Evaluating Rare Events in Mission Critical Dispatching Systems

Esa Hyytiä* and Rhonda Righter†
Department of Computer Science*, University of Iceland
Department of Industrial Engineering and Operations Research†, University of California Berkeley

***Abstract*—Dispatching systems, where jobs are routed to servers immediately upon arrival, appear frequently in parallel computing systems. With a dynamic dispatching policy, the system is generally analytically intractable and performance evaluation is carried out by means of Monte Carlo simulations. A typical performance metric is the mean response time which is often easy to estimate. In contrast, we consider systems where events generating costs are extremely rare. In our reference system, jobs have deadlines for waiting time. When deadlines are loose when compared to the system's load, novel rare event simulation techniques must be applied. We consider both conditioning and importance sampling to this end. The proposed techniques are illustrated in numerical examples, where we discover interesting performance relationships among the classical dispatching policies; Random split (RND), Round-robin (RR), Join-the-shortest-queue (JSQ) and Least-work-left (LWL).**

I. Introduction

Many frequently used services in today's Internet, such as Facebook and Google search, are realized using a set of parallel servers. Similarly, in cloud and super computing a vast amount of information is processed in parallel. Multicore CPUs are an everyday commodity in embedded systems. In daily life, we have parallel servers in supermarket queues, call centres (help desks), and at airports (check in counters). Many services, such as those in above, can tolerate a relatively high load as occasional failure to respond quickly is generally accepted. In contrast, in *mission critical systems*, such as control systems for autonomous driving, or emergency services within healthcare, there is more at stake than, e.g., if a response to a web query is slightly delayed. In mission critical systems, (computing) jobs or tasks may have strict target deadlines that should be met with a very high probability. Consequently, the average load cannot be too high without jeopardizing the basic operation.

The above systems can be modelled as dispatching systems, which consists of a set of parallel servers and a dispatcher that assigns jobs immediately upon arrival to one of the available servers. Scalability and strict real-time requirements often deter other load balancing methods (e.g., a common queue).

In this paper, we study heterogeneous dispatching systems and focus on the deadline cost structure, where each job has a maximum allowable waiting time, denoted by τ. If the maximum waiting time is exceeded, a deadline violation occurs and the system incurs a unit cost. However, deadline violations are rare if the job arrival rate λ is small relative to the number of servers and τ. We refer to this type of dispatching system as a mission critical system, as the total service rate (processing capacity) is assumed to be set sufficiently high to ensure that almost all jobs receive service in time.

Due to the infinite, often uncountable, state space, the exact performance analysis of a dispatching system is generally beyond analytical means, and one resorts to Monte Carlo simulations. However, deadline violations being *rare events* in mission critical systems means that straightforward simulation efforts are in vain as reasonable confidence intervals would require prohibitively long simulation runs.

Splitting is one common rare-event simulation technique [1], [2], [3], where multiple independent simulation trajectories are launched when the process is about to enter an interesting area in the state-space (e.g., a queue length increases beyond some value). Consequently, more simulation effort is put into trajectories that are more likely to include the sought rare events. Our first method, *paired path protocol* (PPP), is based on conditioning and is similar to splitting, but we explicitly include (and appropriately weight) trajectories from each "level" (cf. stratified sampling). Hence, one trajectory in each sample *is guaranteed to enter* the most interesting region. Another standard technique is *importance sampling*, where the underlying probability mass is modified in such a way that the interesting events appear more frequently [4], [3], [5]. Our second method is an application of importance sampling.

The main contribution of this paper is the design of the aforementioned efficient simulation methods to evaluate the performance of a dispatching policy in a mission critical system. We also provide interesting numerical observations for the relative performance between some well-known dispatching policies at the limit when the offered load tends to zero.

The rest of the paper is organized as follows. First, in Section II we introduce formally the dispatching system considered in this paper. In Section III, we discuss the reasons for the impracticality of straightforward simulation when the system's load is light. Motivated by this, in Sections IV and V we then develop two advanced simulation techniques that are subsequently illustrated in Section VI. Section VII concludes the paper.

II. Modeling and Performance Metrics

We make the standard assumptions. Jobs arrive according to a Poisson process with rate λ, job sizes are i.i.d., $X_i \sim X$, parallel servers are heterogeneous with service rates $\nu_1, \ldots, \nu_\kappa$, and the jobs are routed using a dispatching policy

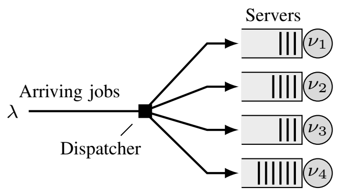

Fig. 1. Dispatching system with $\kappa = 4$ parallel servers.

α immediately upon arrival. The jobs incur a cost if they do not start service by time τ, which is referred to as the *deadline* for the (in-queue) waiting time [6]. Our performance metric is the *mean cost per job*, i.e., the deadline violation probability, denoted by η.

A busy period starts when a job arrives to an empty system and continues until all servers are idle again. We can write (cf., renewal reward)

$$\eta = \frac{\mathrm{E}[C]}{\mathrm{E}[N]},$$

where random variables C and N denote the incurred costs and the number of jobs during a busy period, respectively. The task of the dispatching policy is to assign jobs to servers so as to minimize η.

In general, we assume that the *offered load*,

$$\rho = \frac{\lambda\,\mathrm{E}[X]}{\sum \nu_i},$$

is small, and consequently, deadline violations are rare with any reasonable policy α.

We note that a real system may also process low priority jobs in the background. Our assumption is that *preemptive priority scheduling* (see, e.g., [7]) at each server makes the high priority jobs immune to possible low priority jobs. This allows us to omit them from the model and cost structure, and focus our attention on the high priority jobs that are served in the first-come-first-server (FCFS) order.

With a static dispatching policy, the system decomposes into κ independent M/G/1 queues that can be analyzed separately.[1] However, with dynamic dispatching policies, such as join-the-shortest-queue (JSQ), the system becomes analytically intractable and one must resort to simulations.

A straightforward Monte Carlo simulation of m busy periods yields a sequence of (total costs, number of jobs) pairs,

$$(C_1, N_1), (C_2, N_2), \ldots, (C_m, N_m)$$

where the (C_i, N_i) are i.i.d. $(C_i, N_i) \sim (C, N)$, and

$$\hat{\eta} = \frac{\sum_{i=1}^{m} C_i}{\sum_{i=1}^{m} N_i}$$

provides an estimate of η. When λ is small, (C_i, N_i) is often $(0, 1)$.

[1]The waiting time distribution in the M/G/1 queue is available in closed form when the service times are, e.g., exponentially distributed or constant.

III. Rare events with deadlines

In this section, we discuss the challenges that a straightforward simulation faces under low load. This motivates the development of advanced simulation techniques later in Sections IV and V.

