, Man and Cybernetics*, 1(1):55–60, 1971.

M. Sobel. Optimal operation of queues. In *Mathematical Methods in Queueing Theory*, pages 231–261. Springer-Verlag, 1974.

T. Tsuchiya and Y. Takahashi. On discrete-time single-server queues with Markov modulated batch Bernoulli input and finite capacity. *Journal of the Operations Research Society of Japan*, 36(1):29–45, 1993. URL `http://ci.nii.ac.jp/naid/110001184370/en/`.

R. R. Weber and S. S. Jr. Optimal control of service rates in networks of queues. *Advances in Applied Probability*, 19(1): 202–218, 1987.

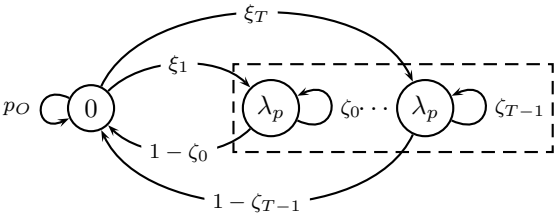


Fig. 4. Transition diagram of the discrete ON/OFF process with TPT ON periods. The dashed box shows the ON states.

Appendix

I. Discrete 1-Burst model

In order to investigate traffic with long-range dependent properties, a discrete time equivalent to the continuous time 1-Burst ON/OFF model from (Schwefel and Lipsky, 2001) is specified here and used in the main part of the paper. Generalization to aggregated ON/OFF models via Kronecker product representation is straightforward.

The discrete 1-Burst model uses the discrete equivalent of truncated Power Tail distributions (Greiner et al., 1999) for the ON period, see Figure 4.

The entrance probabilities ξ_i decay geometrically with parameter θ, while the mean state-holding time increases geometrically, as reflected by a geometric increase of the probabilities ζ_i with parameter γ. This will cause some ON periods to be very long resulting in the power-tail alike behavior of the ON times, see (Greiner et al., 1999) for proofs in the continuous case. The parameters used in Section IV-A are $\alpha = 1.5, \theta = 0.5, T = 5$. The parameter α sets the slope of the power-tail and it is determined by the two geometric factors for the entrance and state-holding probability, see (Greiner et al., 1999) for the relation. T determines the number of phases and therefore the Power-tail range, see (Schwefel and Lipsky, 2001) for a more formal definition. The entrance probabilities result as

$$\xi_i = (1 - p_O)\frac{1-\theta}{1-\theta^T}\theta^i,$$

and the state-holding probabilities as

$$\zeta_i = \frac{\gamma^i}{\gamma^i + \mu}$$

with $\gamma = 1/\theta^{1/\alpha}$ and μ is a variable used to fix the mean of the overall TPT distribution in the same way as in (Greiner et al., 1999).

PERFORMANCE OF DOWNLINK SHARED CHANNELS IN UMTS SCENARIOS USING MARKOV FLUID PROCESS

Marie-Ange Remiche
Miguel de Vega Rodrigo
Université Libre de Bruxelles
CoDe - CP 165/15,
Av. F.D. Roosevelt
B-1050 - Bruxelles, Belgium
email: [mremiche,mdevegar]@ulb.ac.be

Laurent Schumacher
Hugues Van Peteghem
FUNDP - The University of Namur
Computer Science Institute,
Rue Grandgagnage 21
B-5000 - Namur, Belgium
email:[lsc,hvp]@info.fundp.ac.be

KEYWORDS
Markov driven fluid flow model, UMTS network.

ABSTRACT

In this work, we propose a new model of the Downlink Shared Channel (DSCH) buffer in UMTS Release '99 cellular systems. Under our assumptions, the available bandwidth dedicated to DSCH traffic is limited by voice connections carried by Dedicated Channels (DCHs) on the one hand, and by traffic on the Forward Access Channel (FACH) on the other hand. We propose to model the DSCH buffer content as a Markov driven fluid queue with jumps and to derive several performance measures, namely stationary DSCH buffer content, bit loss probability and overflow probability. These measures are obtained through adequate transformation of the original fluid queue.

INTRODUCTION

3G/UMTS has been specified as an integrated solution for mobile voice and data with wide area coverage. In its initial phase (Release '99) it offers theoretical bit rates of up to 384 kbps in high mobility situations, rising as high as 2 Mbps in stationary/nomadic user environments. In its latest releases (Releases 5 and 6) the use of High Speed Downlink and Uplink Packet Access technologies (HSDPA and HSUPA, respectively) increase theoretical downlink speeds as high as 14.4 Mbps (and respectively 5.8 Mbps uplink).

In the praxis, a number of factors draw UMTS performance away from such theoretical limits. The users being served by a base station must share common, limited radio resources. Thus, the capacity experienced by each one of them is not only affected by the characteristics of his/her transmission (e.g. user application, connection type, modulation scheme, spreading factor, distance to the base station and scheduling scheme in HSDPA), but also by the number of simultaneously active users and their own transmission characteristics. These cross-dependencies represent a major challenge when trying to describe the behaviour of each shared channel in a certain scenario.

In spite of these difficulties this problem has received much attention by researchers and UMTS developers due to its enormous relevance [4, 5] and references therein. Most of the studies so far, like [1], have focused on the impact of the physical layer on the available bandwidth. However, there is a lack of understanding at system level, where the cross-dependencies among different users play a major role. This stage is extremely important for the performance experienced by the end users since it explains how the bandwidth available at the physical layer is actually distributed among them. Our main focus in this paper is precisely to develop a Markov driven fluid model which includes the effects at system level. To the best knowledge of the authors, such an analytical model has not been proposed for UMTS cellular networks yet.

In this work, we extend some of the most recent results in the field of Markov driven fluid process in order to analyse the performance of a model of UMTS DSCH channel. Numerical computations of the derived analytical model are checked against emulation results obtained with an IPv6, Linux-based testbed mimicking the behaviour of a UMTS Terrestrial Radio Access Network segment [10].

This paper is organized as follows. In the first section, we completely describe the system under analysis and highlight input stochastic parameters of the model. Next, we derive the resulting Markov driven fluid model that exhibits jumps. Performance measures of interest, that is overflow and loss probability and stationary buffer content distribution, are derived in the following section. In the Numerical Analysis Section we validate our model by comparing numerical computations to emulation results on the testbed. We finally highlights the future work items we are currently working on.

UMTS Scenario

Let consider the downlink of a Release'99 Universal Mobile Telecommunication System (UMTS) cell. The base station, called NodeB, supervises the delivery of traffic to the users it is serving. This downlink traffic is split into four classes, depending on the nature of data to be transmitted. Real-Time (RT) traffic falls into Conversational and Streaming classes, whereas Non Real-Time (NRT) traffic is identified as either Interactive or Background

traffic. These four classes of traffic are delivered over the air through three different transport channels. The most exclusive of them, the Dedicated Channel (DCH), only carries traffic related to a given user. The user enjoys thus the full benefit of the resources allocated to that transport channel. On the other hand, the two other transport channels, the Downlink Shared Channel (DSCH) and the Forward Access Channel (FACH), are shared by the users, meaning that their data are multiplexed within these channels. The fact that a given user traffic ends up in a DSCH or a FACH depends whether the user is simultaneously transmitting through a DCH or not. When a user has already access to a DCH, its NRT traffic can be transported through a DSCH. With no open DCH connection, his/her traffic will go through the FACH.

In the present paper, we will study the fluctuations of the shared DSCH bandwidth, considering the competing activity of open DCHs and the FACH of the cell. For the sake of simplicity, we will assume that the data is spread with a Spreading Factor (SF) 64, and is half-rate coded, which means that each transport channel is eager to achieve a bandwidth multiple of 30 kbps. At any given time, we regard the full downlink bandwidth to be split into four types of transport channels:

1. The FACH of the cell (bandwidth fixed to 30 kbps),
2. The $K(t)$ DCHs transporting the $K(t)$ currently active voice connections (each with 30 kbps),
3. The DSCH carrying the signaling of the open DCHs (fixed to 30 kbps),
4. The DSCH transporting the NRT traffic of the users already having a DCH.

We do not constrain the mapping of transport channels to the Orthogonal Variable Spreading Factor (OVSF) tree as in [5]. As a result, the DSCH enjoys the residual bandwidth to its full extend, as shown in Figure 1. Actually, in a real UMTS system, a single DSCH carries both DCH signaling and NRT traffic. We separate these traffics in order to isolate the fluctuating part of the shared DSCH bandwidth. Finally, we will assume perfect Fast Power Control. Consequently, every user transmitting in a DCH will enjoy the full prescribed 30 kbps bandwidth, despite of fading, shadowing, intra- and inter-cell interference. With this model, a NodeB can process up to 62 simultaneous voice connections.

We model the DSCH buffer at layer 2. NRT traffic is composed of bursts of random size generated by the active users, that instantaneously accumulate in the shared DSCH downlink buffer as delivered by the higher layer. As a result, the model of the NRT traffic captures the fluctuating number of active users. The more active users, the more intense the NRT traffic.

