MINIMIZING MEAN RESPONSE TIME IN BATCH-ARRIVAL NON-OBSERVABLE SYSTEMS WITH SINGLE-SERVER FIFO QUEUES OPERATING IN PARALLEL

Mikhail Konovalov and Rostislav Razumchik
Institute of Informatics Problems of the FRC CSC RAS
Vavilova, 44-2, 119333, Moscow, Russia
Email: mkonovalov@ipiran.ru, rrazumchik@ipiran.ru

KEYWORDS

dispatching, unobservable, mean response time, batch arrivals, load balancing, unreliable servers

ABSTRACT

Consideration is given to dispatching systems, where jobs, arriving in batches, cannot be stored and thus must be immediately routed to single-server FIFO queues operating in parallel. The dispatcher can memorize its routing decisions, but at any time instant does not have any system's state information. The only information available is the batch/job size and inter-arrival time distributions, and the servers' service rates. Under these conditions, one is interested in the routing policies which minimize the job's long-run mean response time. The single-parameter routing policy is being proposed which, according to the numerical experiments, outperforms best routing rules known by now for non-observable dispatching systems: probabilistic and deterministic. Both batch- and job-wise assignments are studied. Extension to systems with unreliable servers is discussed in short.

INTRODUCTION

Efficient resource allocation is the typical problem faced by system designers in various fields of study (transportation, distributed/parallel computing, customs inspection etc.). The particular problem studied in this paper has its roots in the volunteer computing. Consider a system in which jobs (of a single class) arrive according to some stochastic process in batches and have to be assigned to one of the single-server queues immediately upon the arrival. Scheduling in each queue is FIFO. The dispatcher, which performs this operation (see Fig. 1), can memorize its routing decisions, but has no online information about the system except for: the cumulative distribution function (CDF) of inter-arrival times, batch size and job size CDFs and servers' service rates. Thus for the dispatcher the system is non-observable (see, for example, Anselmi and Gaujal (2011); Lingenbrink and Krishnamurthy (2017)). The task is to find the routing policy, which minimizes the job's long-run mean sojourn time in the system[1]. Since the dispatcher may decide to split batches, when making decisions, both cases — batch- and job-wise assignments — need to be considered.

[1]Or, equivalently, the system's mean response time.

The described system is closely related to the basic dispatching problem studied extensively in the literature (Harchol-Balter, Crovella and Murta (1999); Hyytiä and Aalto (2013); Feng, Misra and Rubenstein (2005)). But, due to the absence of any online information, from the long list of routing rules, only the following two naive (in terminology of (Mengistu and Che, 2019, Section 2.3)) policies are feasible: probabilistic and deterministic. According to the numerical experiments (see Konovalov and Razumchik (2020)), both these rules can be outperformed by the algorithm, which implements the so-called arrival-aware policy (see Konovalov and Razumchik (2018)) for single-arrival non-observable systems. In this paper numerical evidence is given that all the three policies (probabilistic, deterministic and arrival-aware) can be applied by the dispatcher in a batch-arrival system. We rank the policies with respect to the minimal mean response time (and its standard deviation) and discuss extensions of the system as well as of the arrival-aware policy.

The paper is organized as follows. In the next section the detailed description of system is given. The third section contains the overview of the routing policies available to the dispatcher in the considered setting. The numerical examples, which follow, demonstrate the performance of the policies. In the Section 5 the study is extended to systems with unreliable servers. The main conclusions are briefly summarized in Section 6.

SYSTEM DESCRIPTION AND THE PROBLEM STATEMENT

The system considered in the paper is illustrated in Fig. 1.

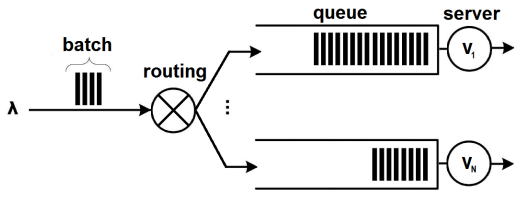

Figure 1: Jobs of a single class arrive in batches and are immediately routed to one of the queues, where they wait in FIFO order for service.

It consists of $N \geq 2$ single server infinite capacity queues, operating in parallel. The queues are numbered from

Communications of the ECMS, Volume 35, Issue 1,
Proceedings, ©ECMS Khalid Al-Begain, Mauro Iacono,
Lelio Campanile, Andrzej Bargiela (Editors)
ISBN: 978-3-937436-72-2 / 78-3-937436-73-9(CD) ISSN 2522-2414

1 to N and the server's service rate of queue m is $v^{(m)} > 0$, $1 \le m \le N$. Jobs (of a single class) arrive to the system in batches and the inter-arrival times are independent and identically distributed (i.i.d.) random variables with the known CDF $A(x)$ and the mean λ^{-1}. Consecutive batches consist[2] of random numbers $B_1, B_2, \dots$ of jobs having the known distribution[3] $\{g_i, i = 1, 2, \dots\}$, with the mean $\mathsf{E}[B]$ and the variance $\mathsf{Var}[B]$. The sizes $X_1, X_2, \dots$ of individual jobs are i.i.d. random variables with the known CDF $S(x)$, mean $\mathsf{E}[X]$ and variance $\mathsf{Var}[X]$. Jobs are served one-by-one and the service discipline employed in each queue is FIFO. Service pre-emption and jockeying between queues is not allowed.

