HYBRID SIMULATION OF ACTIVE TRAFFIC MANAGEMENT

Anna V. Korolkova, Tatyana R. Velieva, Pavel O. Abaev
Department of Applied Probability and Informatics
Peoples' Friendship University of Russia
Miklukho-Maklaya str. 6, Moscow, 117198, Russia
Email: akorolkova@sci.pfu.edu.ru, trvelieva@gmail.com, pabaev@sci.pfu.edu.ru
Leonid A. Sevastianov
Department of Applied Probability and Informatics
Peoples' Friendship University of Russia
Miklukho-Maklaya str. 6, Moscow, 117198, Russia
and Bogoliubov Laboratory of Theoretical Physics
Joint Institute for Nuclear Research
Joliot-Curie 6, Dubna, Moscow region, 141980, Russia
Email: leonid.sevast@gmail.com
Dmitry S. Kulyabov
Department of Applied Probability and Informatics
Peoples' Friendship University of Russia
Miklukho-Maklaya str. 6, Moscow, 117198, Russia
and Laboratory of Information Technologies
Joint Institute for Nuclear Research
Joliot-Curie 6, Dubna, Moscow region, 141980, Russia
Email: yamadharma@gmail.com

KEYWORDS

Hybrid modeling, fluid model, active queue management, random early detection, Modelica

ABSTRACT

For the study and verification of our mathematical model of RED-like active traffic management module a discrete simulation model and a continuous analytical model were developed. However, for various reasons, these implementations are not entirely satisfactory. It is necessary to develop a more adequate simulation model, possibly using a different modeling paradigm. In order to modeling of the TCP source, the RED control module, and the process of their interaction it is proposed to use a hybrid (continuous-discrete) approach. For computer implementation of the model the physical modeling language Modelica is used. Because the language Modelica has multiple implementations we have selected the OpenModelica compiler. The hybrid approach allows us to take into account the transitions between different states in the continuous model of the TCP protocol. The hybrid approach simplified the consideration of the model due to the conversion of a differential inclusions into a set of differential equations with discrete transitions. The considered approach allowed to obtain a simple simulation model of interaction between RED module and TCP source. This model has great potential for expansion. It is possible to implement different types of TCP and RED. Furthermore, it is possible to use a hybrid approach not only for the simulation but also for analytical modeling.

INTRODUCTION

While complex systems modeling there is the problem of choosing a model approach.When using a discrete-continuous dichotomy, we always have some inappropriate elements. These elements of the model do not well correspond to the selected approach. In particular, the existing models of control systems can not fully meet our needs.

As the implementation of the system with a threshold control we investigated the RED (seeFloyd and Jacobson (1993); Feng et al. (2015); Lautenschlaeger and Francini (2015); Karmeshu et al. (2016)) active traffic management unit for the TCP protocol. During simulation of TCP protocol and RED mechanism the serious problems arised. As it turned out adequate TCP models are simply missing. Not even a common method for its modeling (see Paxson and Floyd (1997, 1995); Leland et al. (1994)) exists.

For example, overload control can be analyzed using the queuing systems theory (see Abaev et al. (2014); Gaidamaka et al. (2014)).

For modeling we used the continuous (fluid) model for TCP and RED (see Demidova et al. (2014); Eferina et al. (2014)). However, this approach allowed us to model the TCP protocol

Proceedings 30th European Conference on Modelling and Simulation ©ECMS Thorsten Claus, Frank Herrmann, Michael Manitz, Oliver Rose (Editors)
ISBN: 978-0-9932440-2-5 / ISBN: 978-0-9932440-3-2 (CD)

only partially. In addition, we received differential inclusions instead of differential equations.

We used software package ns-2 for the resulting model verification. Discrete event simulator ns-2 (see Altman and Jiménez (2012); Issariyakul and Hossain (2012)) implements some network protocols. Because of this, it has a low scalability and is suitable for modeling of small networks at small time intervals.

Also the verification on the basis of the software router (see Velieva et al. (2014)) was carried out.