Since we assume perfect Fast Power Control, the transmit power of a given mobile user is continuously adjusted in order to achieve a fixed Block Error Rate

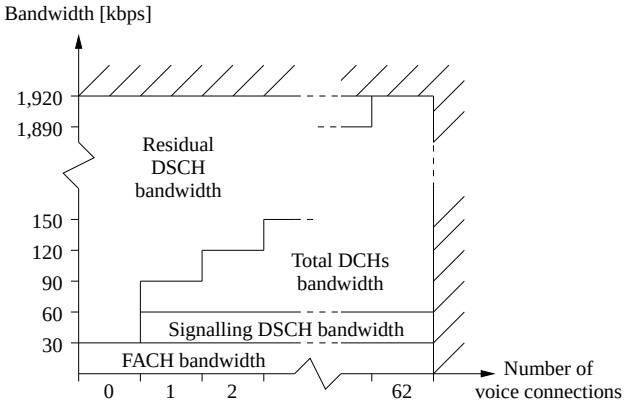


Figure 1: Residual DSCH bandwidth vs. number of voice connections

(BLER) at reception. Assuming also that the NRT traffic is handled by the Radio Link Control (RLC) in Transparent Mode, there is no reliability mechanism in layer 2, i.e. no link retransmissions. Additionally, we regard this specific NRT traffic to be transported on User Datagram Protocol (UDP), such that we do not have to take layer 4 retransmissions into account in the traffic model.

Traffic is then transmitted at a speed that depends on $K(t)$. This transmission speed $R(t)$ is determined as follows:

$$R(t) = \begin{cases} \{64 - [K(t) + 2]\} \,.\, 30\,\text{kbps} \\ \qquad \text{when } 1 \leq K(t) \leq 62 \\ 63 \,.\, 30\,\text{kbps} \\ \qquad \text{when } K(t) = 0 \end{cases} \tag{1}$$

The cellular network industry is actually rather focusing on HSxPA and Long-Term evolutions. In that perspective, this Rel'99 scenario is used to calibrate the comparison of the numerical computations of the analytical model against the emulation results of the testbed. A next step will be to address HSxPA scenarios, where the variation of the bandwidth will no longer depend on the fluctuating number of active voice sessions, but rather on the selected Modulation and Coding Scheme (MCS) based on the Channel Quality Indicator (CQI) fed back by the receiver to the transmitter.

MARKOV DRIVEN FLUID MODEL

This section is composed of three parts. First, we briefly recall some background definitions. Next, we specify the stochastic processes assumed in the system we model. Finally, we put into evidence the resulting Markov driven fluid model, that will be further analysed.

Background definitions

Let $\{L(t), t \in I\!R^+\}$ be an absorbing Markov process with generator

$$\begin{pmatrix} \mathbf{0} & 0 \\ G & \boldsymbol{g} \end{pmatrix},$$

and initial phase distribution $(0\ \boldsymbol{\alpha})$, defined on the state-space $\{0, 1, 2, \ldots, n\}$, with n finite and 0 being the absorbing state. The time to absorption X is a continuous Phase-type random variable of order n with parameter $(\boldsymbol{\alpha}, G)$; and

$$P[X \leq x] = 1 - \boldsymbol{\alpha} \exp\{Gx\}\mathbf{1},$$

with $x \in I\!R^+$.

We say that $\{(N(t), J(t)), t \in I\!R^+\}$ is a stationary Markovian Arrival Process (MAP) with representation (D_0, D_1) and initial stationary distribution $\boldsymbol{\xi}$ if

- $\{(N(t), J(t)), t \in I\!R^+\}$ is a bidimensional Markov process defined on $I\!N \times \{1, \ldots, m\}$ with m finite,
- whose the corresponding generator is

$$\begin{pmatrix} D_0 & D_1 & 0 & \ldots \\ 0 & D_0 & D_1 & \ldots \\ 0 & 0 & D_0 & \ldots \\ & & & \ddots \end{pmatrix},$$

- and $\boldsymbol{\xi}$ is the minimal non-negative solution of

$$\begin{aligned} \boldsymbol{\xi}(D_0 + D_1) &= \mathbf{0} \\ \boldsymbol{\xi}\mathbf{1} &= 1 \end{aligned}$$

Mathematical model assumptions

We make the following system assumptions. First, voice communications are initiated randomly in time as a Poisson process with parameter λ and have an exponential random duration, with parameter μ. According to the system's description, the number $K(t)$ of active voice communications at time t is determined by a $M/M/62/62$ queue, with generator:

$$A = \begin{pmatrix} -\lambda & \lambda & 0 & \ldots & 0 \\ \mu & -(\lambda+\mu) & \lambda & \ldots & 0 \\ 0 & 2\mu & -(\lambda+2\mu) & \ldots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \ldots & -(62\mu) \end{pmatrix}.$$

Second, we assume that successive burst sizes are independent and identically distributed PH-distributed random variables with parameters $(\boldsymbol{\alpha}, G)$, while random time epochs $\{T_n; n \in I\!N_0\}$ are given by a MAP with representation (D_0, D_1) and stationary distribution δ. Such an assumption is not restrictive due to the ability of the MAP to mimic self-similar and bursty traffic.

Resulting Model

Our objective is to model the DSCH buffer content. Because a bit can be considered as an infinitesimal quantity with respect to the maximum buffer capacity B, the DSCH buffer content $\{X^B(t)\}$ can be modelled as a fluid process.

Let us assume that the buffer is initially empty. It will remain so as long as the system records no burst arrival. When entering the buffer, a burst is instantaneously available for transmission. Between two consecutive burst arrivals, the buffer content decreases at a speed determined by $R(t)$, as defined in (1). When $K(t) = 62$, the buffer content remains at the same level. When the burst size is larger than the available buffer capacity, only the exceeding burst part is lost, while in real systems, all bits of the packet that is responsible for the overtaking would be lost. However, in our analysis, such an assumption will reveal acceptable in the light of results presented in the Numerical Analysis Section. Upon touching level B, the buffer content immediately decreases or stays at that level in case $K(t) = 62$.

The fluid queue $\{X^B(t)\}$ evolution is completely driven by a particular Markov process $\{\phi(t)\}$ defined on a state-space $\mathcal{S}$. We also define the processes $\{\tilde{X}^B(t)\}$ and $\{\bar{X}^B(t)\}$ as follows. The first is obtained by replacing each jump occurring in $\{X^B(t)\}$ by an interval of time where the process continuously increases at a rate of 1. The latter $\{\bar{X}^B(t)\}$ is obtained through an adequate transformation of the process $\{\tilde{X}^B(t)\}$: level decreases now occur instantaneously and correspond to downwards jumps. Each jump's amplitude is equal to the total level decrease observed over time periods where $\{\tilde{X}^B(t)\}$ was continuously decreasing. According to the definitions above, the evolution of $\{X^B(t)\}, \{\tilde{X}^B(t)\}$ and $\{\bar{X}^B(t)\}$ are driven by the same Markov process $\{\phi(t)\}$, the only difference being the behaviour of each fluid queue as a function of the phase visited by $\{\phi(t)\}$.

In the appendix, we completely detail the construction of the generator of the Markov process $\{\phi(t)\}$ and explain how it drives the content of the DSCH buffer. It results that the state-space $\mathcal{S}$ might be decomposed into three disjoint subsets $\mathcal{S}_0 \cup \mathcal{S}_- \cup \mathcal{S}_u$ regrouping phases that correspond to null, negative rates of evolution and finally phases that make the DSCH buffer content instantaneously increases.

PERFORMANCE MEASURES

Our objective is threefold. We aim at characterizing first the stationary buffer content, next the bit loss probability $P_L(B)$ and finally the overflow probability $P_O(B)$.

The bit loss probability, as defined in Lin et al. [7] is the fraction of average lost work divided by the average total workload in the same period of time. The overflow probability may be approximated with the well-known tail probability method as suggested in Sakasegawa et al. [9] for example.

Stationary buffer content distribution

Our approach can be summarized as extending the results of Dzial et al [3] to the finite buffer case analysis. The stationary distribution of the process $\{\tilde{X}^B(t)\}$ is first characterized. With an appropriate transformation, the sta-

tionary distribution of the process $\{X^B(t)\}$ is next completely obtained. We put ourselves in the specific context of the process described earlier but it is a simple matter to extend our analysis to any kind of jump processes.

As mentioned in the appendix, the state-space of $\{\phi(t)\}$ might be decomposed into three disjoint subspaces, those are $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_- \cup \mathcal{S}_+$, following a same decomposition for the generator T, we get:

$$T = \begin{pmatrix} T_{00} & T_{0-} & T_{0+} \\ T_{-0} & T_{--} & T_{-+} \\ T_{+0} & T_{+-} & T_{++} \end{pmatrix}.$$

It is a simple matter to determine the elements of this matrix according to their exact content that is provided in the appendix.

We assume T is irreducible and let $\boldsymbol{\xi}$ be its stationary probability vector, that is $\boldsymbol{\xi} = (\boldsymbol{\xi}_0, \boldsymbol{\xi}_-, \boldsymbol{\xi}_+)$. In our case study, both $\mathcal{S}_+$ and $\mathcal{S}_-$ are not empty, in which case the fluid queue is positive recurrent for any value of the drift $\mu = \boldsymbol{\xi}_+ - \boldsymbol{\xi}_- \boldsymbol{r}_-$, where $\boldsymbol{r}_-$ is a vector composed of different values of $R(t)$ determined for $K(t) = i$ with $(i, j, 0) \in \mathcal{S}_-$.