The dispatcher operates either in the batch-wise or in the job-wise mode. The former means that it treats a batch as if it were a single (macro-) job and makes a single routing decision (i.e. all the jobs in the batch are assigned to the same queue). In the latter case the dispatcher splits batches and chooses a queue for each job individually. Switching between the modes is not allowed. Upon receiving a batch the dispatcher must immediately[4] make a routing decision.

Fix an arbitrary integer $n \ge 1$. Let $0 \le t_1 < \dots < t_n$ and $b_1, b_2, \dots, b_n$ denote the arrival instants and the sizes of the first n batches, correspondingly. Let $y_1, y_2, \dots, y_k$ be the sequence of routing decisions[5] made so far at the instants $t_1, t_2, \dots, t_{n-1}$. Each y_j takes a value from the set $\{1, 2, \dots, N\}$. In order to make routing decisions at the instant t_n, the dispatcher may use only the following information: the values $t_1, t_2, \dots, t_n$ and $b_1, b_2, \dots, b_n$; the values $y_1, y_2, \dots, y_k$; the distributions $A(x)$, $S(x)$ and $\{g_i, i = 1, 2, \dots\}$; the service rates $v^{(1)}, v^{(2)}, \dots, v^{(N)}$ in each queue. No online information (like the arriving job size, current queues' sizes etc.) is available. The dispatcher's task is to minimize job's long-run mean sojourn time in the system.

OVERVIEW OF THE AVAILABLE DISPATCHING POLICIES

The list of dispatching policies, which are available to the dispatcher under the assumptions made (i.e. in the absence of any system's state information), is very short: probabilistic (see, for example, Bell and Stidham (1983)), deterministic (see Hordijk and van der Laan (2004)) and arrival-aware (see Konovalov and Razumchik (2020)).

According to a probabilistic routing policy (further referred to as RND) a job is routed to the queue n with the probability p_n independently of the previous decisions. Recall that in the batch-wise mode the dispatcher assigns the same queue for each job within a batch. Let the arrival flow of batches be Poisson with the rate λ. Then the n^{th} queue is the $M^{[X]}/GI/1$-FIFO queue with the arrival rate λp_n and, whenever it is stable, the mean response time $\mathsf{E}[T_n]^{\text{FIFO}}$ of an arbitrary job is equal to (see (Cooper, 1981, p. 241))

$$\mathsf{E}[T_n]^{\text{FIFO}} = \frac{(\mathsf{E}[B]+1)\mathsf{E}[X^{(n)}]}{2} + \frac{\mathsf{Var}[B]\mathsf{E}[X^{(n)}]}{2\mathsf{E}[B]} + \\ + \frac{\lambda p_n \mathsf{E}[B]^2 \mathsf{E}[X^{(n)}]^2 \left(1 + \frac{\mathsf{E}[B]\mathsf{Var}[X^{(n)}] + \mathsf{E}[X^{(n)}]^2 \mathsf{Var}[B]}{\mathsf{E}[B]^2 \mathsf{E}[X^{(n)}]^2}\right)}{2(1 - \lambda p_n \mathsf{E}[B]\mathsf{E}[X^{(n)}])}, \quad (1)$$

where the notation $X^{(n)} = X/v^{(n)}$ is used. The optimal probabilistic routing policy, further referred to as RND-opt, is the probability distribution $(p_1, p_2, \dots, p_N)$ that minimizes the mean response time given by

$$\sum_{n=1}^{N} p_n \mathsf{E}[T_n]^{\text{FIFO}} \quad (2)$$

under the constraint $0 \le \lambda p_n \mathsf{E}[X^{(n)}]\mathsf{E}[B] < 1$ for each n. This problem can be approached either analytically[6] or numerically and, in the latter case, it can be solved at a satisfactory level. If the batch-arrival process is not Poisson, but a general renewal process with the mean λ^{-1} and SCV (squared coefficient of variation) equal to C_A^2, then the sequence of arrival instants to the queue n constitutes the renewal process with the mean $(\lambda p_n)^{-1}$ and the SCV equal to $1 + (C_A^2 - 1)p_n$. But (1) is not valid any more and we are unaware of any feasible way to compute numerically both $\mathsf{E}[T_n]^{\text{FIFO}}$ and the optimal N-tuple $(p_1, p_2, \dots, p_N)$. For small values of N simulation can be used to find the approximate solution of the minimization problem. In general, load balancing (i.e. $p_n = v^{(n)} / \sum_{i=1}^{N} v^{(i)}$) seems to be the only reasonable trade-off, but in most cases it is far from being optimal. When the dispatcher operates in the job-wise mode, a queue is assigned for each job individually according to the given probability distribution $(p_1, p_2, \dots, p_N)$. But even if the batches arrive according to a Poisson flow, the optimal N-tuple is not the solution of (2). For a batch of size k the probability that k_1 jobs are routed to the queue 1, ..., k_N jobs — to the queue N, is equal to

$$\frac{k!}{k_1! k_2! \dots k_N!} (p_1)^{k_1} \dots (p_N)^{k_N}, \quad (3)$$

where $\sum_{i=1}^{N} k_i = k$, $0 \le k_i \le N$. Since the consecutive batches are treated independently, for the systems with Poisson arrivals and exponential job size distributions (3) can be used to write out the balance equations[7], which can be solved numerically (after truncation of the state space). Once the joint stationary distribution of

[2] No precedence constraints are imposed on jobs within a batch.