For further development of our model, it was decided to use a hybrid approach (see Maler (1992, 2002); Färnqvist et al. (2002); Hespanha et al. (2001); Bohacek and Lee (2001)). This article discusses a general approach to a hybrid modeling of TCP and RED. The implementation is demonstrated on the basis of Modelica (see Fritzson (2003, 2011)) language with the help of OpenModelica system.

The structure of the article is as follows. The first section describes the hybrid paradigm of mathematical modeling. The general information about the language of physical modeling named Modelica is given in the same section. Modelica implements continuous and hybrid paradigms. The TCP Reno protocol as also continuous and hybrid models of this protocol are presented in the second section. Similarly, in the third section this is done for the RED active traffic management module. The last section describes the network topology and the overall structure of the hybrid model for its implementation.

THE HYBRID APPROACH TO MODELING

The hybrid[1] (see Maler (1992, 2002); Färnqvist et al. (2002); Hespanha et al. (2001); Bohacek and Lee (2001)) system has both continuous and discrete aspects of behavior. The hybrid behavior may be due to different reasons.

- Hybrid behavior is due to the joint operation of the continuous and discrete objects. For example, the automatic control system with continuous control object and discrete control device.
- Hybrid behavior is caused by changes in the structure of the system. A system with variable number of components may be considered as an example.
- Hybrid behavior may be caused by instant qualitative changes in a continuous object. In this case, qualitative changes during the simulation of continuous systems are presented as discrete events. Asesult, the hybridism is not an inherent characteristic of the system, but the modeling technique.

Hybrid systems may be considered as discrete-continuous or continuous-discrete systems.

- The waiting time for the next input and the duration of the output action can be taken into account in discrete systems.
- The coexistence of instant and long-term processes in a continuous-time model.

We will add discrete elements to the initially developed continuous dynamic model.

Discrete events in continuous dynamic models can be created by the following components:

- initial conditions and parameter values in the right sides;
- the form of the right sides;
- the number of equations.

The change of the initial conditions and the stepwise change of the parameters may be considered as the same type changes because the stepwise change of parameters can be described as a replacement of the initial conditions for a new system of equations.

Thus, one can use both indicator functions and differential inclusions within the hybrid model. This technique allows to replace the system with a variable right side by the system with the constant right side and the variable initial conditions.

For example, let's suppose a system of differential equations with a piecewise constant parameter π:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x,t,\pi),$$
$$\pi = \begin{cases} \psi_1, & x \in \mathfrak{X}_1, \\ \psi_2, & x \in \mathfrak{X}_2. \end{cases}$$

Then it can be replaced by the following system:

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t} = f(x,t,\pi), \\ \dfrac{\mathrm{d}\pi}{\mathrm{d}t} = 0, \end{cases}$$

with the following initial conditions:

$$\begin{cases} \pi(0) = \psi_1, & x \in \mathfrak{X}_1, \\ \pi(0) = \psi_2, & x \in \mathfrak{X}_2. \end{cases}$$

This technique may be applied in modeling of the behavior of the TCP protocol and RED mechanism in different states.

Modelica modeling language

Modelica language (see Fritzson (2003, 2011)) is developed by a nonprofit organization Modelica, which also develops a free library for this language. Modelica is positioned as an object-oriented physical modeling language.

The classes are the basis of modeling in Modelica. Modelica class includes not only the fields and methods, but also the equations which relate variables to each other. The equations are a special entity in the language. The number of equations and variables in the program must be the same. Also, the fields may have a different type of variability: a constant parameter (does not change under the current modeling), a variable. The field in the class can be both the object of embedded types and the object of user-defined types. The classes can be inherited, the entire contents of inherited class is copied to the inheriting class, including equations.

[1] Other names are *continuous-discrete system*, *system with variable structure*, *event-driven system*.