This finite buffer case was studied in da Silva and Latouche, see [2] and we propose here to recall some of their results which are of interest in the sequel.

Let us first define the matrix $\dot{T}$ as

$$\begin{aligned} \dot{T} &= \begin{pmatrix} \dot{T}_{++} & \dot{T}_{+-} \\ \dot{T}_{-+} & \dot{T}_{--} \end{pmatrix} \\ &= \begin{pmatrix} C_+^{-1} & 0 \\ 0 & C_-^{-1} \end{pmatrix} \left\{ \begin{pmatrix} T_{++} & T_{+-} \\ T_{-+} & T_{--} \end{pmatrix} \right. \\ &\quad \left. + \begin{pmatrix} T_{+0} \\ T_{-0} \end{pmatrix} (-T_{00})^{-1} \begin{pmatrix} T_{0+} & T_{0-} \end{pmatrix} \right\} \end{aligned}$$

with $C_+ = I$ and C_- being a diagonal matrix obtained from the vector $\boldsymbol{r}_-$. We define C as a diagonal matrix composed of the different rates of the system. A key matrix is the matrix Ψ, that is the minimal non-negative solution of the following algebraic Riccati equation:

$$\dot{T}_{+-} + \dot{T}_{++}\Psi + \Psi\dot{T}_{--} + \Psi\dot{T}_{-+}\Psi = 0,$$

such that $\Psi\mathbf{1} = \mathbf{1}$. Based on Ψ, we define

$$\begin{aligned} U &= \dot{T}_{--} + \dot{T}_{-+}\Psi, \\ K &= \dot{T}_{++} + \Psi\dot{T}_{-+}. \end{aligned}$$

It has to be noted that efficient matrix analytic algorithms have been developed to obtain Ψ, see Ramaswami [8].

Another key matrix is $\hat{\Psi}$, defined as the minimal non-negative solution of the next algebraic Riccati equation:

$$\dot{T}_{-+} + \dot{T}_{--}\hat{\Psi} + \hat{\Psi}\dot{T}_{++} + \hat{\Psi}\dot{T}_{+-}\hat{\Psi} = 0.$$

We ultimately build up the next matrices:

$$\begin{aligned} \hat{U} &= \dot{T}_{++} + \dot{T}_{+-}\hat{\Psi}, \\ \hat{K} &= \dot{T}_{--} + \hat{\Psi}\dot{T}_{+-}. \end{aligned}$$

We define the vector $\boldsymbol{p}^0 = (\boldsymbol{p}_0^0, \boldsymbol{p}_-^0, \mathbf{0})$ (respectively $\boldsymbol{p}^B = (\boldsymbol{p}_0^B, \mathbf{0}, \boldsymbol{p}_+^B)$)) as the probability mass vector of the fluid process $\{X^B(t)\}$ at level 0 (respectively level B). The density probability vector $\boldsymbol{\pi}^B(x)$ is composed of $(\boldsymbol{\pi}_0^B(x), \boldsymbol{\pi}_-^B(x), \boldsymbol{\pi}_+^B(x))$, for $0 < x < B$.

In da Silva and Latouche [2], the following results are established. Let us first define the matrices $\Lambda_{++}^{(B)}, \Psi_{+-}^{(B)}, \hat{\Psi}_{-+}^{(B)}$ and $\hat{\Lambda}_{--}^{(B)}$ as the solutions of the following system:

$$\begin{pmatrix} \Lambda_{++}^{(B)} & \Psi_{+-}^{(B)} \\ \hat{\Psi}_{-+}^{(B)} & \hat{\Lambda}_{--}^{(B)} \end{pmatrix} \begin{pmatrix} I & e^{KB}\Psi \\ e^{\hat{K}B}\hat{\Psi} & I \end{pmatrix} = \begin{pmatrix} e^{\hat{U}B} & \Psi \\ \hat{\Psi} & e^{UB} \end{pmatrix}$$

whose coefficient matrix is nonsingular if $\mu \neq 0$.

The next theorem establishes equations for the stationary probability distribution of the fluid process without any jumps.

Theorem 1 *If $\mu \neq 0$, the stationary density of the finite buffer fluid queue is given by*

$$(\boldsymbol{\pi}_+^B(x), \boldsymbol{\pi}_-^B(x)) = \gamma(\boldsymbol{\nu}_+^B, \boldsymbol{\nu}_-^0)|\dot{C}|^{-1} \begin{pmatrix} e^{Kx} & e^{Kx}\Psi \\ e^{\hat{K}(B-x)}\hat{\Psi} & e^{\hat{K}(B-x)} \end{pmatrix}$$

where

$$(\boldsymbol{\nu}_+^B, \boldsymbol{\nu}_-^0) = (\boldsymbol{\eta}_+^B, \boldsymbol{\eta}_-^0) \begin{pmatrix} 0 & \dot{T}_{+-} \\ \dot{T}_{-+} & 0 \end{pmatrix} \begin{pmatrix} I & e^{KB}\Psi \\ e^{\hat{K}B}\hat{\Psi} & I \end{pmatrix}^{-1},$$

with

$$\dot{C} = \begin{pmatrix} C_+ & 0 \\ 0 & C_- \end{pmatrix}.$$

The parameter γ is $\boldsymbol{\xi}|C|\mathbf{1}$ and the vector $(\boldsymbol{\eta}_+^B, \boldsymbol{\eta}_-^0)$ is

$$(\boldsymbol{\eta}_+^B, \boldsymbol{\eta}_-^0) = (\dot{\boldsymbol{\xi}}_+\Lambda_{++}^{(B)}, \dot{\boldsymbol{\xi}}_-\hat{\Lambda}_{--}^{(B)})$$

where $\dot{\boldsymbol{\xi}} = (\dot{\boldsymbol{\xi}}_+, \dot{\boldsymbol{\xi}}_-)$ respects $\dot{\boldsymbol{\xi}}\dot{T} = \mathbf{0}$ and $\dot{\boldsymbol{\xi}}\mathbf{1} = \mathbf{1}$. Finally,

$$\boldsymbol{\pi}_0^B(x) = (\boldsymbol{\pi}_+^B(x), \boldsymbol{\pi}_-^B(x)) \begin{pmatrix} T_{+0} \\ T_{-0} \end{pmatrix} (-T_{00})^{-1}.$$

Furthermore, the mass probability vector is given by the following theorem.

Theorem 2 *The vector $\boldsymbol{p}^0$ is such that*

$$\begin{aligned} \boldsymbol{p}_-^0 &= \gamma\boldsymbol{\eta}_-^0(C_-)^{-1} \\ \boldsymbol{p}_0^0 &= \boldsymbol{p}_-^0 T_{-0}(-T_{00})^{-1} \end{aligned}$$

and $\boldsymbol{p}^B$

$$\begin{aligned} \boldsymbol{p}_+^B &= \gamma\boldsymbol{\eta}_+^B(C_+)^{-1} \\ \boldsymbol{p}_0^B &= \boldsymbol{p}_+^B T_{+0}(-T_{00})^{-1} \end{aligned}$$

As it is stated in Latouche and Takine [6], the stationary distribution of the original fluid queue can be obtained by an appropriate transformation of the stationary distribution of the fluid queue $\{\tilde{X}^B(t)\}$. In our case, the probability mass in increasing phase has not to be taken into account in the fluid queue with upwards jumps. This is stated in the next theorem.

Theorem 3 *For $0 < x < B$, the stationary density vector $\boldsymbol{\zeta}^B(x)$ of the fluid queue with upwards jumps $\{X^B(t)\}$ is given by*

$$\boldsymbol{\zeta}^B(x) = (\boldsymbol{\zeta}_0^B(x), \boldsymbol{\zeta}_-^B(x), \boldsymbol{\zeta}_+^B(x)) = \delta(\boldsymbol{\pi}_0^B(x), \boldsymbol{\pi}_-^B(x), \mathbf{0}).$$

The stationary probability mass vector $\boldsymbol{\rho}^0$ of level 0 of the process $\{X^B(t)\}$ is given by

$$\boldsymbol{\rho}^0 = (\boldsymbol{\rho}_0^0, \boldsymbol{\rho}_-^0, \boldsymbol{\rho}_+^0) = \delta(\boldsymbol{p}_0^0, \boldsymbol{p}_-^0, \mathbf{0}),$$

similarly, we have for the stationary probability mass vector $\boldsymbol{\rho}^B$ of level B

$$\boldsymbol{\rho}^B = (\boldsymbol{\rho}_0^B, \boldsymbol{\rho}_-^B, \boldsymbol{\rho}_+^B) = \delta(\boldsymbol{p}_0^B, \mathbf{0}, \mathbf{0}),$$

where δ is a normalization factor, that is:

$$\delta = \left(\boldsymbol{\rho}^0 \mathbf{1} + \int_0^B \boldsymbol{\zeta}(x)\mathbf{1}dx + \boldsymbol{\rho}^B \mathbf{1} \right)^{-1}.$$

The bit loss probability

We propose to obtain the bit loss probability by an adequate transformation of the model $\{\tilde{X}^B(t)\}$ into $\{\bar{X}^B(t)\}$. Those two processes were introduced in Resulting Model Subsection.