[3] It is worth noticing that in batch-arrival queues the limiting distributions of some quantities may not exist without additional restrictions on the batch-size distribution $\{g_i\}$ (see, for example, (van Ommeren, 1990, p. 679)).

[4] I.e. it does not have a room for storing the jobs.

[5] The total number k of decisions before the instant t_n depends on the operation mode of the dispatcher.

[6] See, for example, traffic/resource allocation problems in Ibaraki and Katoh (1988).

[7] For example, for $N = 2$ the evolution of the system's content can be described by the QBD process. Assuming the capacity of either queue to be finite, the generator is then of $M/G/1$-type.

queue-sizes' is found, standard argument can be used to find (approximately) the system's mean response time. This procedure theoretically can be used to search for the (close to) optimal N-tuple $(p_1, p_2, \ldots, p_N)$. But in practice (especially in heavy traffic) the computational complexity becomes prohibitive and, as well as in the case of a general renewal arrival process, one has to resort to simulation.

According to a deterministic policy jobs are dispatched according to the infinite sequence $a_1, \ldots, a_n, \ldots$, where a_i is the queue number, whereto the i^{th} job is routed[8]. Finding optimal deterministic sequence for N single server queues, operating in parallel with single arrivals, is a difficult problem for which no general procedures exist; obviously the same holds for batch-arrival systems. Yet when jobs arrive to the dispatcher not in the batches but one at a time, very good results can be achieved by special deterministic sequences – billiard sequences[9], which can be constructed using greedy algorithms. One of such algorithms further referred to as SG (Special Greedy) is due to (Hordijk and van der Laan, 2004, p.184). According to the SG policy the i^{th} job is routed to the queue a_i:

$$a_i = \text{argmin}_{1\le n\le N}\left(\frac{x_n + \kappa^n(i-1)}{d_n}\right), \qquad (4)$$

where $\kappa^n(i)$ is equal to the the number of jobs (among the first i jobs) sent to the queue n so far, d_n is the fraction of jobs, which has to be routed to the queue n, and x_n are properly chosen non-negative rational numbers[10]. Finding the optimal densities $(d_1, d_2, \ldots, d_N)$ is an open problem and, in general, can be approached only using simulation. As a rule of thumb one can use in (4) instead of $(d_1, d_2, \ldots, d_N)$ the N-tuple $(p_1, p_2, \ldots, p_N)$ computed for the probabilistic routing policy. Even though such choice usually results not in the optimal solution, deterministic routing is more efficient[11] than the probabilistic routing. We are unaware of any deterministic policy developed particularly for batch-arrival systems. And since all the quantities in (4) remain properly defined for both batch-wise and job-wise routing, we choose (4) as the basic deterministic dispatching policy.

The third class of routing policies further referred to as AA (Arrival-Aware) is based on the following intuitive idea (see Konovalov and Razumchik (2018)): longer inter-arrival times increase the probability of those system's states, which correspond to lower workloads in queues and vice versa. It is not straightforward to implement this idea in a batch-arrival system[12]. But in Konovalov and Razumchik (2020) for single-arrival unobservable systems it was suggested to put the idea into practice by means of the algorithm[13], which is reproduced below. Fix the positive real $c > 0$. Let us associate with the i-th job arriving at the dispatcher, N numbers, say $u_i^{(1)}, \ldots, u_i^{(N)}$, which are defined recursively as follows:

$$\tilde{u}_i^{(n)} = \max\left(0, u_{i-1}^{(n)} - (t_i - t_{i-1})\right),\ 1 \le n \le N,\ i \ge 1,$$

$$u_i^{(n)} = \begin{cases} \tilde{u}_i^{(\tilde{y}_i)} + \frac{c}{v^{(\tilde{y}_i)}}, & \text{if } n = \tilde{y}_i, \\ \tilde{u}_i^{(n)}, & \text{otherwise,} \end{cases}$$

where $u_0^{(1)} = b_1, \ldots, u_0^{(N)} = b_N$ and[14]

$$\tilde{y}_i = \text{argmin}_{1\le n\le N}\left(\tilde{u}_i^{(n)} + \frac{c}{v^{(n)}}\right).$$

The AA policy prescribes to route the i^{th} job to the queue $y_i = \tilde{y}_i$. The pseudocode for the policy is given below (see Algorithm 1). Unlike the RND and SG policies, the AA policy depends only on a single parameter, which, in general, must be estimated using simulation. All the quantities in the Algorithm 1 remain[15] properly defined for batch-arrival systems as well. Thus we choose it as the basic AA policy.