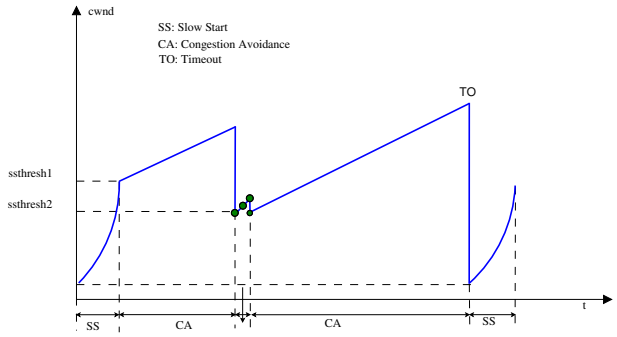

Fig. 1. TCP phases

Modelica supports continuous and hybrid (continuous-discrete) paradigms. However, the discrete elements are also present in the language.

Modelica language is presented by a large number of commercial implementations such as:

- Dymola [http://www.claytex.com/];
- CATIA [http://www.3ds.com/products-services/catia/];
- MapleSim [http://www.maplesoft.com/products/maplesim/];
- Wolfram SystemModeler [http://www.mathcore.com/].

There are also open source implementations of Modelica language and the environment:

- OpenModelica [https://openmodelica.org/];
- Scicos [http://www.scicos.org/];
- JModelica.org [http://jmodelica.org/].

HOW TCP WORKS

The TCP protocol uses a sliding window mechanism to avoid a congestion. The implementation of this mechanism depends on the particular type of TCP protocol.

TCP congestion control mechanism

Since the original model (see Misra et al. (1999, 2000); Velieva et al. (2014)) is based on the TCP Reno protocol, then this particular protocol will be simulated.

In TCP Reno protocol the congestion control mechanism consists of the following phases: slow start, congestion avoidance, fast transfer and fast recovery. Dynamics of changes in congestion window size (CWND) depends on the specific phase (see Fig. 1).

Each time the source receives a delivery notification (Acknowledge, ACK), in slow start phase congestion window is increased. The source increases the congestion window size depending on the number of confirmed segments (Segment Size, SS): $cwnd = cwnd+1$ for each transmitted ACK. Initial congestion window size (Maximum Segment Size, MSS) can take a value of 1, 2 or 10 segments. The receiver sends an ACK for each packet, but in reality it can be assumed that confirmation come together at the end of the double turnaround time (Round-Trip Time, RTT). Thus, the congestion window is doubled after the round-trip time.

When TCP Reno window size takes the certain value the protocol mechanism enters the congestion avoidance phase. In this phase, the congestion window is increased by the amount of $1/cwnd$ for each acknowledgment ACK, which is equivalent to increasing of the window by one packet for the double-turn.

Protocol TCP Reno monitors two options of packet loss:

- Triple Duplicate ACK (TD). Let n-th package is not delivered, and subsequent packets ($n+1$, $n+2$, etc.) are delivered. For each packet delivered in violation of prioritization(for $n+1$, $n+2$, and so on), the recipient sends ACK message for the last undelivered (n-th) package. With receiving three such packets the source resends the n-th package. In addition, the window size is decreased by 2 times $cwnd \to cwnd/2$.
- Timeout (TO). While sending a package the timeout timer is started. After receiving the confirmation the timer is restarted. Wherein the window size is set to the initial value of the congestion window. The first lost package is resent. The protocol passes into a slow start phase.

Overall congestion control algorithm belongs to AIMD algorithms type (Additive Increase, Multiplicative Decrease) — an additive increase of the window size and multiplicative decrease of it.

The transition to the continuous model for the congestion window

Because we want to construct the hybrid continuous-discrete model, we need pass to the continuous-time model in order to describe the operation of each TCP phase. The transition between the phases will be described by discrete states.

Using the results of section , the behavior of our model may be formalized. A congestion window change is described by an elementary event, which corresponds to a single acknowledgment or confirmation of all. Let us assume that the elementary event is the arrival of all acknowledgments that occurs during the round-trip time (RTT).