Let the vector $\bar{\boldsymbol{p}}^0 = (\bar{\boldsymbol{p}}_0^0, \bar{\boldsymbol{p}}_-^0, \mathbf{0})$ (respectively $\bar{\boldsymbol{p}}^B = (\bar{\boldsymbol{p}}_0^B, \mathbf{0}, \bar{\boldsymbol{p}}_+^B)$) be defined as the probability mass vector of the fluid process $\{\bar{X}^B(t)\}$ at level 0 (respectively level B). The density probability vector $\bar{\boldsymbol{\pi}}^B(x)$ is composed of $(\bar{\boldsymbol{\pi}}_0^B(x), \bar{\boldsymbol{\pi}}_-^B(x), \bar{\boldsymbol{\pi}}_+^B(x))$, for $0 < x < B$.

Following again an approach similar to that of Dzial et al. [3], we obtain the next result that concerns the stationary probability of the fluid process $\{\bar{X}^B(t)\}$, where only downward jumps occur.

Theorem 4 *For $0 < x < B$, the stationary density vector $\bar{\boldsymbol{\pi}}^B(x)$ of the fluid queue with downwards jumps only is given by*

$$\bar{\boldsymbol{\pi}}^B(x) = (\bar{\boldsymbol{\pi}}_0^B(x), \bar{\boldsymbol{\pi}}_-^B(x), \bar{\boldsymbol{\pi}}_+^B(x)) = \delta(\mathbf{0}, \mathbf{0}, \boldsymbol{\pi}_+^B(x)).$$

The stationary probability mass vector $\bar{\boldsymbol{p}}^0$ of level 0 of the process with jumps is given by

$$\bar{\boldsymbol{p}}^0 = (\bar{\boldsymbol{p}}_0^0, \bar{\boldsymbol{p}}_-^0, \bar{\boldsymbol{p}}_+^0) = (\mathbf{0}, \mathbf{0}, \mathbf{0}),$$

similarly, we have for the stationary probability mass vector $\bar{\boldsymbol{p}}^B$ of level B

$$\bar{\boldsymbol{p}}^B = (\bar{\boldsymbol{p}}_0^B, \bar{\boldsymbol{p}}_-^B, \bar{\boldsymbol{p}}_+^B) = \delta(\mathbf{0}, \mathbf{0}, \boldsymbol{p}_+^B),$$

where δ is a normalization factor, that is:

$$\delta = \left(\int_0^B \bar{\boldsymbol{\pi}}(x)\mathbf{1}dx + \bar{\boldsymbol{p}}^B \mathbf{1} \right)^{-1}.$$

In the framework of $\{\bar{X}^B(t)\}$, the loss probability is readily obtained by noting that $\bar{\boldsymbol{p}}_+^B$ bits are in average lost over a total workload of $\int_0^B \bar{X}_+^B(x)\mathbf{1}dx + \bar{\boldsymbol{p}}_+^B$ bits; that is $P_L(B) = \bar{\boldsymbol{p}}_+^B$.

Indeed, we immediately observe that over a unit period, the time spent by $\bar{X}^B(t)$ in levels $(0, B]$ increasing can be also interpreted as the total workload the system has to transmit; while the time spent in level B with an increasing phase is exactly the lost part of this workload.

The overflow probability

This performance measure is obtained by considering the same fluid process with no buffer limit, that is $\{X(t)\}$. This fluid queue is still driven by the evolution of the phase process $\{\phi(t)\}$. As mentioned earlier, we approximate the overflow probability $P_O(B)$ with the tail distribution method, that is

$$\begin{aligned} P_O(B) &= \lim_{t\to\infty} P[X(t) \geq B] = \int_B^\infty \boldsymbol{\pi}(x)\mathbf{1}dx \\ &= 1 - \left(\int_0^B \boldsymbol{\pi}(x)\mathbf{1}dx + p_0 \right) \end{aligned}$$

Again, we transform this fluid queue with jumps into a fluid queue such that, each time there is an upwards instantaneous jump, we replace it by an interval of time, of length equal to the size of the jump, and during which the fluid decreases at a rate 1. This exactly what was studied in Dzial et al. [3].

NUMERICAL ANALYSIS

We present a short numerical analysis of our model. Results presented here are of two types. First, we measure the impact of burst arrival rate on the stationary DSCH buffer content. Next, we illustrate the effect of burst arrival burstiness on the stationary DSCH buffer content.

We use the following parameters. Burst arrivals form a Poisson process in time of rate λ, their size is in average 18×30 kbits. Indeed, the Poisson process is a particular case of MAP. Voice communication appears at a rate of one every 100 seconds in average and lasts for 40 seconds in average. Buffer size is about 50×30 kbits.

We present in Figure 2 the distribution function of the stationary buffer content when λ ranges from 1 to 3. The curves represent the outcome of the numerical computations of the analytical model, while the markers stand for the emulation results obtained on the testbed [10]. Each emulation run lasted 600s, with up to 62 simultaneously active users.

Clearly, there is a good match between the analytical model and the results of the testbed. A goodness-of-fit test, the one-sample Kolmogorov-Smirnov test, has successfully been applied to the samples. Moreover, the previously made assumption, that is the limitation of the loss to the exceeding part of the burst when the buffer is saturated, does not significantly affect the validity of the

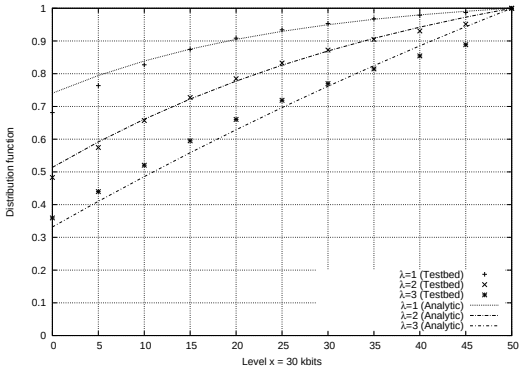


Figure 2: Stationary buffer content distribution function for different burst arrival rates

model. As far as the behaviour of the buffer is concerned, we notice that the greater the lambda, the greater the mass probability on high buffer content levels.

Let us now assume hyper-exponential distributions for the burst inter-arrival times, where coefficient of variation ranges from 4 to 100 and average arrival rate is 3 bursts per second. Figure 3 pictures the distribution function for different coefficients of variation.

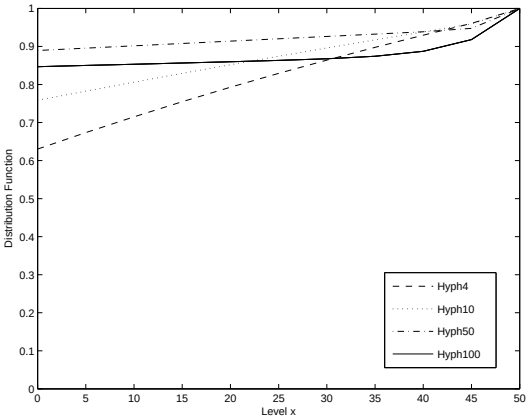


Figure 3: Stationary buffer content distribution function for different burst arrival coefficient of variation

We may clearly identify the effect of the burstiness of the processes: for very bursty arrivals, the buffer is either empty or very nearly full and spends very little time in the intermediate states. Impacts of such arrival patterns and of other more realistic ones (like on-off models) on DSCH buffer content have to be further studied.

CONCLUSION AND FUTURE WORK

In this paper, we have extended results from Dzial et al. [3] in order to model the buffer content of the DSCH of UMTS Rel'99 as a Markov driven fluid model. Performance measures, i.e. buffer content, bit loss and overflow probabilities, have been derived analytically. Assuming Poisson distributed voice and data traffics, preliminary numerical results hint at a higher buffer occupation when the traffic is heavier.

While Poisson-like traffic patterns permitted us to validate our model, these other patterns'analysis will now allow one to rapidly zoom on meaningful and interesting sets of parameter values to feed into our UMTS testbed [10]. This is the subject of future works.

Appendix

For sake of clarity, we explain the evolution of the process $\{\tilde{X}^B(t)\}$, that is driven by the phase process $\{\phi(t)\}$. In our model, the corresponding state-space $\mathcal{S}$ might be decomposed into the following three disjoint subspaces:

$$\begin{aligned}\mathcal{S}_0 &= \{(62,j,0); 1 \le j \le m\}\\ \mathcal{S}_- &= \{(i,j,0); 0 \le i < 62,\ 1 \le j \le m\}\\ \mathcal{S}_+ &= \{(i,j,k); 0 \le i \le 62,\ 1 \le j \le m,\ 1 \le k \le n\}.\end{aligned}$$

We have that $\mathcal{S}_+ = \mathcal{S}_u$, since in the framework of $\{\tilde{X}^B(t)\}$, these phases when visited make the fluid linearly increases.