As the first job of a busy period receives service immediately, a busy period must consist of at least two jobs, $N \geq 2$, before a deadline violation may occur. In practice, with many dynamic policies, the number of jobs needed for a deadline violation to occur is more than the number of servers. Intuitively, this means that deadline violations under low load happen only when unusually many jobs arrive in a burst.

When $\rho \to 0$, most jobs see an empty system, $\mathrm{E}[C] \to 0$, $\mathrm{E}[N] \to 1$ and $\eta \to 0$ with any dispatching policy α. In this sense, every α is "optimal" at this limit. However, we may want to know, e.g., *how fast* $\mathrm{E}[C] \to 1$. Moreover, when comparing two dispatching policies, α_1 and α_2, the interesting quantity is, e.g., the performance ratio

$$\theta = \frac{\eta_2}{\eta_1}.$$

It is unclear what the limiting value of θ might be as $\lambda \to 0$. In fact, it turns out that the simulation noise about the $\mathrm{E}[C]$ becomes an issue, whereas a small uncertainty in $\mathrm{E}[N]$ is unimportant.

Lemma 1 *Suppose that $C = 0$ when $N < k$ and define*

$$Z \triangleq \mathrm{P}\{N \geq k\} \cdot (C \mid N \geq k) = p \cdot C', \tag{1}$$

where C' corresponds to a sample from a more interesting subset where $N \geq k$ and C is more likely to be positive. Then

$$\mathrm{E}[Z] = \mathrm{E}[C], \tag{2}$$

$$\mathrm{V}[Z] = p\,\mathrm{V}[C] - (1-p)\,\mathrm{E}[C]^2. \tag{3}$$

Proof: For the mean (2), we have

$$\mathrm{E}[C] = (1-p)\,\mathrm{E}[C \mid N < k] + p\,\mathrm{E}[C \mid N \geq k] = \mathrm{E}[Z].$$

Similarly for the variance,

$$\begin{aligned}\mathrm{V}[C] &= \mathrm{E}[C^2] - \mathrm{E}[C]^2 \\ &= (1-p)\,\mathrm{E}[C^2 \mid N < k] + p\,\mathrm{E}[C^2 \mid N \geq k] - \mathrm{E}[C]^2 \\ &= (1/p)\,\mathrm{E}[Z^2] - \mathrm{E}[C]^2 \\ &= \frac{1}{p}\,\mathrm{V}[Z] + \frac{1-p}{p}\mathrm{E}[C]^2\end{aligned}$$

which reduces to (3). $\square$

According to Lemma 1, taking samples of Z instead of C yields an unbiased estimator for $\mathrm{E}[C]$ with a strictly smaller variance when $p < 1$. In other words, one should exclude the samples with no costs systematically if possible.

Conversely, as $\mathrm{V}[Z] \geq 0$, we have an elementary lower bound for the squared coefficient of variation,

$$\mathrm{SCV}(C) = \frac{\mathrm{V}[C]}{\mathrm{E}[C]^2} \geq \frac{1-p}{p}.$$

Hence, when most samples incur no costs, i.e., when p is small, $\mathrm{SCV}(C)$ explodes and estimating the mean becomes difficult.

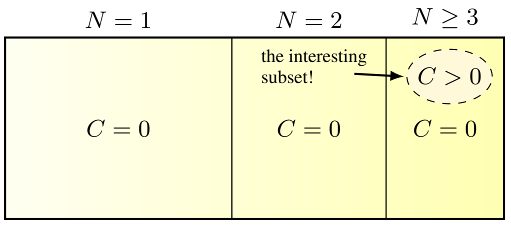

Fig. 2. Partitioning the sample space in an example case with two servers and JSQ. Busy periods with $N < 3$ jobs are guaranteed to incur no costs.

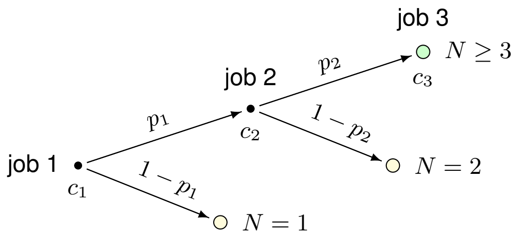

Fig. 3. Paired path protocol (PPP) samples all branches in the above tree simultaneously by conditioning on the next job arriving before the system becomes idle.

A. Increasing the odds

Suppose a busy period starts with a size x_1 job, and that the dispatching policy α assigns the first job to server j. The service time of the first job is $s_1 = x_1/\nu_j$. If λ is small, there is a high chance that no other jobs arrive during time s_1 and $C = 0$, which we wanted to avoid. We can proceed two ways:

- **Idea 1:** Condition on more jobs arriving during the busy period. This means that at least one more job arrives during the service time s_1 of the first job starting the busy period. After that, we can condition again that the third job arrives before the system becomes idle, etc.
- **Idea 2:** Adjust the arrival rate from the nominal λ to a bit higher λ^* so that it is more likely that a busy period consists of more than one job.

Note that the latter is an application of importance sampling (IS) [4], [3], [5]. The idea is effectively the same as in [8], where the authors argued that for the classical M/M/1 queue, the optimal simulation approach for the mean time to reach an excessively large backlog of N jobs is obtained by interchanging the arrival rate λ and the service rate μ. The GI/GI/m queue in the same setting was later considered in [9]. The motivation for large N was buffer overflows, e.g., in routers. For more details, see [3]. In our case, N would be much smaller and correspond to a good or acceptable quality-of-experience (QoE). That is, we implicitly assume a light system load and very small deadline violation probabilities.

Next we discuss both approaches.

IV. Paired path protocol based on conditioning

Let us start with the conditioning approach yielding the *paired path protocol* (PPP). The mean costs incurred in a busy period, $\mathrm{E}[C]$, can be obtained by conditioning on the number of jobs in the busy period N, e.g.,

$$\mathrm{E}[C] = \mathrm{E}[C \,|\, N = 1]\,\mathrm{P}\{N = 1\} + \mathrm{E}[C \,|\, N > 1]\,\mathrm{P}\{N > 1\}.$$

As discussed, the first term is often zero or is easy to determine in general, and therefore, the simulation effort should focus on estimating the second term. Given a Poisson arrival process with rate λ, we can proceed as follows:

- Suppose that a busy period starts with a size x_1 job, the dispatching policy assigns the job to server j, and the system incurs cost c_1.
- Let s_1 be the corresponding service time, $s_1 = x_1/\nu_j$, where ν_j is the service rate of server j.
- Then we have two possible cases:
  1) With probability $q_1 = e^{-\lambda s_1}$ the next job arrives later, and this busy period ends with a total cost of c_1 (zero if deadline cost structure). We get a weighted sample $(q_1, c_1, 1) =$ (weight,costs,jobs).
  2) With probability $p_1 = 1 - e^{-\lambda s_1}$, the busy period has more than one job. Instead of using an Exp(λ) distribution, we generate a random arrival time for *the second job* from a truncated Exp-distribution,

     $$T_1 \sim \text{Exp}(\lambda, s_1),$$

     thus ensuring (conditioning) that the busy period has at least one more job. Let c_2 denote the costs it incurs. The future jobs in this busy period, if any, arrive according to a Poisson process with rate λ. This yields a weighted sample $(p_1, c_1 + c_2 + \ldots + c_n, n)$, where n denotes the number of jobs in the busy period.
- The combined estimated sample data is then

$$\begin{aligned}(\hat{c}, \hat{n}) &= q_1(c_1, 1) + p_1(c_1 + c_2 + \ldots + c_n, n) \\ &= (c_1 + p_1(c_2 + \ldots + c_n), 1 + p_1(n - 1)).\end{aligned}$$

This simulation gives samples $(\hat{c}_i, \hat{n}_i)$, $i = 1, \ldots, m$, of the costs and the number of jobs during busy periods.