In the slow start phase a congestion window size increases with each occurrence of confirmation (ACK):

$$W(t_n^{ACK} + \Delta t^{ACK}) = W(t_n^{ACK}) + 1. \qquad (1)$$

We now rewrite (1) relative to the round-trip time T:

$$W(t_n + \Delta t) = W(t_n) + 1 \cdot W(t_n),$$
$$\frac{W(t_n + \Delta t) - W(t_n)}{\Delta t} = \frac{W(t_n)}{\Delta t}.$$

Assuming that $\Delta t = T$, we obtain

$$\frac{\mathrm{d}W}{\mathrm{d}t} = \frac{W}{T},$$

$$\mathrm{d}\ln W = \frac{\mathrm{d}t}{T}, \quad \ln W = \frac{1}{T}; \quad W = \exp\left\{\frac{1}{T}\right\}.$$

Thus, the window grows exponentially, as it should be in a slow start phase in accordance with the TCP description.

Similarly, we examine congestion avoidance phase. For each occurrence of an ACK the window size is increased:

$$W(t_n^{ACK} + \Delta t^{ACK}) = W(t_n^{ACK}) + \frac{1}{W(t_n^{ACK})}.$$

We rewrite this for a round-trip time:

$$W(t_n + \Delta t) = W(t_n) + \frac{1}{W(t_n)}W(t_n);$$
$$\frac{W(t_n + \Delta t) - W(t_n)}{\Delta t} = \frac{1}{\Delta t}.$$

Assuming that $\Delta t = T$, we have

$$\frac{\mathrm{d}W}{\mathrm{d}t} = \frac{1}{T},$$
$$\mathrm{d}W = \frac{\mathrm{d}t}{T}, \quad W = \frac{1}{T}.$$

The result is a linear increase of the window, as described in the specification of the TCP.

Construction of hybrid model for TCP

To construct a hybrid model we need:

- to write a dynamic model for each state (done in section);
- to replace the system with step parameters by the system with variable initial conditions;
- to write the state diagram (Fig. 2).

The resulting chart (Fig. 2) can be converted to a Modelica program. We give a fragment of the listing. Here we demonstrate only the state transition algorithm for TCP. As can be seen, it is made by almost verbatim copying of UML-diagrams (it is theoretically possible to carry out the code generation based on the corresponding chart). For greater clarity, the certain variables before the algorithm[2] are given.

RED ADAPTIVE CONGESTION CONTROL MECHANISM

To improve the channel performance the queue management on routers needs to be optimized. One of the possible approaches is to use the random early detection algorithm (Random Early Detection, RED) (see Floyd and Jacobson (1993)) or RED modifications.

Generally, the RED algorithm is very simple, but at the same time it presents the effective model of active traffic management. In addition, it imposes virtually no restrictions on the researcher. As a result, researchers are constantly creating new modifications of this algorithm (see Floyd and Jacobson (1993); Feng et al. (2015); Lautenschlaeger and Francini (2015); Karmeshu et al. (2016)).

[2]Note that the function *edge()* (used only with *Boolean* variables) is set to *true* only in case the operand only just received value *true*(that is, its previous value was *false*).

Listing 1. State transition algorithm for TCP protocol

```
model TCPSender "Transmission Control
    Protocol"
  // Variables
  type TCPState = enumeration(slowStart,
      fastRecov, congestAvoid, timeOut);
  discrete TCPState state(start =
      TCPState_slowStart);
  parameter Real timeout_th = 4.0 "
      Window threshhold for entering
      timeout";
  Real ssth(start = 32.0) "Slow start
      thresholds";
  Real w(start = MinSS, min = 1, max =
      w_max) "Communication window sizes,
       no of packets";
  Real drop_timer(start = 0) "Drop timer
      ";
  Real retr_timer(start = 0) "
      Retransmission timer";
  Boolean DelayD(start = false) " Delay
      induced Drop detected";
  //
  // Some code
  //
algorithm
  // State transitions
  state := TCPState.slowStart;
  when edge(DelayD) and w >= timeout_th
      and (state == TCPState.slowStart or
       state == TCPState.congestAvoid)
      then
    state := TCPState.fastRecov;
  elsewhen w >= ssth and state ==
      TCPState.slowStart then
    state := TCPState.congestAvoid;
  elsewhen w < timeout_th and edge(
      DelayD) and (state ==
      TCPState.slowStart or state ==
      TCPState.congestAvoid) then
    state := TCPState.timeOut;
  elsewhen retr_timer < 0 and state ==
      TCPState.fastRecov then
    state := TCPState.congestAvoid;
  elsewhen retr_timer < 0 and state ==
      TCPState.timeOut then
    state := TCPState.slowStart;
  end when;
end TCPSender;
```