The evolution of $\{\tilde{X}^B(t)\}$ is different according to observed level $\tilde{X}^B(t)$. One needs indeed to distinguish between three different types of evolution, i.e. when the buffer content is empty, partly filled or full. We explain each one in a row.

First, we assume $0 < \tilde{X}(t) < B$. Between two increasing periods, the buffer content may decrease or may remain at the same level. When the observed phase at time t is $(i,j,0)$ then the decrease rate is $R(t)$ with $K(t) = i$, see Equation (1). When the content is decreasing, we may only observe the following change of phase:

$$(i_1,j_1,0) \to (i_2,j_2,0) : I_{i_1 i_2}(D_0)_{j_1 j_2} + A_{i_1 i_2} I_{j_1 j_2},$$

where I_{ij} is the ijth element of the identity matrix I. Indeed only two types of events are possible: a change of MAP phase or a change in the current number of ongoing voice communications. When the number of ongoing voice communication is 62, the buffer content remains at the same level, that is

$$(62,j_1,0) \to (62,j_2,0) : -62\mu I_{j_1 j_2} + (D_0)_{j_1 j_2}.$$

The system switches from decreasing to increasing phase when the MAP records an arrival. The first visited phase is thus determined according to $\boldsymbol{\alpha}$, it gives:

$$(i_1,j_1,0) \to (i_2,j_2,k) : I_{i_1 i_2}(D_1)_{j_1 j_2}\boldsymbol{\alpha}_k,$$

with $1 \le i_1, i_2 \le 62$.

Upon an arrival of a burst, only the phase of the absorbing Markov process (that controls the size of a burst) may change, until reaching phase 0. Other phases evolution are frozen. This corresponds to rates of transition in the process $\{\phi(t)\}$

$$\begin{aligned}(i_1,j_1,k_1) \to (i_2,j_2,k_2) &: I_{i_1 i_2} I_{j_1 j_2} G_{k_1 k_2},\\ (i_1,j_1,k_1) \to (i_2,j_2,0) &: I_{i_1 i_2} I_{j_1 j_2} \boldsymbol{g}_{k_1},\end{aligned}$$

the latter corresponding to the system switching back to its preceding kind of evolution.

Table 1: Fluid process generators according to the fluid level

$$T = \left(\begin{array}{c|c|cc} -62\mu I_m + D_0 & e_{62}^t 62\mu \otimes I_m & D_1 \otimes \boldsymbol{\alpha} & O_{m\times 62mn} \\ \hline e_{62}\lambda \otimes I_m & I_{62} \otimes D_0 + A \otimes I_m & O_{62m\times mn} & I_{62} \otimes D_1 \otimes \boldsymbol{\alpha} \\ \hline I_m \otimes \boldsymbol{g} & O_{mn\times 62m} & I_m \otimes G & O_{mn\times 62mn} \\ O_{62mn\times m} & I_{62} \otimes I_m \otimes \boldsymbol{g} & O_{62mn\times mn} & I_{62} \otimes I_m \otimes G \end{array}\right), \quad (2)$$

$$T^0 = \left(\begin{array}{c|c|cc} -62\mu I_m + D_0 & e_{62}^t 62\mu \otimes I_m & D_1 \otimes \boldsymbol{\alpha} & O_{m\times 62mn} \\ \hline e_{62}\lambda \otimes I_m & I_{62} \otimes D_0 + A \otimes I_m & O_{62m\times mn} & I_{62} \otimes D_1 \otimes \boldsymbol{\alpha} \end{array}\right). \quad (3)$$

$$T^B = \left(\begin{array}{c|cc|c} -62\mu I_m + D_0 & D_1 \otimes \boldsymbol{\alpha} & O_{m\times 62mn} & e_{62}^t 62\mu I_m \\ \hline I_m \otimes \boldsymbol{g} & I_m \otimes G & O_{mn\times 62mn} & O_{mn\times 62m} \\ O_{62mn\times m} & O_{62mn\times mn} & I_{62} \otimes I_m \otimes G & I_{62} \otimes I_m \otimes \boldsymbol{g} \end{array}\right). \quad (4)$$

Only when the maximum number of DCH channels is reached, the DSCH buffer content will remain at the same level. This happens with a rate of probability:

$$(i_1, j_1, 0) \to (62, j_2, 0) : (e_{62})_{i_1} \lambda I_{j_1 j_2},$$

where e_{62} is a column vector of size 62 full of 0's except for a 1 at entry 62. When a voice communication ends up, then the content process rate of evolution decreases back. This happens when:

$$(62, j_1, 0) \to (i_2, j_2, k) : 62(\mu e_{62}^t)_{i_2} I_{j_1 j_2}.$$

If a burst arrives in such a period, then we observe

$$(62, j_1, 0) \to (62, j_2, k) : (D_1)_{j_1 j_2} \boldsymbol{\alpha}_k.$$

Let us consider the previously proposed partition of the $\mathcal{S}$, that is $\mathcal{S}_0 \cup \mathcal{S}_- \cup \mathcal{S}_+$. Let us partition the generator T of the phase process $\{\phi(t)\}$ in a manner conformant to that of $\mathcal{S}$, we thus obtain Generator (2) explicitly given in Table 1. where $O_{m\times n}$ is a matrix full of zeros of size $m \times n$, and I_m is the identity matrix of size m.

In the same manner, one can establish that the evolution of the phase process when the fluid content is at level 0; is the generator T^0 as described in (3). When the fluid touches level B and so only phases belonging to $\mathcal{S}_0$ and $\mathcal{S}_+$ make the fluid staying at that level, this gives Generator (4). It is worth noting that when the fluid stays at level B with a phase belonging to $\mathcal{S}_0$, we still record the next burst arrival. One may object that this does not reflect reality. Fortunately, to obtain the stationary density of $X^B(t)$, a time transform is used so that those added virtual times have no effect at all as we prove in the analytical study. Furthermore, we observe that in this particular case study, matrices T, T^0 and T^B have the same content; for corresponding indexes.

References

[1] R. Bestak, P. Goldlewski, and P. Martins. RLC Buffer Occupancy When Using a TCP Connection Over UMTS. In *Proceedings of 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC 2002*, pages 161–165, Lisbon, Portugal, September 2002.

[2] A. da Silva Soares and G. Latouche. Matrix-analytic methods for fluid queues with finite buffers. *Perform. Eval.*, 63:295–314, 2006.

[3] T. Dzial, L. Breuer, A. da Silva Soares, G. Latouche, and M.-A. Remiche. Fluid queues to solve jump processes. *Perform. Eval.*, 62(1-4):132–146, 2005.

[4] H. Holma and A. Toskala. *WCDMA for UMTS - Radio Access for Third Generation Mobile Communications - Third Edition*. John Wiley & Sons, Ltd., 2004.

[5] J. Perez-Romero, O. Sallent and R. Agusti. *On Dimensioning UTRA-FDD Downlink Shared Channel*. In *Proceedings of 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC 2004*, pages 1777–1781, Barcelona, Spain, September 2004.

[6] G. Latouche and T. Takine. Markov renewal fluid queues. *J. Appl. Probab.*, 41:746–757, 2004.

[7] G. Lin, T. Suda, and F. Ishizaki. Loss Probability for a Finite Buffer Multiplexer with the M/G/ ∞ Input Process. *Telecommunication Systems*, 29(3):181–197, 2005.

[8] V. Ramaswami. Matrix analytic methods for stochastic fluid flows. In D. Smith and P. Hey, editors, *Teletraffic Engineering in a Competitive World (Proceedings of the 16th International Teletraffic Congress)*, pages 1019–1030, Edinburgh, UK, 1999. Elsevier Science B.V.

[9] H. Sakasegawa, M. Miazawa, and G. Yamazaki. Evaluating the Overflow Probability Using the Infinite Queue. *Management Science*, 39(10):1238–1245, 1993.

[10] H. Van Peteghem and L. Schumacher. Implementation of an Open-Source UTRAN Testbed. In *Proceedings of the 2nd International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TridentCom)*, Barcelone, Spain, 2006.

EXPANDED HIDDEN MARKOV MODELS: ALLOWING SYMBOL EMISSIONS AT STATE CHANGES

Claudia Krull, Graham Horton
Institut für Simulation und Graphik
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2, 39016 Magdeburg, Germany
E-mail: {claudia,graham}@sim-md.de

KEYWORDS

Hidden Markov Models, Stochastic Petri Nets.

ABSTRACT

In this paper we formally expand hidden Markov models (HMM) by symbol emissions at state changes. These expanded hidden Markov models (eHMM) can contain more information than original HMM with the same number of states. This is a necessary step towards the definition of hidden non-Markovian models on the basis of discrete stochastic models. These are most of the time event driven, which makes it necessary to attach information to the state changes that represent the events. The paper shows that the extended paradigm is to some extent equivalent to original HMM, and gives an example of the new possibilities using hidden non-Markovian models.

MOTIVATION

Simulation most of the time involves the need to model a real or planned system and then predict its behavior to some extent. This process heavily depends on structural and statistical knowledge about the system's behavior and parameters. For various reasons, sometimes one is not able or not willing to observe the complete system internals, but rather focuses on the system's visible interaction with the environment. This could be the case, when one wants to diagnose the engine of a car without taking it apart, but by solely looking at some protocol that can be read out electronically. This error protocol can also be viewed as a sequence of signals that the hidden system (engine internals) emits. In case of a satellite in orbit it might not even be feasible to examine the object itself, but one has to rely on the signals emitted by the satellite in order to diagnose it.