In the above, the second case was conditioned on at least one more job arriving in the busy period, $N \geq 2$. Suppose that *with a given (reasonable) dispatching policy* α the first $k - 1$ jobs are routed to idle servers and hence never incur costs, and therefore we would like to include sample paths with k jobs or more, see Fig. 2. For example, with JSQ we can set $k = \kappa + 1$, where κ denotes the number of servers, as the first κ jobs are guaranteed to receive service immediately and no deadline violations are possible. Next we generalize the procedure to take simultaneous samples of busy periods with $N = 1, 2, \ldots, (k - 1)$ and $N \geq k$ jobs with arbitrary k.

The idea is illustrated in Fig. 3. For jobs $j = 2, \ldots, (k - 1)$ we recursively define their inter-arrival times, t_j, and s_j, the remaining time before the system would be idle if no further jobs arrive after the arrival of the jth job of the busy period, by conditioning on $N \geq k$. To this end, the inter-arrival time after jobs $1, \ldots, (k - 1)$ are drawn from the appropriate *truncated*

TABLE I
SUBROUTINES FOR THE SIMULATION ALGORITHMS.

Function	**Returns:**
$\alpha(\mathbf{z},\mathbf{x})$:	the server chosen when job $\mathbf{x}$ arrives in state $\mathbf{z}$
$\mathsf{backlog}(\mathbf{z})$:	the remaining time until the system becomes idle
$\mathsf{cost}(\mathbf{z},\mathbf{x},j)$:	the immediate cost if job $\mathbf{x}$ is assigned to server j
$\mathsf{add}(\mathbf{z},\mathbf{x},j)$:	the state after assigning job $\mathbf{x}$ to server j
$\mathsf{process}(\mathbf{z},t)$:	the state after processing present jobs for time t

Exp-distributions, $\mathrm{Exp}(\lambda, s_j)$, and we have the thinning factor for each sample of a busy period with $N \geq k$ jobs,

$$p_1 \cdots p_{k-1} = \prod_{j=1}^{k-1} (1 - e^{-\lambda s_j}).$$

Similarly, the busy periods with n jobs, $n < k$, have weight

$$p_1 \cdots p_{n-1} \cdot q_n = \left(\prod_{j=1}^{n-1} (1 - e^{-\lambda s_j}) \right) \cdot e^{-\lambda s_n}.$$

The process of moving along the upper-most branch in Fig. 3 we call *priming* the system state so that non-zero costs can be observed more frequently. PPP resembles the well-known splitting technique, where multiple sub-trajectories are launched when the process enters an interesting region. The main difference is that in our case the splitting is explicit and each sample includes simultaneous trajectories of busy periods with $1, 2, \ldots, (k-1)$ and at least k jobs.

A. Simulation algorithm

Let us next describe a complete algorithm that at each round generates simultaneous samples of costs for cases where N is $1, 2, \ldots, k-1$ and $N \geq k$.

Let $\mathbf{x}$ denote a job. In the general case, in addition to the size information, jobs may have such attributes as job- or class-specific holding costs or deadlines, and this information must be included in $\mathbf{x}$. Then we let $\mathbf{z}_j$ denote the state of server j, defining jobs present and their remaining service times, and let $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_\kappa)$ denote the state of the system. The simulation algorithm utilizes the five subroutines listed in Table I. We let the $\mathsf{backlog}(\mathbf{z}_j)$ be the current total scaled work at server j, i.e., the time at which it would become idle if no new arrivals were routed to it. We let $\mathsf{backlog}(\mathbf{z})$ be the time at which all servers would become idle if there were no more arrivals. We note that with FCFS and a single deadline τ for all jobs, it is sufficient to let x denote the size of a job, and the immediate deadline violation cost is

$$\mathsf{cost}(\mathbf{z}, x, j) = \mathbf{1}(\mathsf{backlog}(\mathbf{z}_j) > \tau).$$

When estimating $\mathrm{E}[N]$, we have $\mathsf{cost}(\mathbf{z}, x, j) = 1$, whereas for the mean waiting time $\mathrm{E}[W]$, $\mathsf{cost}(\mathbf{z}, x, j) = \mathsf{backlog}(\mathbf{z}_j)$. In practice, we collect all relevant statistics at the same time.

The actual simulation algorithm in pseudo code is given in Table II. Note that the algorithm ensures that k jobs arrive in the main branch, and that each sample consists of k sub-samples (sample path trajectories). With $k = 1$ the procedure reduces to the basic Monte Carlo simulation of a busy period.

TABLE II
PAIRED PATH PROTOCOL (PPP) ALGORITHM BASED ON CONDITIONING.

```
Initialize:
  1) Initialize state $\mathbf{z}$                    (empty system)
  2) $\hat{n}_b = \hat{c}_b = c = 0$, $w = 1$
For $n = 1, \ldots, k-1$:
  1) Draw a new job $\mathbf{x}$                      (size and possible attributes)
  2) With $j = \alpha(\mathbf{z}, \mathbf{x})$:
       • $c = c + \mathsf{cost}(\mathbf{z}, \mathbf{x}, j)$
       • $\mathbf{z} = \mathsf{add}(\mathbf{z}, \mathbf{x}, j)$
  3) $s = \mathsf{backlog}(\mathbf{z})$
  4) $q = e^{-\lambda s}$
  5) $\hat{n}_b = \hat{n}_b + wq \cdot n$
  6) $\hat{c}_b = \hat{c}_b + wq \cdot c$
  7) $w = w \cdot (1 - q)$
  8) Draw $t \sim \mathrm{Exp}(\lambda, s)$           (time to next arrival)
  9) $\mathbf{z} = \mathsf{process}(\mathbf{z}, t)$   (state upon arrival)
For $n = k, k+1, \ldots$:
  1) Draw a new job $\mathbf{x}$                      (size and possible attributes)
  2) With $j = \alpha(\mathbf{z}, \mathbf{x})$:
       • $c = c + \mathsf{cost}(\mathbf{z}, \mathbf{x}, j)$
       • $\mathbf{z} = \mathsf{add}(\mathbf{z}, \mathbf{x}, j)$
  3) Draw $t \sim \mathrm{Exp}(\lambda)$              (time to next arrival)
  4) If $t > \mathsf{backlog}(\mathbf{z})$:           (the busy period ends)
       • $\hat{n}_b = \hat{n}_b + w \cdot n$
       • $\hat{c}_b = \hat{c}_b + w \cdot c$
       • Return $(\hat{n}_b, \hat{c}_b)$
  5) $\mathbf{z} = \mathsf{process}(\mathbf{z}, t)$   (state upon arrival)
```