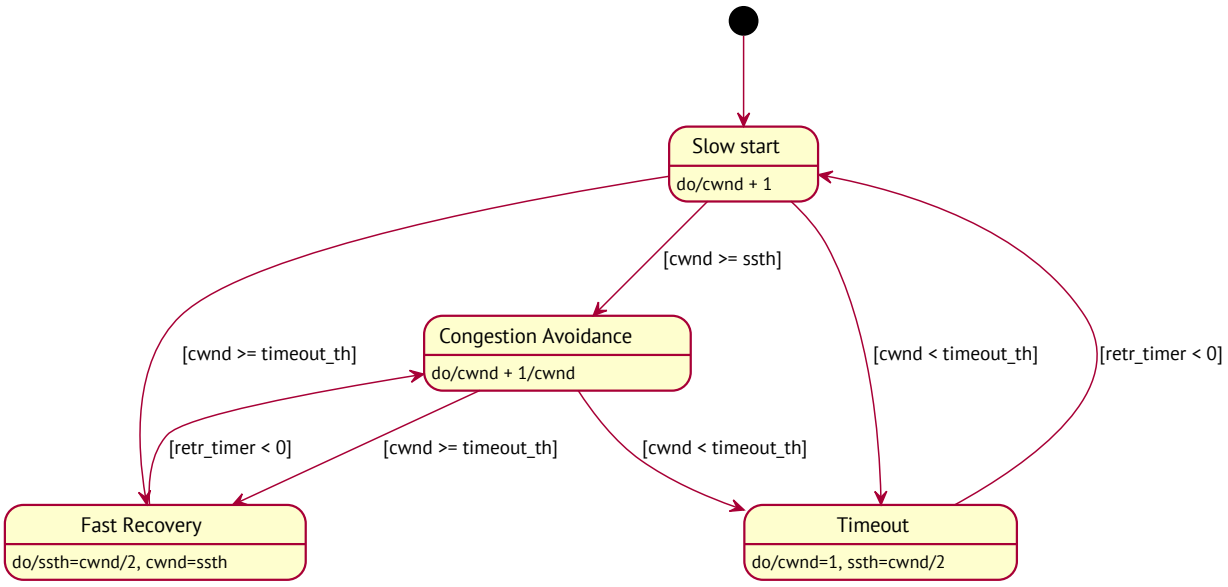

Fig. 2. TCP state diagram

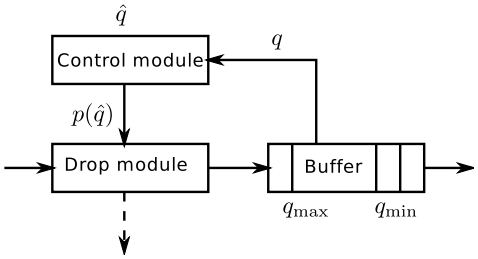

Fig. 3. RED module

The previous implementation of the RED algorithm (see Velieva et al. (2014)) was done in the ideology of continuous modeling. Therefore, we were limited to the following model assumptions:

- we have considered only the congestion avoidance stage;
- we took into account the loss of packets only by triple duplication.

The hybrid modeling allows us to overcome this barrier and to investigate the mechanism in its entirety.

How RED works

The module that implements the RED-type algorithm can be schematically represented as follows (fig. 3):

RED algorithm uses a weighted value of the queue length $\hat{Q}$ as factor for the determination of packet dropping probability. In order to calculate $\hat{Q}$ the exponentially weighted moving-average (EWMA) is used:

$$\hat{Q}_{k+1} = (1 - w_q)\hat{Q}_k + w_q Q_k, \quad k = 0, 1, 2, \ldots,$$

where w_q, $0 < w_q < 1$ is a weight coefficient of the exponentially weighted moving-average.