Many real systems can be represented using discrete stochastic models such as Petri nets. They are event-driven; the state transitions are the active elements and they model the dynamic elements of the real process. So-called impulse rewards can be associated with state transitions, and can model costs, protocol signals and much more. However, the analysis methods for Petri nets do not account for hidden information and the system is always assumed to be observable as a whole.

Hidden Markov models (HMM) on the other hand can model hidden processes with observable outputs. HMM consider the symbol emitted in a certain state, regardless of the previous or following state. The hidden model of an HMM is a discrete-time Markov chain (DTMC). This restricts the modeling capabilities of the paradigm to geometric state duration distributions in the discrete case. Combining the advantages of HMM and Petri nets, it would be possible to build more general hidden models which are event driven. Hidden non-Markovian models that can model more realistic behavior than DTMCs could be analyzed with the methods of HMM, which expands the range of possible application areas.

The first step is to make HMM event-driven. This is done by associating symbol emissions with transitions, rather than with states. The second step is the formalization of hidden non-Markovian models, which will not be described here. In this paper we formally define expanded hidden Markov models (eHMM) by associating symbol emissions with the state changes of HMM. We also adapt the three major algorithms that are used for the different analysis procedures, namely the forward algorithm for Evaluation, Viterbi for Decoding and Baum-Welch for Training of eHMM. The paper also gives an example of a hidden non-Markovian model to show the possible applications of the new paradigm.

STATE OF THE ART

Hidden Markov models and their application in speech recognition were first published around 1970; one of the first papers is (Baum et al. 1970). A summary of the theory including algorithms and application examples can be found in (Rabiner 1989), which is also the basis for the notation used in the current paper. In order to make the models more flexible, research was done on explicit state duration densities (hidden semi-Markov models), complicating the solution algorithms (Russel and Moore 1985). Expanding all HMM states to a sub-HMM was used to realize more general state duration distributions (expanded state HMM) (Russel and Cook 1987). This increased the number of free parameters and the sub-HMM topologies were not very flexible. The performance of both approaches was tuned to speech-recognition systems. In (Wickborn et al. 2006) a method is described to find the trace probability and state sequence for stochastic models with generally distributed transitions, but the method is not applicable for the training of models. The implicit assumption of that paper, which associated symbols with state

changes, is formalized in the current paper. In (Isensee et al. 2006) a method for the training of hidden non-Markovian models using phase-type distributions was introduced.

Hidden Markov Model Background

Classical HMM are discrete-time Markov chains (DTMC) that stochastically emit a symbol at every time step, depending on the state that the hidden process currently resides in. These so-called doubly stochastic processes are also known as signal models. They are widely used in speech and pattern recognition. The stochastically emitted symbols can also be interpreted as only partially observed model states, which leaves room for the analysis of incomplete observations.

A hidden Markov model is described by a 5-tuple (S,V,A,B,Π). The elements are: the set of states (S), the transition probability matrix (A) and the initial probability vector of the DTMC (Π), the output probability matrix (B) and the set of output symbols (V), where b_{ij} is the probability to output symbol v_j in state s_i. $Q=\{q_1,q_2,q_3,\ldots, q_T\}$ describes a sequence of DTMC states and $O=\{o_1,o_2,o_3,\ldots,o_T\}$ describes an observed output sequence with T being the number of time steps. A specific HMM is often denoted by $\lambda=(A,B,\Pi)$ which fully defines the model. The three basic (analysis) tasks that can be performed for an HMM and a given output sequence (trace) are:

1. Find the probability of producing the given trace O with the given model λ.
2. Find the most probable state sequence Q of the model λ that produced the observed trace O.
3. Maximize the probability with which the model λ produces the trace O by training the model.

The first task can be solved by the forward algorithm (Rabiner 1989). The most likely state sequence can be determined by the Viterbi algorithm (Viterbi 1967). The third task of training an HMM is solved by the so-called Baum-Welch algorithm (Baum et al. 1970).

eHMM – DEFINITION AND NOTATION

First the basic definition of eHMM is adapted from HMM. Most of the components stay the same. The changes are that the state sequence is extended by one state and starts at time $t=0$. This makes it also one element longer than the symbol trace, since n+1 states are needed for n transitions with symbol outputs. The most important change can be found in the emission probabilities B. These are no longer attached to a specific state, but to a transition between two states.

$$5-Tuple: (S,V,A,B,\Pi)$$
$$Model: \lambda = (A,B,\Pi)$$
$$States: S = \{s_1,\ldots s_N\}$$
$$Symbols: V = \{v_1,\ldots v_M\}$$
$$Symbol\ Sequence: O = \{o_1\ldots o_T\}$$
$$State\ Sequence: Q = \{q_0\ldots q_T\}$$

Transition Matrix:

$$A = \{a_{ij}\}_{N\times N}$$
$$a_{ij} = P(q_t = s_j \mid q_{t-1} = s_i\}$$

Output Probabilities:

$$B(k) = \{b_{ij}(k)\}_{N\times N} \quad k = 1\ldots M$$
$$b_{ij}(k) = P(v_k\ at\ t \mid q_{t-1} = s_i \wedge q_t = s_j)$$

Initial Probability Vector:

$$\Pi = \{\pi_i\}_N$$
$$\pi_i = P(q_0 = s_i)$$

The emission probabilities B can no longer be stored in one matrix, but in a set of matrices, one for each symbol, that contains the probability that the symbol is emitted at a specific state change. The sum of the emission probabilities for each state change has to be 1. Therefore the following condition holds:

$$\sum_{k=1}^{M} b_{ij}(k) = 1\ \forall i,j,a_{ij} > 0$$

eHMM - THE THREE ANALYSIS ALGORITHMS

The three basic problems that can be solved using HMM are Evaluation, Decoding and Training. Each of them has different practical applications. These will now be adapted to eHMM.

Problem I : Evaluation

The evaluation problem takes a given model $\lambda=(A,B,\Pi)$ and a given output sequence O. The question is the following: What is the probability that the given output sequence was produced by the given model?

$$O,\lambda \Rightarrow P(O \mid \lambda)$$

Forward Algorithm

The forward algorithm is an iterative algorithm, which computes the probabilities α of the growing sub-sequences step by step. It has a complexity of $O(N*T)$ as opposed to the brute force approach $O(N^T)$ that evaluates all possible paths separately.

$$\alpha_t(i) = P(o_1 o_2 \ldots o_t, q_t = s_i \mid \lambda)$$

Initialization:

$$\alpha_0(i) = \pi_i$$

Induction:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_{ij}(o_t) \quad t = 1\ldots T$$

Termination:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

The forward algorithm can be easily modified to allow for symbol emissions at state changes. The value of $\alpha_t(i)$ now describes the probability of having emitted the symbol sequence $o_1\ldots o_t$ and ending up in state s_i.

Backward Algorithm

The backward algorithm works analogously to the forward algorithm, but starts to evaluate the trace at the end. It is mentioned here for completeness and because the β values are needed later on for the Baum-Welch algorithm.

$$\beta_t(i) = P(o_{t+1} \ldots o_T \mid q_t = s_i, \lambda)$$

Initialization:

$$\beta_T(i) = 1$$

Induction:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_{ij}(o_{t+1}) \beta_{t+1}(j) \quad t = T-1 \ldots 0$$

Termination:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \beta_0(i) \pi_i$$

The backward algorithm can also be easily modified to allow for symbol emissions at state changes. The value of $\beta_t(i)$ now describes the probability of having emitted the trace $o_{t+1} \ldots o_T$ when starting in state s_i.

Problem II : Decoding

The decoding problem takes a given model $\lambda=(A,B,\Pi)$ and a given output sequence O. One then tries to retrace/reconstruct the behavior of the hidden model that most likely produced the given output according to some measure.

$$O, \lambda \Rightarrow \arg\max_{Q} P(O \mid Q, \lambda)$$

Individual most probable states

Using the original HMM definition, one way of decoding a given trace is to determine the individually most probable state to produce a certain output symbol in the sequence and then take that as the solution. The problem is that the resulting state sequence might not be valid, since transitions between two successive states might not be possible if the transition probability is 0.

This approach does not work here, since two successive most probable state changes cannot be as easily connected as two successive states. The target state of the first transition and the source state of the second transition have to be the same to produce a valid state sequence. This is even more serious than having zero probability to change between two successive states.