Algorithm 1 Pseudocode of the AA policy

function NextDecision(N,$v^{(1)}$,…,$v^{(N)}$,$u_{i-1}^{(1)}$,…,$u_{i-1}^{(N)}$,t_i, t_{i-1},c)
  for $n = 1 \to N$ **do**
    $u_i^{(n)} = \max\left(0, u_{i-1}^{(n)} - (t_i - t_{i-1})\right)$
  end for
  $y_i = \text{argmin}_{1\le n\le N}\left(u_i^{(n)} + c/v^{(n)}\right)$
  $u_i^{(y_i)} = u_i^{(y_i)} + c/v^{(y_i)}$
  return $y_i, u_i^{(1)}, \ldots, u_i^{(N)}$
end function

[a] The function NextDecision(N,$v^{(1)}$,…,$v^{(N)}$,$u_{i-1}^{(1)}$,…,$u_{i-1}^{(N)}$,t_i,t_{i-1}) returns for the i^{th} job the routing decision y_i based on the i^{th} job arrival instant t_i and the arrival instant t_{i-1} of the $(i-1)^{th}$ job, servers' speeds $v^{(1)}, \ldots, v^{(N)}$, auxiliary values $u_{i-1}^{(1)}, \ldots, u_{i-1}^{(N)}$ and c.
[b] The values $u_0^{(1)}, \ldots, u_0^{(N)}$ are the initial remaining workloads (including server). For the initially empty system $u_0^{(1)} = \cdots = u_0^{(N)} = 0$.
[c] The positive real value c is the parameter of the algorithm, which must be set manually.

Numerical examples in the next section give some impression on how these dispatching policies are ranked,

[8] One of the well-known examples of such a policy is the Round-Robin policy. It rotates the jobs between the queues in the cyclic order and thus is applicable only in the systems with homogeneous servers (i.e. having equal service rates). For heterogeneous systems some generalizations do exist (see Arian and Levy (1992)).

[9] Although other deterministic policies besides (4) do exist (see, for example, GRR, CGRR, and mBS policies in Arian and Levy (1992); Sano and Miyoshi (2000)), according to our experience with scheduling problems in non-observable queues (see, for example, Konovalov and Razumchik (2018, 2020)), (4) shows the best performance.

[10] For example, one can put $x_n = 1$ if the queue n has the fastest server and $x_n = 0$ otherwise.

[11] Some numerical and analytic evidences are given, for example, in Anselmi (2017).

[12] We conjecture that the algorithm of Konovalov and Razumchik (2018) can be adopted to the considered system at least in the case when the dispatcher operates in the batch-wise mode, i.e. assigns the whole batch to the same server.

[13] See also the footnote 9 on page 400 in Konovalov and Razumchik (2020).

[14] When argmin() is being evaluated, ties are broken in the favour of the fastest server and randomly between the fastest servers.

[15] The presented version of the AA policy (Algorithm 1) is tuned for the batch-wise assignment. In case of job-wise assignment the Algorithm 1 has to be applied to each job in the batch.

when one seeks to minimize the mean (and even the standard deviation of the) sojourn time of an arbitrary job as well as of a whole batch.

NUMERICAL EXPERIMENT

In the first example we consider an elementary system with the two servers processing jobs with the total service rate equal to 1. Let $v^{(1)} = 0,326$ and $v^{(2)} = 0,674$. Batches arrive according to the Poisson process with the rate λ. The batch size distribution is geometric with the mean $\mathsf{E}[B] = 3$ ($\mathsf{Var}[B] = 6$) and the job size distribution is exponential with the mean $\mathsf{E}[X] = 1$. The offered load to the whole system is thus $\rho = \lambda \mathsf{E}[B]\mathsf{E}[X]/\sum_{i=1}^{2} v^{(i)} = 3\lambda$. Since all the three considered policies (RND, SG and AA) can be applied with batch-wise and job-wise assignment, this yields 3 batch- and 3 job-specific policies. In the considered two-server fully exponential system the approximate values of the optimal parameters (p_1, p_2) for the RND policies are found numerically following the guidelines in the previous section. The (close to) optimal parameters (d_1, d_2) and c of the SG and the AA policies, respectively, are estimated through simulation. The numerical results (values of the mean and the standard deviation of the response time depending on the offered load ρ) for the batch-wise assignment are given in the Table 1 and for the job-wise assignment — in the Table 2.

The first observation from the data in the Tables 1 and 2 is that the SG policy and the AA policy, which were to be used for single-arrival systems, show meaningful results for the batch-arrival system as well. And this observation remained valid in all our numerical experiments. Next, both the SG and the AA policies give lower mean (and standard deviation of the) response time than the RND policy (with the optimal parameters' values). As the system's load increases it becomes less and less appealing to route the jobs/batches according to the RND-opt (even though it has the strongest theoretical support among the three). The performance of the SG and the AA policies is (roughly speaking) the same. Yet it can be noticed that for low and moderate system's load the AA policy is slightly (but persistently) better. The optimal values (p_1, p_2) and (d_1, d_2) are not equal (except for the case of high system's load): using (p_1, p_2) instead of (d_1, d_2) in the SG policy (4) leads to worse results.

When the dispatcher splits the batches (i.e. performs job-wise assignment), it may be important to route jobs in such a way so as to minimize the mean sojourn time of the whole batch[16]. The extent at which the RND, SG and AA policies cope with this task can be assessed from the Tables 2 and 3. By comparing the data it can be said that the results follow the intuition: the mean response time of an individual job is lower than of a whole batch. In can be also seen that the ranking of the policies remains unchanged.