Proposition 1 *The paired path protocol approach reduces the variance in the deadline violation cost estimate when $k \geq 2$.*

Proof: Referring to Fig. 2, the probability that a sample incurs costs, $C > 0$, increases, which, according to Lemma 1 reduces the variance. □

V. IMPORTANCE SAMPLING BY BIASED ARRIVAL RATE

In this section, we use a biased arrival rate to make (e.g.) deadline violations to occur more frequently in our samples of busy periods. We can use a biased arrival rate λ^*

1) during the service time s_1 of the first job, $\Delta = s_1$,
2) until the first deadline violation occurs (or the busy period ends), or
3) for the whole busy period B^*, $\Delta = B^*$.

In all cases, the busy period B^*, with increased arrival rate for some time Δ, is on average longer than the busy period with the original arrival rate λ (assumed to be small). We note that this is an application of the well-known importance sampling (IS) method [4], [8], [9]. The idea is illustrated in Fig. 4 and the appropriate weights to debias the samples can be derived as follows:

- Let n be the number of jobs that arrive *with the biased arrival rate* λ^* during time Δ (excluding the initial job starting the busy period).
- The probabilities that there are n arrivals during time Δ with arrival rates λ and λ^* are

$$p = \frac{(\lambda\Delta)^n}{n!} e^{-\lambda\Delta}, \text{ and } p^* = \frac{(\lambda^*\Delta)^n}{n!} e^{-\lambda^*\Delta},$$

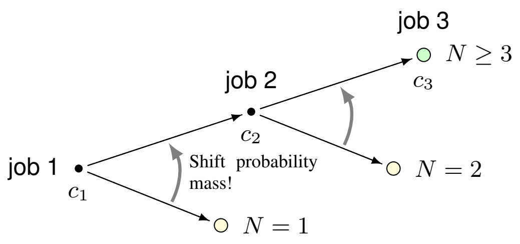

Fig. 4. Importance sampling by biasing the arrival rate λ shifts the probability mass towards the "upper branch" thus making the rare events more probable.

and hence the *importance ratio* (or likelihood ratio) is

$$\beta(n) = \frac{p}{p^*} = \left(\frac{\lambda}{\lambda^*}\right)^n e^{(\lambda^* - \lambda)\Delta}.$$

- Simulating samples of busy periods with a biased arrival rate gives samples of (i) the number of jobs in a busy period, N^*, and (ii) the total costs incurred, C^*. Both quantities must be "debiased", so that

$$\mathrm{E}[N] = \mathrm{E}[\beta(n)\, N^*],$$
$$\mathrm{E}[C] = \mathrm{E}[\beta(n)\, C^*].$$

That is, simulations of a busy period with biased arrival rates give samples of form $(\beta(n)\, N^*, \beta(n)\, C^*)$, where the multiplication with the importance ratio $\beta(n)$ ensures that the estimates (for the mean) are unbiased. This holds with any λ^*, but not all choices are good. The common criterion is to minimize the variance of the estimate. Here, however, we have two (or more) quantities that need to be estimated.

The estimate for the mean cost per job is the same as before,

$$\hat{\eta} = \frac{C_1 + \ldots + C_m}{N_1 + \ldots + N_m} = \frac{\beta(n_1) C_1^* + \ldots + \beta(n_m) C_m^*}{\beta(n_1) N_1^* + \ldots + \beta(n_m) N_m^*}.$$

Note that biasing may become useful both when $\rho \to 1$ and $\rho \to 0$, depending on the cost metric.

In case 1), where $\Delta = s_1$, one can choose λ^* so that $\lambda^* \Delta = k$ for some constant k, defining the expected number of new jobs during the service time of the first job. In some sense, this is a probabilistic version of the paired path protocol discussed earlier in Section IV. Then $p^* = k^n/(n! e^k)$ and

$$\beta(n) = \left(\frac{k}{\lambda\Delta}\right)^n e^{\lambda\Delta - k}.$$

Case 2), where Δ is equal to the first deadline violation (or some similar rare event), is similar to those considered in [8] and [9]. The main distinction is that here the rare event takes only a few jobs arriving sufficiently quickly, whereas in [8], [9] the load is moderate and the rare event (buffer overflow) is assumed to require a large number of jobs.

In case 3), where $\Delta = B^*$ and $n = N^* - 1$, one should store the 3-tuple, (λ^*, N^*, C^*), as that allows "extrapolation" of the results to any other λ (as long as the corresponding ρ is also stable!). For example, with a stable M/D/1 queue, the probability of having a busy period with a single job is

$$\mathrm{P}\{X < A\} = e^{-\rho} > 1/e \approx 0.368.$$

TABLE III
IS SIMULATION ALGORITHM BASED ON BIASING THE ARRIVAL RATE.

```
Initialize and the first job:
  1) Initialize state $\mathbf{z}$                         (empty system)
  2) $T = 0$, $N = 1$
  3) Draw a new job $\mathbf{x}$                           (size and possible attributes)
  4) With $j = \alpha(\mathbf{z}, \mathbf{x})$:
       • $C = \mathsf{cost}(\mathbf{z}, \mathbf{x}, j)$
       • $\mathbf{z} = \mathsf{add}(\mathbf{z}, \mathbf{x}, j)$
  5) $\Delta = \mathsf{backlog}(\mathbf{z})$
  6) Draw $t \sim \mathrm{Exp}(\lambda^*)$                    (time to next arrival)
While $t < \Delta - T$:
  1) $\mathbf{z} = \mathsf{process}(\mathbf{z}, t)$
  2) $T = T + t$
  3) $N = N + 1$
  4) Draw a new job $\mathbf{x}$                           (size and possible attributes)
  5) With $j = \alpha(\mathbf{z}, \mathbf{x})$:
       • $C = C + \mathsf{cost}(\mathbf{z}, \mathbf{x}, j)$
       • $\mathbf{z} = \mathsf{add}(\mathbf{z}, \mathbf{x}, j)$
  6) Draw $t \sim \mathrm{Exp}(\lambda^*)$                    (time to next arrival)
Remaining time with the biased $\lambda^*$:
  1) $\mathbf{z} = \mathsf{process}(\mathbf{z}, \Delta - T)$           (no arrivals until time $T = \Delta$)
  2) $\beta = (\lambda/\lambda^*)^{N-1} e^{-(\lambda - \lambda^*)\Delta}$      (as $n = N - 1$)
  3) Draw $t \sim \mathrm{Exp}(\lambda)$                      (next arrival at $T + t$)
While $t < \mathsf{backlog}(\mathbf{z})$:
  1) $\mathbf{z} = \mathsf{process}(\mathbf{z}, t)$                 ($T = T + t$)
  2) $N = N + 1$
  3) Draw a new job $\mathbf{x}$                           (size and possible attributes)
  4) With $j = \alpha(\mathbf{z}, \mathbf{x})$:
       • $C = C + \mathsf{cost}(\mathbf{z}, \mathbf{x}, j)$
       • $\mathbf{z} = \mathsf{add}(\mathbf{z}, \mathbf{x}, j)$
  5) Draw $t \sim \mathrm{Exp}(\lambda)$                      (time to next arrival)
Unbias:
  1) Return $(\beta N, \beta C)$
```