Viterbi Algorithm

The Viterbi algorithm decodes a given symbol sequence by determining the most probable sequence of states to have produced it. The algorithm has a similar structure to the forward algorithm.

$$\delta_t(i) = \max_{q_0 \cdots q_{t-1}} P(q_0 \ldots q_t = s_i, o_1 \ldots o_t \mid \lambda)$$

$$\delta_{t+1}(j) = \max_i \left[\delta_t(i) a_{ij} b_{ij}(o_{t+1}) \right]$$

Initialization:

$$\delta_0(i) = \pi_i$$

$$\psi_0(i) = 0$$

Recursion:

$$\delta_t(j) = \max_i \left[\delta_{t-1}(i) a_{ij} b_{ij}(o_t) \right] \quad t = 1 \ldots T$$

$$\psi_t(j) = \arg\max_i \left[\delta_{t-1}(i) a_{ij} b_{ij}(o_t) \right] \quad t = 1 \ldots T$$

Termination:

$$P^* = \max_{i=1 \ldots N} \left[\delta_T(i) \right]$$

$$q_T^* = \arg\max_{i=1 \ldots N} \left[\delta_T(i) \right]$$

Backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*)$$

The Viterbi algorithm can also be easily modified to allow for symbol emissions at state changes. The value of $\delta_t(i)$ now contains the probability of the most likely path that emitted the symbol sequence $o_1 \ldots o_t$ and ends up in state s_i. The most likely state sequence so far for state s_i, which is needed for backtracking, is stored in $\psi_t(i)$. The resulting most likely state sequence to have produced the output O is called the Viterbi path.

Problem III : Training

The problem of training HMMs is by far the most difficult, and is more an optimization task than a direct solution. The training problem takes a given output sequence O and a model size given by the model states S and the output symbols V. The task is to find the most likely model to have produced this output sequence.

$$O \Rightarrow \lambda = \arg\max_{\lambda} P(O \mid \lambda)$$

Baum-Welch Algorithm

The Baum-Welch algorithm is an optimization algorithm that takes an initial model specification and improves it successively through several iterations called epochs, ensuring that the probability of emitting the output sequence is increased in every iteration.

The $\alpha_t(i)$ and $\beta_t(i)$ are the terms calculated by the forward and backward algorithm. $\gamma_t(i)$ is the probability of being in state s_i at time t given the current model configuration and output sequence. The sum of the $\gamma_t(i)$ over the length of the trace is the expected number of transitions from state s_i given the current model and trace. $\xi_t(i,j)$ is the probability of the state change from s_i to s_j emitting the symbol o_t from time t-1 to t. This is only defined if $a_{ij}>0$. The sum of the $\xi_t(i,j)$ over the length of the trace is the expected number of state changes from s_i to s_j given the current model and trace.

$$\xi_t(i,j) = P(q_{t-1} = s_i, q_i = s_j \mid O, \lambda)$$

$$= \frac{\alpha_{t-1}(i) a_{ij} b_{ij}(o_t) \beta_t(j)}{P(O \mid \lambda)}$$

$$= \frac{\alpha_{t-1}(i) a_{ij} b_{ij}(o_t) \beta_t(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{t-1}(i) a_{ij} b_{ij}(o_t) \beta_t(j)}$$

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \quad t = 0 \ldots T-1$$

$$\sum_{t=0}^{T-1} \gamma_t(i) = \textit{expected number of transitions from } s_i$$

$$\sum_{t=1}^{T} \xi_t(i,j) = \textit{expected number of transitions from } s_i \textit{ to } s_j$$

These values are now used to re-estimate the model parameters $\lambda=(A,B,\Pi)$ as follows. The re-estimation step needs to be performed until a predefined stability criterion is satisfied.

$$\textit{Re-estimation Formulas}$$

$$\overline{\pi_i} = \textit{expected frequency in state } s_i \textit{ at time } t = 0$$
$$= \gamma_0(i)$$

$$\overline{a_{ij}} = \frac{\textit{expected number of transitions from } s_i \textit{ to } s_j}{\textit{expected number of transitions from } s_i}$$
$$= \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=0}^{T-1} \gamma_t(i)}$$

$$\overline{b_{ij}(k)} = \frac{\textit{expected number of transitions from } s_i \textit{ to } s_j \textit{ observing } v_k}{\textit{expected number of transitions from } s_i \textit{ to } s_j}$$
$$= \frac{\sum_{t=1 \& o_t = v_k}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \xi_t(i,j)}$$

The re-estimation formulas can be transformed to fit the new model definition. Thus the expanded HMM can also be trained using the Baum-Welch algorithm. The resulting procedure should still suffer from the same drawbacks as the original: It is still a local optimization method, which moves in the direction of steepest descent and is very expensive, having to run through the forward and backward procedures for every epoch. The training of an eHMM will most likely require more effort (data or time) than an HMM, since the model has an additional dimension and therefore more parameters.

COMPARING HMM AND eHMM

In this section we will show that a standard HMM can be transformed into an expanded HMM, which makes the latter at least as powerful as HMM. The example system is a small web server that cannot be observed directly. An internet user is trying to deduce the server's current condition based on its answers to ping requests. The HMM models the web server that can be either in the states *Idle, Busy* or *Failed* (see Figure 1). To the user the actual state of the web server is not visible, he can only observe the answers once every time unit, either a *ping* reply or a *timeout*. By analyzing the observed traces, he can draw inferences about the actual state of the web server. The parameters of the model are the initial probability vector Π, the transition probability matrix *A* and the output probability matrix *B*, which can all be derived from the Figure.

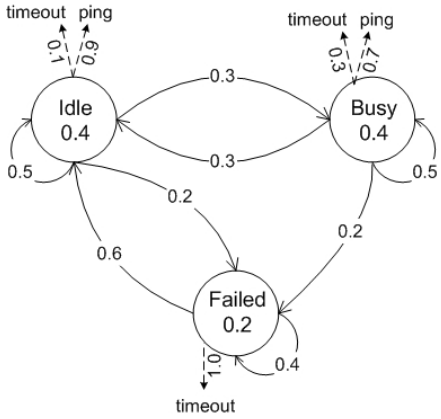


Figure 1 : Original HMM with Parameters

This model $\lambda=(A,B,\Pi)$ can easily be transformed into an eHMM $\lambda'=(A',B',\Pi')$:

- The transition probabilities stay the same $A' := A$
- The output probabilities shift from the states to the state changes, by using the output probabilities of the target state for a state transition $b_{ij}'(k) := b_j(k)$
- The initial probability vector has to be computed by solving the following linear system of equations (assuming it to be a row vector) $\Pi = \Pi' A$.

The new calculation of the initial probability vector becomes necessary, because the state sequence of the eHMM has one more element than the output sequence or the original state sequence of the HMM.

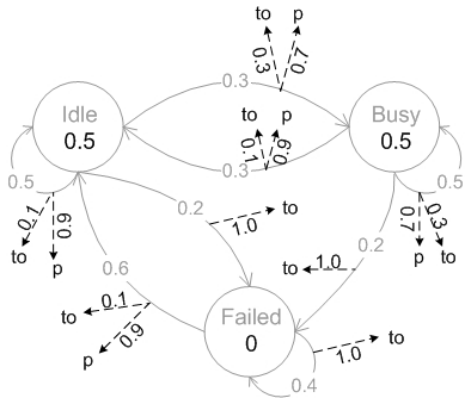


Figure 2 : Equivalent Expanded HMM with Parameters

These transformations lead to the model shown in Figure 2. Changed elements are highlighted. The major change is the shifting of the output probabilities from the states to the state transitions. The two example models are equivalent in the sense that they produce the same output sequences with the same probabilities. The output probabilities of two sequences were computed using the original and the modified forward algorithm. The Viterbi algorithm was used to compute the most likely state sequences that produced the output.

The internet user has observed two sequences: (*ping, ping, ping*) and (*ping, timeout, ping*). He wants to know the absolute probability of each trace and the most likely hidden state sequences that produced them. The

results of the forward and Viterbi algorithm for the given models are shown in Table 1. $P(O|\lambda)$ is the absolute probability of emitting trace O with the original HMM; $P(O|\lambda')$ is the probability of emitting trace O with the equivalent eHMM. The probabilities are exactly the same in both cases. Q is the Viterbi path of the HMM, and Q' the Viterbi path of the eHMM. F denotes the hidden model state *Failed* and I the state *Idle*. The last three elements of the paths are equivalent.

Table 1 : Results of HMM and eHMM

Sequence *O*	***(ping,ping,ping)***	***(ping,timeout,ping)***
HMM Probability	$P(O\|\lambda)$=0.26448	$P(O\|\lambda)$=0.132672
eHMM Probability	$P(O\|\lambda')$=0.26448	$P(O\|\lambda')$=0.132672
HMM Viterbi path	Q=I,I,I	Q=I,F,I
eHMM Viterbi path	Q'=I,I,I,I	Q'=I,I,F,I

The equivalence of the output trace probabilities can be proven inductively, as the following formulas show.