If one ranks the policies, depending on the number of parameters requiring estimation, then in the first example all the policies are identical. The second example, considered further, is intended to bring in the difference in this issue. Consider the system with 128 servers processing jobs with the total service rate equal to 1. Servers are aggregated into 8 groups of equal size i.e. each of the 16 servers in the group number i, $1 \le i \le 8$, has the service rate $i/576$. Batches arrive according to the hyper-exponential process with SCV= 4, two phases and balanced means i.e. the phase probabilities are

$$\alpha_1 = \frac{1}{2}\left(1 + \sqrt{\frac{\mathrm{SCV} - 1}{\mathrm{SCV} + 1}}\right) \approx 0,8873, \quad \alpha_2 \approx 0,1127.$$

Thus the arrival rate is $\lambda = (0,8873/\lambda_1 + 0,1127/\lambda_2)^{-1}$. The values of λ_1 and λ_2, fixed to fit the chosen values of the system's load, are reported in the Tables 4 and 5. The job size distribution is chosen to be bimodal: a job has size either $0,5$ or 9 with probabilities of $16/17$ and $1/17$ respectively. Thus the mean job size is $\mathsf{E}[X] = 1$ ($\mathsf{Var}[X] = 4$). The assumed batch size distribution is specified in the following table:

i	1	2	4	8
g_i	$0,375$	$0,125$	$0,125$	$0,375$

The mean batch size is $\mathsf{E}[B] = 4,125$ ($\mathsf{Var}[B] \approx 9,86$). The offered load to the whole system is thus $\rho = \lambda \mathsf{E}[B]\mathsf{E}[X]/\sum_{i=1}^{128} v^{(i)} = 4,125\lambda$. In order to determine the parameters of the RND and SG policies, we notice that in each group the service rates are identical; thus it is reasonable[17] to use Round-Robin routing within a group. Consequently for each of the two policies, RND and SG, one has to estimate 7 parameters. The values of $(p_2, \ldots, p_8)$ for the RND policy can be set so as to balance the load (this choice is further denoted by RND-lb) or they can be optimized through simulation (RND-opt, see Table 6). By analogy there are two options for the SG policy: SG-lb and SG-opt[18]. The AA policy requires a single parameter c, which is estimated through simulation. The values of the mean and the standard deviation of the response time depending on the offered load ρ for the batch-wise assignment are given in the Table 4 and for the job-wise assignment — in the Table 5.

Compared to the first example here all the random quantities (inter-arrival time, batch-size and job-size) are more variable. Yet qualitatively the results are similar. The SG-opt and AA policies always outperform (any of the two) the RND policies. With respect to both the mean response time and its standard deviation the AA policy is

[16] Under such a requirement, all jobs belonging to the same batch wait until the last member of the batch is processed, and thus the considered system becomes somewhat similar to a fork-join or a split-merge system.

[17] And such a rule is superior to random routing.

[18] We note that in the presented examples the parameters $(d_2, \ldots, d_8)$ of the SG-opt policy were, in fact, set equal to the parameters of the RND-opt, given in the Table 6. Thus the results under the SG policy can be potentially improved but not affecting the conclusions made.

Table 1: Job's mean (and the standard deviation of the) response time in the two-server system. Batch-wise assignment. Service rates: $\nu^{(1)} = 0,326$ and $\nu^{(2)} = 0,674$. Poisson batch-arrivals with the rate λ. Batch size is geometrically distributed with the mean $\mathsf{E}[B] = 3$. Job size distribution is exponential with the mean $\mathsf{E}[X] = 1$. The offered load is $\rho = 3\lambda$.

	$\lambda = 0,05$ $\rho = 0,15$	$\lambda = 0,1$ $\rho = 0,30$	$\lambda = 0,18$ $\rho = 0,54$	$\lambda = 0,2$ $\rho = 0,60$	$\lambda = 0,23$ $\rho = 0,69$	$\lambda = 0,25$ $\rho = 0,75$	$\lambda = 0,32$ $\rho = 0,96$
RND-opt	5,73 (5,74) $p_1 = 0$	7,68 (7,87) $p_1 = 0,129$	12,29 (12,69) $p_1 = 0,254$	14,22 (14,63) $p_1 = 0,269$	18,48 (18,97) $p_1 = 0,288$	22,99 (23,66) $p_1 = 0,297$	145,12 (145) $p_1 = 0,322$
SG-opt	5,725 (5,74) $d_1 = 0$	7,2 (7,35) $d_1 = 0,212$	10,42 (10,44) $d_1 = 0,27$	**11,81 (12,08)** $d_1 = 0,3$	**14,96 (15,07)** $d_1 = 0,3$	**18,4 (18,7)** $d_1 = 0,31$	**112 (110)** $d_1 = 0,322$
AA	**5,709 (5,74)** $c = 1$	**6,99 (7)** $c = 3,9$	**10,35 (10,58)** $c = 4,26$	**11,81 (12,2)** $c = 4,25$	15,08 (15,59) $c = 3,85$	18,5 (18,97) $c = 3,6$	112 (118) $c = 3,5$

Table 2: Job's mean (and the standard deviation of the) response time in the two-server system. Job-wise assignment. The input parameters are the same as in the Table 2.