Thus, informally, it seems to be relatively safe to extrapolate towards lower arrival rates as sample sets, even with $\rho^* \approx 1$, have a large fraction of samples of busy periods with one or a few jobs. The opposite, however, can be more challenging when $\rho \approx 1$, because typical busy periods are very long, and such long busy periods basically never exist for ρ^* moderate or small. Therefore even a large but finite sample set of (simulated) busy periods may not be sufficiently representative.

The simulation algorithm in pseudo code with a biased arrival rate for Case 1) is given in Table III. Note that we stop updating the accumulated time T when that information is no longer needed. The pseudo code for Cases 2) and 3) is very similar and omitted for brevity.

A. Splitting

The splitting technique [1], [2], [3] could also be applied in our setting. This is illustrated in Fig. 5, where multiple trajectories are launched when the second job arrives. In splitting, one does not change the dynamics of the stochastic system, but simply replicates the process when it approaches a region of interest. In our setting, with a small λ, most busy periods consist of a single job, incur no costs, and there is no chance to split to begin with! For this reason, PPP and IS are more attractive techniques for us. However, when the offered load increases splitting also becomes a viable option.

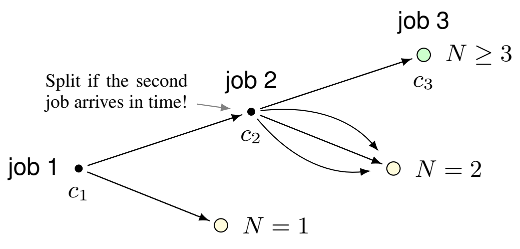

Fig. 5. Splitting launches multiple trajectories when approaching the interesting region thus increasing the chances that a trajectory could eventually incur a positive cost. However, when λ is very small, most trajectories immediately continue "downwards".

VI. Numerical examples

In this section, we utilize the rare event simulation techniques developed in the previous sections. We consider $\kappa = 2, 3, 4$ identical exponential servers and the deadline cost structure. First we study the variance reduction (Section VI-B), and then illustrate the performance of different policies near the light traffic limit as $\rho \to 0$ (Section VI-C).

A. Dispatching policies

Let us start by introducing our example dispatching policies. The random (Bernoulli) split is the easiest policy to analyze:

- **Random split (RND)** assigns each job uniformly at random to κ servers, thus balancing the load.

With exponential service times, the system decomposes into independent M/M/1 queues, and we have an explicit result for the waiting time distribution,

$$\eta = \mathrm{P}\{W > \tau\} = \rho e^{-(\mu-\lambda)\tau},$$

where (λ, μ) are the queue-specific quantities. When ρ is small, $\lambda \ll \mu$, we have a linear relationship,

$$\eta \approx \rho e^{-\mu\tau} = \rho e^{-\delta}, \tag{4}$$

where $\delta = \mu\tau$ so that the system's performance is characterized by two dimensionless quantities, (ρ, δ). Similarly it is easy to obtain results for heterogeneous κ-server systems with an exponential job size and load balancing random split dispatching policy.

We consider also four other dispatching policies:

- **Round-robin (RR)** assigns jobs sequentially in a fixed order, $1, 2, \ldots, \kappa, 1, 2, \ldots$. In some sense, RR is clearly stateful. However, with Poisson arrivals, the system decomposes to κ identical Erl(κ, λ)/G/1 queues as the interval-arrival times are now Erlang-distributed. Effectively, RR *regulates* the arrival process, which tends to improve the system's performance.
- **Join-the-shortest-queue (JSQ)** chooses the queue with the least number of jobs, so the state is the number of jobs at each server (*number-aware*). Ties can be resolved in favor of the server with a smaller index. Equivalently, ties could be resolved, e.g., randomly (cf. lack of memory).
- **Least-work-left (LWL)** chooses the queue with the shortest backlog, so the state is the backlog, or normalized work, which requires size information for jobs (*size-aware*). Again, ties are resolved in favor of the server with the smaller index.
- **Hybrid-work-left (HWL)** chooses the queue with the shortest backlog, if that avoids a deadline violation. Otherwise the queue *with the longest backlog* is chosen. The motivation is that this increases the chances for the *next job* to receive service in time.

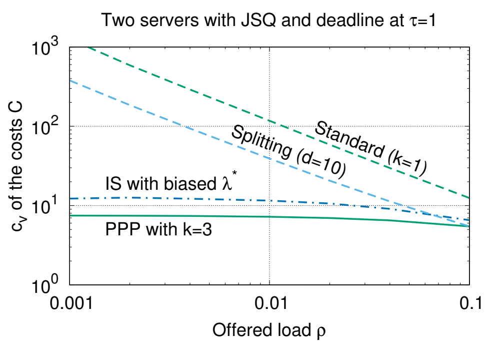

Fig. 6. Coefficient of variation of the random quantity C (the number of deadline violations in a busy period) with different simulation approaches.

Both JSQ and LWL are greedy or selfish policies where the (expected) waiting time of the new job is minimized. Therefore, they can be expected to work rather well also with the deadline cost structure. HWL is marginally less myopic than LWL, and is expected to work well under low load. It is very similar to the Dead-k policy, introduced in [6], which routes a job bound to miss its deadline to a fixed subset of servers, thus also improving the chances that the next job will receive service in time. However, when the load is high, HWL, like Dead-k, is prone to stability problems. For details on the general merits and weaknesses of RR, JSQ and LWL, see, e.g., [10] and the references therein. For more advanced policies, tailored specifically for deadline cost structures, see, e.g., [6], [11], [12].

B. Variance Reduction

Let us first study the variance reduction with different approaches. To this end, consider the random variable C corresponding to the costs incurred in a busy period. Moreover, we fix the dispatching policy to JSQ and assume two servers, $\kappa = 2$, exponential service times, $\mathrm{E}[X] = 1/\mu = 1$, and a target deadline of $\tau = 1$, so that $\delta = \mu\tau = 1$.