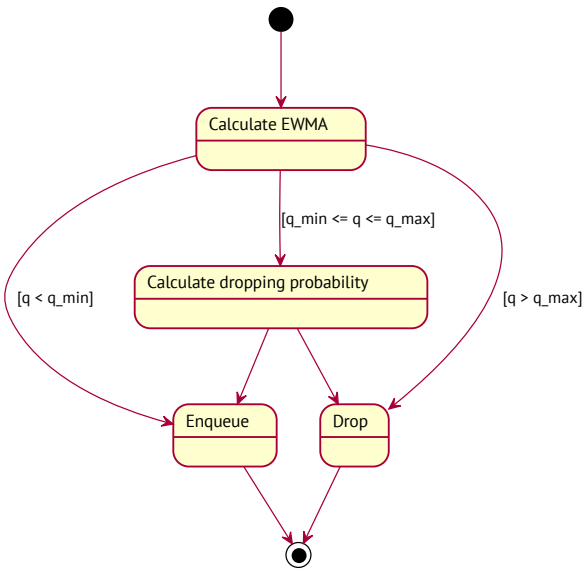

Fig. 4. RED state diagram

As the the average queue length increases, the packets drop probability also increases (see. (2)). For dropping management algorithm the dropping function with two average queue length thresholds $Q_{\min}$ and $Q_{\max}$ as arguments (Fig. 5) is used:

$$p(\hat{Q}) = \begin{cases} 0, & 0 < \hat{Q} \leqslant Q_{\min}, \\ \dfrac{\hat{Q} - Q_{\min}}{Q_{\max} - Q_{\min}} p_{\max}, & Q_{\min} < \hat{Q} \leqslant Q_{\max}, \\ 1, & \hat{Q} > Q_{\max}. \end{cases} \quad (2)$$

Here $p(\hat{Q})$ is the package dropping function, $\hat{Q}$ is the queue length weighted average, $Q_{\min}$ and $Q_{\max}$ are thresholds of queue length weighted average, $p_{\max}$ is the maximum level of packages reset.

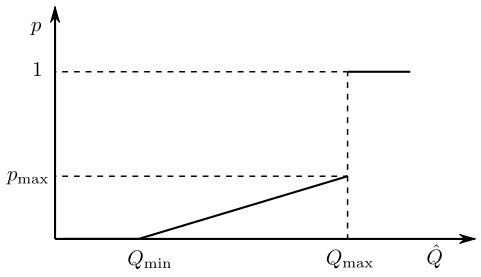

Fig. 5. RED drop function

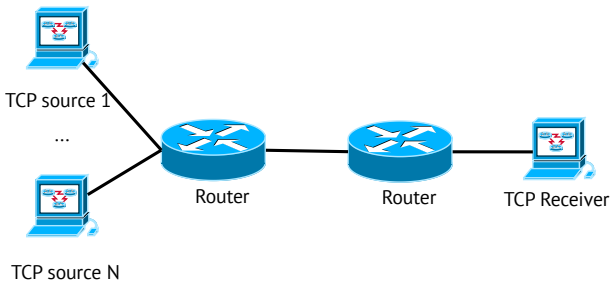

Fig. 6. Dumbbell topology

The transition to the continuous model for RED algorithm

The behavior of an exponentially weighted moving average queue length, which is a constraint equation between the source and the receiver, is described further.

Let w_q is the parameter of the moving average (see Floyd and Jacobson (1993)). Based on the formula for exponentially weighted moving average, we can write:

$$\hat{Q}(t_n + \Delta t) = (1 - w_q)\hat{Q}(t_n) + w_q Q(t_n),$$
$$\hat{Q}(t_n + \Delta t) - \hat{Q}(t_n) = -w_q\hat{Q}(t_n) + w_q Q(t_n),$$
$$\frac{\hat{Q}(t_n + \Delta t) - \hat{Q}(t_n)}{\Delta t} = \frac{w_q}{\Delta t}(Q(t_n) - \hat{Q}(t_n)).$$

Let $\delta = \Delta t$. Let us give δ meaning of time spent by one packet in the queue. Assuming that C is the service intensity, we can write $\delta = \frac{1}{C}$. Then the equation for the exponentially weighted moving average queue length takes the form:

$$\frac{\mathrm{d}\hat{Q}}{\mathrm{d}t} = \frac{w_q}{\delta}(Q - \hat{Q}) = w_q C(Q - \hat{Q}).$$

Construction of hybrid model for RED algorithm

To construct a hybrid model, we need:

- to write a dynamic model for each state;
- to replace the system with step parameters by the system with variable initial conditions;
- to write the state diagram (Fig. 4).