	$\lambda = 0,05$ $\rho = 0,15$	$\lambda = 0,1$ $\rho = 0,30$	$\lambda = 0,18$ $\rho = 0,54$	$\lambda = 0,2$ $\rho = 0,60$	$\lambda = 0,23$ $\rho = 0,69$	$\lambda = 0,25$ $\rho = 0,75$	$\lambda = 0,32$ $\rho = 0,96$
RND-opt	4,64 (4,69) $p_1 = 0,25$	5,64 (5,66) $p_1 = 0,27$	8,59 (8,66) $p_1 = 0,3$	9,89 (10) $p_1 = 0,31$	12,8 (13,04) $p_1 = 0,315$	15,9 (16,12) $p_1 = 0,32$	100 (100) $p_1 = 0,322$
SG-opt	4,27 (4,24) $d_1 = 0,25$	5,11 (5,1) $d_1 = 0,29$	7,69 (7,68) $d_1 = 0,31$	8,83 (8,89) $d_1 = 0,32$	**11,4 (11,4)** $d_1 = 0,315$	14,1 (14,14) $d_1 = 0,31$	**88,6 (88,31)** $d_1 = 0,322$
AA	**4,17 (4,17)** $c = 0,75$	**5,05 (5,1)** $c = 1,2$	**7,66 (7,75)** $c = 1,25$	**8,79 (8,83)** $c = 1$	**11,4 (11,4)** $c = 1,1$	**14 (14,14)** $c = 1,15$	86,7 (86,6) $c = 1$

Table 3: Mean (and the standard deviation of the) response time of a batch in the two-server system. Job-wise assignment. See Table 1 for input parameters, Table 2 for values of policies' parameters.

	$\lambda = 0,05$ $\rho = 0,15$	$\lambda = 0,1$ $\rho = 0,30$	$\lambda = 0,18$ $\rho = 0,54$	$\lambda = 0,2$ $\rho = 0,60$	$\lambda = 0,23$ $\rho = 0,69$	$\lambda = 0,25$ $\rho = 0,75$	$\lambda = 0,32$ $\rho = 0,96$
RND	5,39 (5,1)	6,58 (6,24)	10,1 (9,53)	11,7 (11,13)	15,2 (14,49)	19 (18,16)	120 (110)
SG	5 (4,58)	6,05 (5,66)	9,1 (8,54)	10,5 (9,9)	**13,5 (12,65)**	**16,5 (15,49)**	**100 (94)**
AA	**4,84 (4,69)**	**5,98 (5,75)**	**9,06 (7,75)**	**10,3 (9,85)**	13,6 (13)	16,7 (15,8)	100 (96)

Table 4: Job's mean (and the standard deviation) response time in the 128-server system. Batch-wise assignment. Batches arrive according to the two-phase hyper-exponential process with the rate $\lambda = (0,8873/\lambda_1 + 0,1127/\lambda_2)^{-1}$. Mean size is $\mathsf{E}[B] = 4,125$, mean job size is $\mathsf{E}[X] = 1$. The offered load is $\rho = 4,125\lambda$. The RND-opt (and SG-opt) policy parameters values are given in the Table 6.

	$\lambda_1 = 0,0355$ $\lambda_2 = 0,0045$ $\rho = 0,0825$	$\lambda_1 = 0,0709$ $\lambda_2 = 0,0090$ $\rho = 0,165$	$\lambda_1 = 0,1479$ $\lambda_2 = 0,0188$ $\rho = 0,34375$	$\lambda_1 = 0,2218$ $\lambda_2 = 0,0282$ $\rho = 0,515625$	$\lambda_1 = 0,3227$ $\lambda_2 = 0,0409$ $\rho = 0,75$
RND-lb	481 (800)	488 (806)	579 (927)	792 (3873)	1620 (2450)
RND-opt	401 (583)	389 (557)	483 (600)	700 (849)	1510 (1673)
SG-lb	483 (800)	491 (806)	572 (905)	780 (1183)	1550 (2280)
SG-opt	400 (574)	390 (548)	478 (600)	**686 (825)**	**1510 (2049)**
AA	**298 (361)** $c = 14$	**338 (400)** $c = 13$	**473 (575)** $c = 10$	**703 (819)** $c = 7$	1560 (2280) $c = 6$

(persistently) the best choice when $\rho < 0,5$; but the SG-opt policy does better under the heavy traffic. An important observation from the data in the Tables 4 and 5 is that the performance of the RND and SG policies is sensitive to the choice of the parameter's values: load balancing leads to visibly larger mean response times. But in large heterogeneous systems searching for good values of $(p_1, \ldots, p_N)$ (and especially $(d_1, \ldots, d_N)$) may not be feasible. Thus, in general, the performance achieved under the RND and SG policies with load balancing is the only one, which can be guaranteed. From this point of view the ranking of the policies becomes independent of the system's load: the AA policy is uniformly the best choice among the three policies. Moreover it does not depend on the system's size: for a system with parallel single-server queues it requires estimation[19] of the single parameter.

[19] Yet no "rule of thumb" can be suggested for its estimation and simulation always has to be engaged.

Table 5: Job's mean (and the standard deviation) response time in the 128-server system. Job-wise assignment. See Table 4 for the input parameters, Table 6 for the policies parameters values.

	$\lambda_1 = 0,0355$ $\lambda_2 = 0,0045$ $\rho = 0,0825$	$\lambda_1 = 0,0709$ $\lambda_2 = 0,0090$ $\rho = 0,165$	$\lambda_1 = 0,1479$ $\lambda_2 = 0,0188$ $\rho = 0,34375$	$\lambda_1 = 0,2218$ $\lambda_2 = 0,0282$ $\rho = 0,515625$	$\lambda_1 = 0,3227$ $\lambda_2 = 0,0409$ $\rho = 0,75$
RND-lb	123 (272)	147 (302)	229 (424)	374 (640)	982 (1612)
RND-opt	111 (232)	131 (251)	209 (346)	354 (520)	912 (1183)
SG-lb	134 (332)	157 (361)	241 (500)	383 (721)	899 (1483)
SG-opt	111 (232)	130 (249)	**207** (**346**)	351 (510)	**830** (**1049**)
AA	**97**, **7** (**187**) $c = 6$	**126** (**228**) $c = 4$	**208** (**331**) $c = 2,3$	**348** (**510**) $c = 1,7$	857 (1225) $c = 1,3$

Table 6: Parameters of the RND-opt (and SG-opt) policies for batch-wise and job-wise assignments for the values of the offered load ρ considered in the Tables 5 and 6: batch-wise | job-wise.