With splitting, we launch $d = 10$ trajectories if the second job arrives before the service of the first job finishes. The size of the second job, X_2, is drawn independently for each trajectory. With IS, we set λ^* so that on average two more jobs arrive during the service time of the first job, $\lambda^* = 2/s_1$. With PPP, we use $k = 3$ so that each sample includes simultaneously samples of busy periods with $N = 1$, $N = 2$ and $N \geq 3$ jobs, and the corresponding cost C is an appropriately weighted sum of the three scenarios.

Fig. 6 depicts the coefficient of variation, $c_v = \sigma_C/\mathrm{E}[C]$, with (i) standard Monte Carlo simulation, (ii) splitting, (iii) IS and (iv) PPP. We see that both PPP and IS reduce the variance efficiently, and in particular, the coefficient of variation, c_v, tends to a constant value as $\rho \to 0$. In contrast, with standard Monte Carlo simulation and splitting, c_v diverges (as expected). We note that, at the cost of computation time, the splitting parameter d could be increased as ρ decreases. This, however, would lead to an unfair comparison because increasing d is equivalent to performing multiple simulations. (In fact, aggregating 10 samples of the standard Monte Carlo simulation yields basically the same results as with splitting!) Results are similar for the other policies and are omitted for brevity.

C. Deadline violations when $\rho \to 0$

Given that the PPP works well, and the variance is clearly under control for all $\rho > 0$, we will utilize it next to study the performance of the different dispatching policies when $\rho \to 0$. The behavior of many queueing systems in the limit as $\rho \to 0$, known as the *light traffic limit*, can be determined analytically [13], and thus it serves as a good reference point for our simulation techniques (which are usable also when ρ is small but strictly positive, as in the assumed mission critical systems).

Consider first the same two-server system. With RND, (4) reduces to $\eta = \rho/e$. The dynamic policies, RR, JSQ, LWL and HWL, can be evaluated using simulation. The numerical results are depicted in Fig. 7 (left) in log-log scale with solid lines. On the x-axis is the offered load ρ, and the y-axis corresponds to the deadline violation probability. The numerical results are based on 10^9 simulated busy periods with $k = 3$ so that at least 3 jobs arrive in every sample busy period.

Next we fit functions of the form $A\rho^r$ using the method of least squares to match the smallest three data points. By considering the slopes, we observe that the performance with RND behaves linearly near the origin, $r \approx 1$, in agreement with (4), whereas the dynamic dispatching policies converge quadratically, $r \approx 2$. When ρ is very small, the performance with LWL and HWL is identical. At $\rho = 0.1$, HWL is marginally better than LWL. Approximately, the numerical results give us

$$\eta_{\text{RR}} \approx \frac{3}{2}\rho^2, \qquad \eta_{\text{JSQ}} \approx \frac{3}{4}\rho^2, \qquad \eta_{\text{LWL}} \approx \frac{3}{4e}\rho^2, \tag{5}$$

suggesting that $\eta_{\text{RR}}/\eta_{\text{JSQ}} \to 2$ and $\eta_{\text{LWL}}/\eta_{\text{JSQ}} \to 1/e$ as $\rho \to 0$. In Fig. 8 (left), we have depicted the relative performance of policy α to JSQ, $\eta_\alpha/\eta_{\text{JSQ}}$, as a function of ρ. The solid lines correspond to $\tau = 1$ and the dashed lines to $\tau = 1/2$ and $\tau = 2$. With RR, all three curves overlap, suggesting that the ratio is insensitive to τ, whereas with LWL a higher τ yields a greater difference relative to JSQ. In particular, the numerical results suggest a more general relationship as $\rho \to 0$:

$$\frac{\eta_{\text{RR}}}{\eta_{\text{JSQ}}} \to 2 \quad \text{and} \quad \frac{\eta_{\text{JSQ}}}{\eta_{\text{LWL}}} \to e^{\delta}.$$

Figs. 7 (middle) and (right) depict the simulation results with $\kappa = 3, 4$ servers. With $\kappa = 3$, we simulated 10^9 busy periods with $k = 4$, and the results with $\kappa = 4$ are based on 10^9 busy periods with $k = 6, \ldots, 8$. As before, the performance with LWL is the same as with HWL in the limit as $\rho \to 0$. Around $\rho = 0.1$, HWL is marginally better.

Applying again the least squares method, we observe that the exponent r for ρ with the dynamic policies is equal to the number of the servers, $\eta \propto \rho^\kappa$, as expected, in contrast to RND, where the relationship is always linear, $\eta \propto \rho$.

Figs. 8 (middle) and (right) show the relative deadline performance ratio to JSQ for $\kappa = 3, 4$ servers. The solid curves correspond to $\tau = 1$, and dashed curves to $\tau = 1/2$ and $\tau = 2$. Comparing first RR to JSQ, we observe an interesting general limiting relationship,

$$\lim_{\rho \to 0} \frac{\eta_{\text{RR}}}{\eta_{\text{JSQ}}} = \kappa!,$$

which indeed seems to be insensitive to the deadline τ as the green solid and dashed curves corresponding to RR overlap. In contrast, the relative performance with LWL and with HWL improves as a function of τ (or δ). This means that size information becomes more valuable as the deadline threshold τ increases. Combining the numerical results with $\kappa = 2, 3, 4$ servers with $\tau = 0.5, 1, 2$, it seems that in general we have:

Observation 1 *With κ identical exponential servers, it holds for the deadline violation probabilities that*

$$\lim_{\rho \to 0} \frac{\eta_{RR}}{\eta_{JSQ}} = \kappa!$$
$$\lim_{\rho \to 0} \frac{\eta_{LWL}}{\eta_{JSQ}} = e^{-\delta(\kappa-1)}$$

where $\delta = \tau\mu$ and $\eta_\alpha = \mathrm{P}\{W > \tau \mid \alpha\}$ is the deadline violation probability with dispatching policy α.

D. Mean waiting time when $\rho \to 0$

Next we look at how the *mean waiting time*, $\mathrm{E}[W]$, behaves as $\rho \to 0$. As with the deadline metric, the majority of the samples incur no costs, $W = 0$, and therefore the advanced simulation techniques are required. The numerical results are depicted in Figure 9. Note that only the HWL policy depends on the deadline parameter $\tau = 1/2, 1, 2$, which is a superfluous parameter for this metric. With the other three policies, RR, JSQ and LWL, the mean waiting time is independent of the value of τ.