Basis

$$\alpha_1(j) = \pi_j b_j(o_1) \quad (\pi_j = \sum_{i=1}^{N} \pi'_i a_{ij} \text{ by construction})$$

$$= \sum_{i=1}^{N} \pi'_i a'_{ij} b_j(o_1) \quad (b_j(o_1) = b'_{ij}(o_1) \text{ by construction})$$

$$= \sum_{i=1}^{N} \alpha'_0(i) a'_{ij} \; b'_{ij}(o_1)$$

$$= \alpha'_1(j) \quad \forall j$$

Assumption

$$\alpha_k(i) = \alpha'_k(i) \quad \forall i$$

Inductive Step

$$\alpha_{k+1}(j) = \sum_{i=1}^{N} \alpha_k(i) a_{ij} b_j(o_{k+1}) \quad (b_j(o_t) = b'_{ij}(o_t) \text{ by construction})$$

$$= \sum_{i=1}^{N} \alpha'_k(i) a'_{ij} \, b'_{ij}(o_{k+1})$$

$$= \alpha'_{k+1}(j)$$

Conclusion

$$\alpha_t(i) = \alpha'_t(i) \quad \forall i, t = 1 \ldots T$$

The terms calculated by the forward algorithm are the same for the HMM as for the equivalent eHMM. All terms of the eHMM are marked by an apostrophe. The Basis shows that the terms $\alpha_1(i)$ and $\alpha'_1(i)$ are equivalent, by replacing the single terms by their equivalents, based on the defined transformation rules. Based on the assumption that the terms for step k are equal, it is concluded that the terms for k+1 must be equal. Therefore this equality holds for all $t>0$.

The proof of equivalence of the Viterbi path can be done analogously to some extent. The assumption is, that the last T elements of the eHMM Viterbi path are the same as the HMM Viterbi path. The idea is that if the first element of the HMM Viterbi path is the same as the second element of the eHMM Viterbi path, then the following elements will also be equivalent. The problem is, that the base assumption is not given in all cases, since the two-element eHMM Viterbi path might not contain the most likely starting state $max(\pi_i)$ of the HMM, since the π_i are the weighted sum rather then the maximum of the π'_i. However, in the given example, the Viterbi paths of the two models are equivalent.

This shows that the example transformation was done correctly and the algorithms produce results that are equivalent. The equivalence was proven for the forward algorithm, and was tested on an example for the Viterbi algorithm. We did not test the Baum-Welch algorithm for several reasons. First of all, the equivalence of the results might not be given, since a different model is trained, with more free parameters, and secondly implementing the adapted algorithm is still future work.

The reverse transformation of an arbitrary eHMM into an HMM is not as intuitive as the approach described in this section. Extra states have to be introduced to account for the increased amount of information in an eHMM. Artificial constructs might become necessary such as states with no probability to stay in. The dual graph that could be constructed using graph theory, is only defined on planar graphs, which does not have to be the case for Markov chains.

MOTIVATING EXAMPLE

An example of an interesting hidden non-Markovian model is given in this section. It will show the increased modeling power of the new paradigm, which involves symbol emissions at state changes. The example is that of a machine in a factory, which is essential for the availability of the whole production line and is often maintained to prevent failures. Even though, the machine fails from time to time, and can therefore be in the three state *OK*, *Failed* or *Maint.* The times between failures (T_F) are Weibull-distributed and the maintenance interval (T_{MI}) is Normally-distributed. The maintenance of a machine can produce different costs, depending on the extent of repair necessary (10€, 20€ or 30€), each having a different probability to occur. Similarly the costs for the repair of a failed machine are dependent on the severity of the disruption (30€, 100€ or 200€). The time needed for repair or maintenance have Lognormal distributions. The production manager only gets the record of the bookings that were the results of the repairs or maintenances, but not the exact events that produced them.

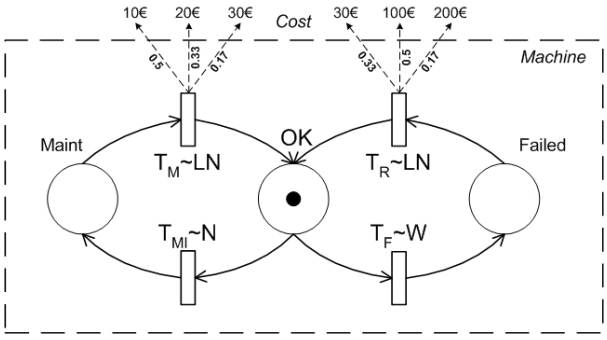


Figure 3 : Example of Hidden non-Markovian Model

This system could be interpreted as a hidden non-Markovian model (see Figure 3), with the machine

being the internal hidden part and the bookings as observable symbol emissions. It would not be possible to model this using an HMM, since the underlying model is not a DTMC.

The initial probability vector of the model is $\Pi=(0,1,0)$ (states $S=\{Maint,OK,Failed\}$). The transition probabilities are no longer static, but depend on the ages of the transitions, the time they have been enabled without firing. Assuming μ is the instantaneous transition rate (hazard rate) of the probability distribution, τ the current age of the transition and Δ the discretization time step, then the transition probabilities calculate as follows.

$$A(\vec{\tau}) := \begin{bmatrix} 1-p_M & p_M = \Delta * \mu_M(\tau_M) & 0 \\ p_{MI} = \Delta * \mu_{MI}(\tau_{MI}) & 1-p_{MI}-p_F & p_F = \Delta * \mu_F(\tau_F) \\ 0 & p_R = \Delta * \mu_R(\tau_R) & 1-p_R \end{bmatrix}$$

The output probabilities have to be specified for every output symbol in $V=\{0,10,20,30,100,200\}$. Two example output probability matrices are the following:

$$B(0) = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \quad B(30) = \begin{bmatrix} 0 & 0.17 & 0 \\ 0 & 0 & 0 \\ 0 & 0.33 & 0 \end{bmatrix}$$

Several new and interesting questions could be answered using this abstraction:

- How probable is a certain booking record? Which series of events produced that record most likely?
- What is the machine availability of the possible generating paths, and how does that compare to the actual observed availability of the machine?
- How would the behavior of the system change, when changing maintenance intervals?
- Are there any frequent failure sequences? What produced them? Could they be prevented by using more reliable system components?

These questions can be answered using hidden non-Markovian models and the analysis methods of both HMM and Petri nets. Numerical results for the example cannot be given yet, since the implementation of the algorithms is future work. The new combined paradigm will enable the direct utilization of system output for the parameterization, diagnosis and analysis of not directly observable behaviour of Petri nets. This is not possible by using either HMM or Petri nets alone.

CONCLUSION

In this paper we expanded the definition of hidden Markov models to symbol emissions at state changes. We first expanded the basic definition of HMM and adapted the three basic analysis algorithms. We then gave examples of the equivalence to the old notation and of the usefulness of hidden non-Markovian models.

This new definition shifts the focus of the model from the states to the state changes: the events. This is necessary to be able to analyze discrete stochastic systems using the proposed methods, since there the events are the driving force of the model behavior. Using this newly defined paradigm of eHMM we can now proceed to the formalization of hidden non-Markovian models. These will open up a whole new range of possible applications of the analysis methods of HMM. These applications might include the analysis of systems by only looking at failure protocols and incomplete observations or the prediction of machine performance based on the previous machine behavior.

Future work includes the implementation of the algorithms and concepts described in this paper and those relating to HnMM. A special focus has to be on the performance of the algorithms, since especially the training of HMM is time consuming. Another interesting question is how the adapted original algorithms perform compared to the algorithms involving Proxels and discrete phases. (Isensee et al. 2006, Wickborn et al. 2006)

REFERENCES

Baum, L.E., T. Petrie, G. Soules, N. Weiss. 1970. “A Maximization Technique in the Statistical Analysis of Probabilistic Functions of Markov Chains”. In *The Annals of Mathematical Statistics*, vol.41, no.1, pp.164-171.

Isensee, C., F. Wickborn and G. Horton. 2006. “Training Hidden non-Markov Models”. In *Proc. of the ASMTA’06,* pp. 105-110.

Rabiner, L. R.. 1989. “A tutorial on hidden Markov models and selected applications in speech recognition”. In *Proceedings of the IEEE.* vol.77, no.2, pp.257–286.

Russel, M. J. and R. K. Moore. 1985. “Explicit modelling of state occupancy in hidden Markov models for automatic speech recognition”. In *Proc. ICASSP’85*, pp. 5-8.

Russell, M. J. and A. E. Cook. 1987. “Experimental evaluation of duration modelling techniques for automatic speech recognition,” in *Proc. ICASSP’87*, pp. 2376–2379.

Viterbi, A. H.. 1967. “Error bounds for convolutional codes and an asymptotically optimal decoding algorithm”. In *IEEE Transactions on Information Theory*, IT-13:260–269.

Wickborn, F., C. Isensee, T. Simon, S. Lazarova-Molnar, G. Horton. 2006. “A New Approach for Computing Conditional Probabilities of General Stochastic Processes”. In *Proc. ANSS’06,* pp. 152-159.

AUTHOR BIOGRAPHIES

CLAUDIA KRULL studied Computer Science at the Otto-von-Guericke-Universität Magdeburg and spent an exchange year at the University of Wisconsin, Stevens Point, where she graduated in 2002. She was awarded her German University diploma in September 2003. Since October 2003 she is a PhD student at the “Lehrstuhl für Simulation” at the Otto-von-Guericke-Universität Magdeburg. Her e-mail address is: `claudia@sim-md.de.`

GRAHAM HORTON studied Computer Science at the University of Erlangen, obtaining his Masters degree ("Diplom") in 1989. He obtained his PhD in Computer Science in 1991 and his "Habilitation" in 1998 at the same university, in the field of simulation. Since 2002, he is Professor for Simulation and Modelling at the Computer Science department of the University of Magdeburg. His email address is: `graham@sim-md.de.`

Author Index