	$\rho = 0,0825$	$\rho = 0,165$	$\rho = 0,34375$	$\rho = 0,515625$	$\rho = 0,75$
p_2	0,001|0,001	0,001|0,001	0,003|0,029	0,040|0,018	0,043|0,042
p_3	0,068|0,067	0,025|0,013	0,052|0,062	0,063|0,054	0,077|0,071
p_4	0,119|0,127	0,101|0,111	0,102|0,106	0,116|0,120	0,116|0,122
p_5	0,159|0,167	0,157|0,166	0,157|0,154	0,152|0,162	0,145|0,152
p_6	0,186|0,183	0,204|0,207	0,201|0,189	0,180|0,187	0,176|0,173
p_7	0,213|0,214	0,233|0,234	0,222|0,213	0,212|0,216	0,206|0,204
p_8	0,246|0,239	0,270|0,265	0,261|0,247	0,235|0,243	0,236|0,233

UNRELIABLE SERVERS

Assume that servers in the queues are unreliable in the following sense. Each server is subject to breakdowns independently of whether it is busy of not and how long it has been busy. A breakdown makes the queue inoperative for a while. Thus each queue alternates between operative (up) and inoperative (down) periods. Once the "down" period is over, job's processing is resumed at the point where it was interrupted. It is assumed that the "up" and "down" periods constitute an alternating renewal process with the known CDFs $U(x)$ and $D(x)$ having means $\mathsf{E}[U]$ and $\mathsf{E}[D]$.

The list of available routing policies in the presence of breakdowns remains the same: RND, SG and AA. For Poisson arrivals and under the batch-wise assignment the arrival process to each queue with RND routing remains Poisson. And under some assumptions on $S(x)$, $U(x)$ and $D(x)$ it may be feasible to find the N-tuple $(p_1, p_2, \ldots, p_N)$ minimizing the mean response time of an arbitrary job. In general, the best parameters' values can be found only through simulation. Even though the structures[20] of the SG policy (4) and of the AA policy (Algorithm 1) do not take into account the presence of breakdowns, numerical experiments show that, when applied as is, both outperform the RND-opt policy. Nevertheless, one can go further and make various amendments in the AA policy, which make it aware of servers' unreliability. One of them is to independently[21] sample (for each queue) "up" and "down" periods (from $U(x)$ and $D(x)$) and to update the values of $\tilde{u}_i^{(n)}$ only if the queue n is in the "up" period. In order to demonstrate the performance of this modified AA policy we again consider the first example, and additionally assume that servers are unavailable for processing 10% of time. Let the time between the breakdowns (for each server) be exponentially distributed with the mean $\mathsf{E}[U] = 30$ and the repair time be exponentially distributed with the mean $\mathsf{E}[D] = 30/9$. Thus the fraction of time each server is in the "up" state is equal to $\mathsf{E}[U]/(\mathsf{E}[U] + \mathsf{E}[D]) = 0,9$. The whole system is stable if and only if $\rho = \lambda \mathsf{E}[B]\mathsf{E}[X]/(0,9 \sum_{i=1}^{2} v^{(i)}) \approx 3,33\lambda$. The values of the mean and the standard deviation of the mean response time depending on the offered load ρ for the batch-wise assignment are given in the Table 7. It can be seen that the presence of the breakdowns does not affect the ranking of the policies. If the markovian assumptions are dropped, according to our numerical experiments, the conclusion remains unchanged.

SUMMARY

The arrival-aware policy (Algorithm 1) always leads to better performance than the RND-opt policy. If the system's load is not high, then the AA policy is also better than the SG policy (with the (close to) optimal densities). Whenever the RND and SG policies adopt load balancing, the AA policy outperforms both of them across all values of the system's load. The performance improvement comes for free: the AA policy (as well as the SG

[20] But implicitly the presence of breakdowns is taken into account through the values of $(d_1, d_2, \ldots, d_N)$ and c.

[21] We note that if the online information about the "up" and "down" periods is available to the dispatcher, then the sampling is not needed and $\tilde{u}_i^{(n)}$ are updated only during the "up" periods of the queue n. This makes the AA policy outperform the SG policy (and RND-opt) for all $0 < \rho < 1$. The AA policy allows one also to take into account (in Bayesian framework) the uncertainty in the information about the "up" and "down" periods (for example, their means).

Table 7: Job's mean (and the standard deviation of the) response time in the system with two unreliable servers, which are available 90% of time. Batch-wise assignment. The input parameters are the same as in the Table 1. The offered load is $\rho \approx 3,33\lambda$.