Observation 2 *With κ identical exponential servers, we have*

$$\lim_{\rho \to 0} \frac{\mathrm{E}[W \mid RR]}{\mathrm{E}[W \mid JSQ]} = \kappa!$$
$$\lim_{\rho \to 0} \frac{\mathrm{E}[W \mid LWL]}{\mathrm{E}[W \mid JSQ]} = \frac{1}{\kappa}$$

where $\mathrm{E}[W \mid \alpha]$ *denotes the mean response time with dispatching policy α.*

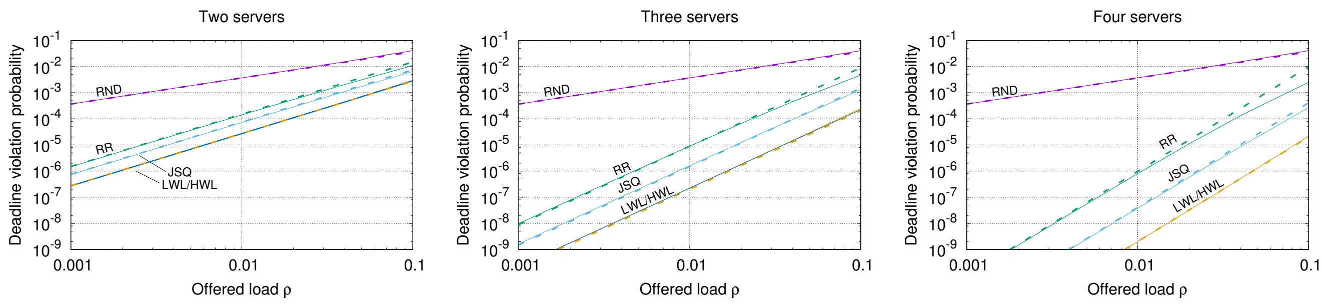

Fig. 7. Deadline violation probability under low load with $\kappa = 2, 3, 4$ servers, $\tau = 1$.

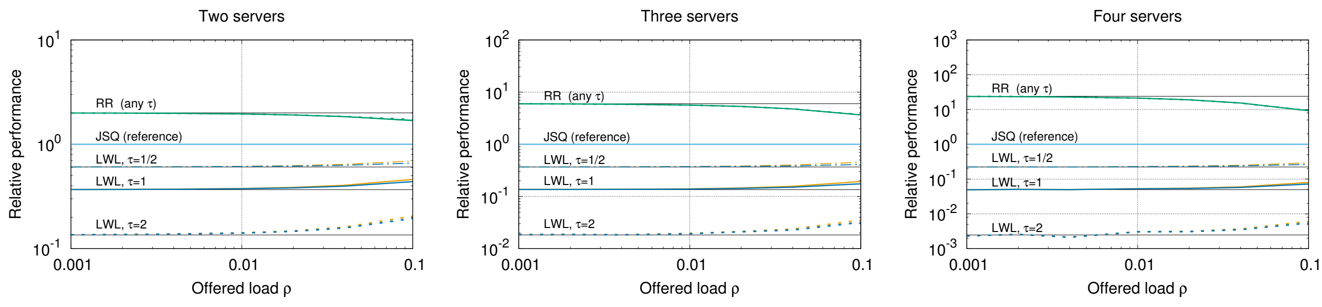

Fig. 8. Relative performance with respect to deadline violations with $\kappa = 2, 3, 4$ servers. Solid lines correspond to $\tau = 1$ and dashed-dotted lines to $\tau = 0.5$ and dashed lines $\tau = 2$. Cases $\tau = 0.5$ and $\tau = 2$ are clearly visible only for LWL/HWL.

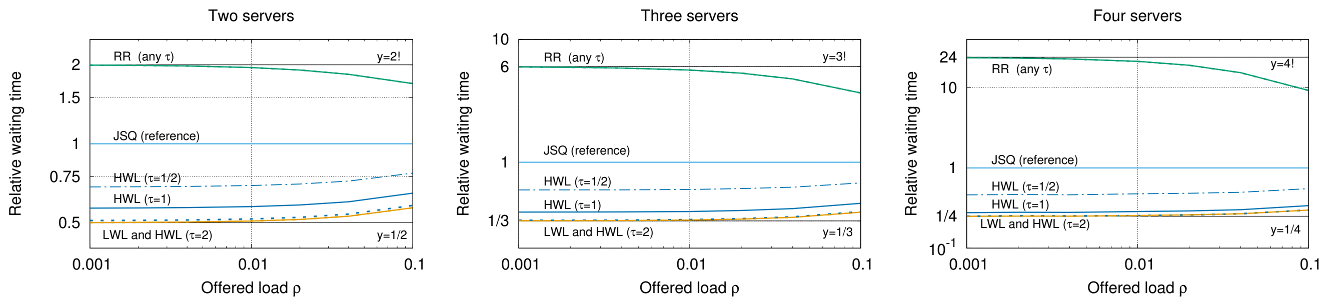

Fig. 9. Relative performance with respect to mean waiting time with $\kappa = 2, 3, 4$ servers. Solid lines correspond to $\tau = 1$ and dashed-dotted lines to $\tau = 0.5$ and dashed lines $\tau = 2$ with HWL. The other policies are insensitive to τ.

E. Intuition for the observed behavior when $\rho \to 0$

The results on the relative performance in the limit as $\rho \to 0$ can be argued by considering the most likely sequence of events leading to a positive cost. As ρ is very small, one ends up considering arrival sequences of $\kappa + 1$ jobs [13].

Let us consider first RR. The probability that the $(\kappa + 1)^{\text{st}}$ job is routed to a busy server with RR is

$$\begin{aligned} p_{\text{RR}} &= \text{P}\{\text{the next job arrives before job 1 departs}\}^{\kappa} \\ &= \left(\frac{\lambda}{\lambda+\mu}\right)^{\kappa}. \end{aligned}$$

With JSQ, the $(\kappa + 1)^{\text{st}}$ job has to wait only if all κ servers are busy, i.e.,

$$\begin{aligned} p_{\text{JSQ}} &= \text{P}\{\text{no job departs before the } (\kappa+1)^{\text{st}} \text{ job arrives}\} \\ &= \frac{\lambda}{\lambda+\mu} \cdot \frac{\lambda}{\lambda+2\mu} \cdots \frac{\lambda}{\lambda+\kappa\mu} = \frac{\lambda^{\kappa}}{\prod_{i=1}^{\kappa}(\lambda+i\mu)}. \end{aligned}$$

Due to lack of memory, the conditional waiting time distribution of the $(\kappa + 1)^{\text{st}}$ job with RR and JSQ is exponential, $W_{\kappa+1} \sim \text{Exp}(\mu)$. Hence, in the limit as $\lambda \to 0$, RR is $\kappa!$ times worse than JSQ with respect to *any waiting time related performance metric* such as the deadline violation probability and mean waiting time. This explains the RR related results in Observations 1 and 2.