In general RED model is more simple than TCP model. Especially considering that the RED model has only one state, hence there is no need to encode the transitions between states.

THE GENERAL SCHEME OF THE MODEL

For the study of interaction between active queue management module and the TCP traffic source we will use a simple dumbbell topology (Fig. 6). For all its simplicity, it gives a good description of the basic elements of our model.

In Modelica language the model is written as follows (see listing 2). Routers play only the role of connectors, without any other functionality. RED module is configured as a separate class. To set a delay in the network, we introduced the links.

Listing 2. Implementation of the topology

```
model RED_hybrid "RED hybrid model"
  Router router1;
  Router router2;
  TCPSender aTCP(L = 500);
  Receiver receiver;
  REDqueue red_queue;
  Link link1(delay = 0.02);
  Link link2(delay = 0.02);
equation
  connect(aTCP.o, link1.i);
  connect(link1.o, router1.i);
  connect(router1.o, red_queue.i);
  connect(red_queue.o, router2.i);
  connect(router2.o, link2.i);
  connect(link2.o, receiver.i);
end RED_hybrid;
```

CONCLUSIONS

We investigated the RED active traffic management mechanism as the implementation of the system with a threshold control. Some mathematical models (both analytical and simulation) of this mechanism using different paradigms and techniques were presented. On closer inspection, the presented modeling techniques have shown their shortcomings.

The considered in the article hybrid (continuous-discrete) approach seems to be the most appropriate for network protocol modeling.

A hybrid approach can be used both in the analytical modeling, and in simulations. A hybrid approach can be used both in the analytical modeling, and in simulations. Unfortunately, this approach does not actively used by researchers, although it is implemented in a number of computer simulation systems.

In the future, it seems useful to develop a library for simulation of the interaction between different types of TCP protocol with different versions of RED algorithm.

a) Notes and Comments: The work is partially supported by RFBR grants No's 14-01-00628, 15-07-08795, and 16-07-00556.

REFERENCES

Abaev, P., Gaidamaka, Y., Samouylov, K., Pechinkin, A., Razumchik, R. and Shorgin, S. (2014), Hysteretic control technique for over-

load problem solution in network of SIP servers, *Computing and Informatics* 33(1), 218–236.

Altman, E. and Jiménez, T. (2012), NS Simulator for Beginners, *Synthesis Lectures on Communication Networks* 5(1), 1–184.

Bohacek, S. and Lee, J. (2001), Analysis of a TCP hybrid model, Proc. of the 39th Annual Allerton Conference on Communication, Control, and Computing, pp. 1–10.

Demidova, A. V., Korolkova, A. V., Kulyabov, D. S. and Sevastyanov, L. A. (2014), The method of constructing models of peer to peer protocols, 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), IEEE, pp. 557–562.

Eferina, E. G., Korolkova, A. V., Gevorkyan, M. N., Kulyabov, D. S. and Sevastyanov, L. A. (2014), One-Step Stochastic Processes Simulation Software Package, *Bulletin of Peoples Friendship University of Russia. Series "Mathematics. Information Sciences. Physics"* (3), 46–59.

Färnqvist, D., Strandemar, K., Johansson, K. H. and Hespanha, J. P. (2002), Hybrid Modeling of Communication Networks Using Modelica, The 2nd International Modelica Conference, pp. 209–213.

Feng, C.-W., Huang, L.-F., Xu, C. and Chang, Y.-C. (2015), Congestion Control Scheme Performance Analysis Based on Nonlinear RED, *IEEE Systems Journal* pp. 1–8.

Floyd, S. and Jacobson, V. (1993), Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Transactions on Networking* 1(4), 397–413.