	$\lambda = 0,05$	$\lambda = 0,1$	$\lambda = 0,18$	$\lambda = 0,2$	$\lambda = 0,23$	$\lambda = 0,25$	$\lambda = 0,32$
	$\rho \approx 0,166$	$\rho \approx 0,333$	$\rho \approx 0,6$	$\rho \approx 0,666$	$\rho \approx 0,766$	$\rho \approx 0,833$	$\rho \approx 1,066$
RND-opt	7,012 (7,2)	9,603 (10)	16,71 (17)	20,20 (21)	29,15 (30)	40,56 (41)	—
	$p_1 = 0,001$	$p_1 = 0,157$	$p_1 = 0,260$	$p_1 = 0,276$	$p_1 = 0,289$	$p_1 = 0,305$	—
SG-opt	7,011 (7,2)	8,810 (9,2)	14,01 (14,6)	**16,60 (17)**	**23,38 (24)**	**32,62 (34)**	—
	$d_1 = 0,001$	$d_1 = 0,25$	$d_1 = 0,31$	$d_1 = 0,315$	$d_1 = 0,32$	$d_1 = 0,325$	—
AA	**6,762 (7)**	**8,569 (8,9)**	**14,00 (14,5)**	16,66 (17)	23,53 (24)	32,63 (34)	—
	$c = 5,7$	$c = 5,5$	$c = 4,7$	$c = 4,4$	$c = 4,0$	$c = 3,8$	—

policy) can be implemented in the dispatcher at very limited costs. In general the gain depends on the properties of the job/batch size distribution, number of queues, service rates: numerical experiments shows that it increases with the decrease of variability of the involved random quantities. Unfortunately the AA policy lacks, so far, the theoretical support. On the other hand, it allows various modifications (for example, accounting for servers' unreliability) and, for a system with $N \geq 2$ queues, requires estimation of the single parameter (compared to the RND and SG policy, which require $N - 1$ parameters). This all taken together makes the AA policy an appealing routing rule for batch-arrival unobservable dispatching systems.

REFERENCES

Anselmi, J. 2017. Asymptotically optimal open-loop load balancing. Queueing Systems. Vol. 87. No. 3-4. Pp. 245–267.

Anselmi, J. and B. Gaujal. 2011. The price of forgetting in parallel and non-observable queues. Perform. Eval. Vol. 68. No. 12. Pp. 1291–1311.

Arian, Y., Levy, Y. 1992. Algorithms for generalized round robin routing. Oper. Res. Lett. Vol. 12. Pp. 313–319.

Bell, C. H., Stidham S. 1983. Individual versus social optimization in the allocation of customers to alternative servers. Management Science. Vol. 29. No. 7. Pp. 831–839.

Cooper, R.B. 1981. Introduction to queueing theory. Elsevier North Holland, New York.

Feng, H., Misra, V., Rubenstein, D. 2005. optimal state-free, size-aware dispatching for heterogeneous $M/G/$-type systems. Performance Evaluation. Vol. 62. No. 1-4. Pp. 475–492.

Harchol-Balter, M., Crovella, M.E., Murta, C.D. 1999. On choosing a task assignment policy for a distributed server system. Journal of Parallel and Distributed Computing. Vol. 59. Pp. 204–228.

Hordijk, A., van der Laan, D. A. 2004. Periodic routing to parallel queues and billiard sequences. Math. Method. Oper. Res., 2004. Vol. 59. No. 2. Pp. 173–192.

Hyytiä, E., Aalto, S. 2011. To Split or not to Split: Selecting the Right Server with Batch Arrivals. Operations Research Letters. Vol. 41. No. 4. Pp. 325–330.

Ibaraki, T.I., Katoh, N. 1988. Resource allocation problems. Cambridge: MIT Press.

Konovalov, M.G., Razumchik, R.V. 2020. A simple dispatching policy for minimizing mean response time in non-observable queues with SRPT policy operating in parallel. Communications of the ECMS. Vol. 34. No. 1. Pp. 398–402.

Konovalov, M.G., Razumchik, R.V. 2018 Improving routing decisions in parallel non-observable queues. Computing. Vol. 100. No. 10. Pp. 1059–1079.

Lingenbrink, D., Iyer K. 2017. Optimal Signaling Mechanisms in Unobservable Queues with Strategic Customers. In Proceedings of the 2017 ACM Conference on Economics and Computation, New York, NY, USA. Pp. 347–347.

Mengistu, T.M., Che, D. 2019. Survey and taxonomy of volunteer computing. ACM Comput. Surv. Vol. 52. No. 3. Art. ID 59.

Sano, S., Miyoshi, N. 2000. Applications of m-balanced sequences to some network scheduling problems. Discrete Event system: Analysis and Control. Proceedings of the 5th Workshop on Discrete Event Systems. Pp. 317–325.

van Ommeren, J. C. W. 1990. Simple approximations for the batch-arrival $M^X/G/1$ queue. Operations Research. Vol. 38, No. 4. Pp. 678–685.

AUTHOR BIOGRAPHIES

MIKHAIL KONOVALOV is a Doctor of Sciences in Technics and holds position of the principal scientist at the Institute of Informatics Problems of the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences (FRC CSC RAS). His research activities are focused on adaptive control of random sequences, modelling and simulation of complex systems. His email address is `mkonovalov@ipiran.ru`.

ROSTISLAV RAZUMCHIK received his Ph.D. degree in Physics and Mathematics in 2011. Since then, he has worked as the leading research fellow at the Institute of Informatics Problems of the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences (FRC CSC RAS). His current research activities are focused on queueing theory and its applications for performance evaluation of stochastic systems. His email address is `rrazumchik@ipiran.ru`