With LWL, the probability that the $(\kappa+1)^{\text{st}}$ job has to wait is the same as with JSQ, $p_{\text{LWL}} = p_{\text{JSQ}}$. However, the conditional waiting time distribution is the minimum of κ exponentially distributed random variables, i.e., $W_{\kappa+1} \sim \text{Exp}(\kappa\mu)$ for LWL.

Thus, the waiting time of the $(\kappa+1)^{\text{st}}$ job of a busy period is "served" at the rate $\kappa\mu$ with LWL, whereas with JSQ the corresponding service rate is only μ. Therefore, in the limit as $\lambda \to 0$, with respect to deadline violation probability we have

$$\lim_{\rho\to 0} \frac{\eta_{\text{LWL}}}{\eta_{\text{JSQ}}} = \frac{e^{-\kappa\mu\tau}}{e^{-\mu\tau}} = e^{-\delta(\kappa-1)},$$

and with respect to the mean waiting time,

$$\lim_{\rho\to 0} \frac{\mathrm{E}[W_{\kappa+1}\,|\,\text{LWL}]}{\mathrm{E}[W_{\kappa+1}\,|\,\text{JSQ}]} = \frac{1/(\kappa\mu)}{1/\mu} = \frac{1}{\kappa},$$

which explain the related results in Observations 1 and 2. As

$$p_\alpha \cdot \mathrm{P}\{W_{\kappa+1} > \tau \,|\, W_{\kappa+1} > 0\} \to \eta_\alpha,$$

we also have the exact light traffic results for the deadline metric (when $\rho \to 0$),

$$\begin{aligned}
\eta_{\text{RR}} &= \kappa^\kappa e^{-\delta} \rho^\kappa, \\
\eta_{\text{JSQ}} &= \frac{\kappa^\kappa e^{-\delta}}{\kappa!} \rho^\kappa, \\
\eta_{\text{LWL}} &= \frac{\kappa^\kappa e^{-\kappa\delta}}{\kappa!} \rho^\kappa,
\end{aligned}$$

which match well with the earlier numerical estimates (5) obtained for $\kappa = 2$.

Referring to Figures 8 and 9, in our example scenarios, with different numbers of servers, dispatching policies and performance metrics, the light traffic estimates are accurate only when the load is very low, $\rho < 0.01$. When $\rho > 0.01$, but still low, the performance evaluation must be carried out by means of our novel Monte Carlo simulation techniques.

VII. Conclusions

The performance of a dispatching system depends on the offered load, the number of servers (and their capacity and internal scheduling discipline), the dispatching policy and the cost metric. We focused on a deadline cost metric, where each job has a certain maximum waiting time it can tolerate. In mission critical dispatching systems, the dimensioning criteria should be such that most of the time the system works flawlessly, i.e., deadline violations are extremely rare. In such cases, it may be difficult to evaluate the (relative) performance of different dispatching policies. The main contribution of this paper is the development of two advanced simulation techniques designed for situations when the offered load is low and events generating costs are rare. In the first approach, referred to as the paired path protocol (PPP), we take samples of busy periods with at least k jobs, where k is a free parameter. In the second approach, based on importance sampling (IS), we utilize a biased arrival rate to the same end, i.e., in both cases the goal is to obtain samples with a strictly positive cost more frequently. We emphasize that the proposed simulation techniques are general in the sense that the dispatching policy, the cost structure, and the service time distribution can be arbitrary and servers heterogeneous. Moreover, they are practical whenever ρ is small, not just in the limit as $\rho \to 0$. In fact, the behavior of many queueing systems and performance metrics in this *light traffic* limit can be deduced analytically [13].

We also demonstrated our approaches by comparing five dispatching policies (RND, RR, JSQ, LWL and HWL) for the basic routing problem with exponential service times, $\kappa = 2, 3, 4$ identical servers, and deadline cost structure. First, the deadline violation probability with the four dynamic policies converges to zero at a rate proportional to ρ^κ, whereas with RND the rate is proportional to ρ. At this limit, RR is $\kappa!$ times worse than JSQ, which in turn is $e^{\delta(\kappa-1)}$ times worse than LWL/HWL, where $\delta = \mu\tau$ is a dimensionless quantity characterizing the deadline in terms of the mean service time. The extensive simulation experiments suggest that the light-traffic limit results are accurate when the offered load is very small, $\rho < 0.01$ in our case. Above that, novel Monte Carlo simulations are needed.

Acknowledgements

This work was supported by the Academy of Finland in the FQ4BD project (grant no. 296206) and by the University of Iceland Research Fund in the RL-STAR project. It is also based upon work from COST Action CA15127 RECODIS supported by COST (European Cooperation in Science and Technology).

References

[1] M. Villén-Altamirano and J. Villén-Altamirano, "RESTART: A method for accelerating rare event simulation," in *Queueing Performance and Control in ATM*, J.W.Cohen and C. D. Pack, Eds., Copenhagen, Denmark, Jun. 1991, pp. 71–76, ITC-13 Workshops.

[2] Z. Haraszti and J. K. Townsend, "The theory of direct probability redistribution and its application to rare event simulation," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 9, no. 2, pp. 105–140, 1999.

[3] G. Rubino and B. Tuffin, *Rare Event Simulation Using Monte Carlo Methods*. Wiley Publishing, 2009.

[4] S. M. Ross, *Introduction to Probability Models*, 7th ed. Academic Press, 2000.

[5] J. Kuhn and M. Mandjes, "Efficient simulation of tail probabilities in a queueing model with heterogeneous servers," *Stochastic Models*, vol. 34, no. 2, pp. 239–267, 2018.

[6] E. Hyytiä and R. Righter, "Routing jobs with deadlines to heterogeneous parallel servers," *Operations Research Letters*, vol. 44, no. 4, pp. 507–513, 2016.

[7] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*, 4th ed. John Wiley & Sons, 2008.

[8] S. Parekh and J. Walrand, "A quick simulation method for excessive backlogs in networks of queues," *IEEE Transactions on Automatic Control*, vol. 34, no. 1, pp. 54–66, Jan. 1989.

[9] J. S. Sadowsky, "Large deviations theory and efficient simulation of excessive backlogs in a GI/GI/m queue," *IEEE Transactions on Automatic Control*, vol. 36, no. 12, pp. 1383–1394, Dec. 1991.

[10] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013.

[11] E. Hyytiä, R. Righter, and J. Virtamo, "Meeting soft deadlines in single- and multi-server systems," in *28th International Teletraffic Congress (ITC'28)*, Würzburg, Germany, Sep. 2016.

[12] E. Hyytiä, R. Righter, O. Bilenne, and X. Wu, "Dispatching fixed-sized jobs with multiple deadlines to parallel heterogeneous servers," *Performance Evaluation*, vol. 114, pp. 32–44, Sep. 2017.

[13] M. I. Reiman and B. Simon, "Open queueing systems in light traffic," *Mathematics of Operations Research*, vol. 14, no. 1, pp. 26–59, 1989.