Fritzson, P. (2003), *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*, Wiley-IEEE Press.

Fritzson, P. (2011), *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica*, John Wiley & Sons, Inc., Hoboken, NJ, USA.

Gaidamaka, Y., Pechinkin, A., Razumchik, R., Samouylov, K. and Sopin, E. (2014), Analysis of an M—G—1—R queue with batch arrivals and two hysteretic overload control policies, *International Journal of Applied Mathematics and Computer Science* 24(3), 519–534.

Hespanha, J. P., Bohacek, S., Obraczka, K. and Lee, J. (2001), Hybrid Modeling of TCP Congestion Control, Lncs, number 2034, pp. 291–304.

Issariyakul, T. and Hossain, E. (2012), *Introduction to network simulator NS2*, Vol. 9781461414.

Karmeshu, Patel, S. and Bhatnagar, S. (2016), Adaptive Mean Queue Size and Its Rate of Change: Queue Management with Random Dropping, pp. 1–17.

Lautenschlaeger, W. and Francini, A. (2015), Global Synchronization Protection for Bandwidth Sharing TCP Flows in High-Speed Links, Proc. 16-th International Conference on High Performance Switching and Routing, IEEE HPSR 2015, Budapest, Hungary.

Leland, W. E., Taqqu, M. S., Willinger, W. and Wilson, D. V. (1994), On the self-similar nature of Ethernet traffic (extended version), *IEEE/ACM Transactions on Networking* 2(1), 1–15.

Maler, O. (1992), Hybrid Systems and Real-World Computations, Workshop on Theory of Hybrid Systems, Springer-Verlag, Lyndby, Denmark.

Maler, O. (2002), Control from computer science, *Annual Reviews in Control* 26(2), 175–187.

Misra, V., Gong, W.-B. and Towsley, D. (1999), Stochastic differential equation modeling and analysis of TCP-windowsize behavior, *Proceedings of PERFORMANCE* 99.

Misra, V., Gong, W.-B. and Towsley, D. (2000), Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED, *ACM SIGCOMM Computer Communication Review* 30(4), 151–160.

Paxson, V. and Floyd, S. (1995), Wide area traffic: the failure of Poisson modeling, *IEEE/ACM Transactions on Networking* 3(3), 226–244.

Paxson, V. and Floyd, S. (1997), Why we don't know how to simulate the Internet, Proceedings of the 29th conference on Winter simulation - WSC '97, ACM Press, New York, New York, USA, pp. 1037–1044.

Velieva, T. R., Korolkova, A. V. and Kulyabov, D. S. (2014), Designing installations for verification of the model of active queue management discipline RED in the GNS3, 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), IEEE, pp. 570–577.

AUTHOR BIOGRAPHIES

ANNA V. KOROLKOVA received his Ph.D. in Mathematics in 2010. Since then, she has worked as associate professor in Peoples' Friendship University of Russia. Her current research activity focuses on mathematical modeling. Her email address is akorolkova@sci.pfu.edu.ru.

DMITRY S. KULYABOV received his Ph.D. in Physics in 2000. Since then, he has worked as associate professor in Peoples' Friendship University of Russia. His current research activity focuses on mathematical modeling. His email address is yamadharma@gmail.com.

LEONID A. SEVASTIANOV received his D.Sc. in Phys.-Math. in 1999. Since then, he has worked as full professor in Peoples' Friendship University of Russia. His current research activity focuses on mathematical modeling. His email address is leonid.sevast@gmail.com.

TATYANA R. VELIEVA postgaduate student in Peoples' Friendship University of Russia. Her current research activity focuses on mathematical modeling. Her email address is trvelieva@gmail.com.

PAVEL O. ABAEV received his Ph.D. in Computer Science from the Peoples' Friendship University of Russia in 2011. He is an Assistant Professor in the Department of Applied Probability and Informatics at Peoples' Friendship University of Russia since 2013. His current research focus is on Software-Defined Network, performance analysis of wireless 5G networks and M2M communications, applied probability and queuing theory, and mathematical modeling of communication networks. His email address is pabaev@sci.pfu.edu.ru.