Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

RARE EVENT SIMULATION OF DELAY IN PACKET SWITCHING NETWORKS USING DPR-BASED SPLITTING

Zsolt Haraszti

Ericsson Radio Systems AB
S-12625 Stockholm, SWEDEN

J. Keith Townsend

Center for Advanced Computing and Communication
Department of Electrical & Computer Engineering
North Carolina State University
Raleigh, NC, 27695-7914 U.S.A.

ABSTRACT

Rare event simulation using splitting has been shown to provide significant speed-up for large classes of problems, especially when queue length distribution is of primary interest. However, choice of the control parameters is much less straightforward in cases where splitting is applied to systems in which the target event is delay, rather than packet loss. In this paper we propose a control strategy for splitting that allows computationally efficient analysis of very low delay threshold probabilities which typically occur in communication networks. A different technique is required for delay because unlike the cell or packet loss case, the target event (delay) and the prerequisite condition that leads to a rare delay event (a full buffer) do not coincide temporally. We demonstrate the technique by using it to measure delay probabilities in three examples: a simple ATM multiplexer, a queueing system with multiple traffic classes, and a tandem queueing network with tagged and background traffic.

1 INTRODUCTION

Accurate design and characterization of high speed communications networks requires estimating the small probabilities associated with rare cell loss and excessive delay. Discrete Event Simulation (DES) is a powerful tool in estimating performance of networks, but its straightforward application has the drawback of requiring prohibitive execution time when estimating rare event probabilities.

Importance sampling (IS) is a technique that can speed up the simulation of rare events under certain conditions (see, e.g., Heidelberger 1995 and references therein). Although IS has the potential for order-of-magnitude speed-up, considerable *a priori* knowledge is required about the system being simulated. In particular, *conventional* IS techniques require careful modification of the underlying probability measures in the system, so that increased occurrence of target events is realized. In addition, conventional IS is plagued with the problem of decreasing likelihood ratios.

In contrast to conventional IS, *trajectory splitting*, (or also referred to as *importance splitting*, or simply *splitting*) achieves increased occurrence of the rare target event by generating several independent sub-trajectories from less rare states. The original idea (see, e.g., Kahn and Harris 1951; Hammersley and Handscomb 1964; Bayes 1970) was first developed into a refined simulation technique and become widely known due to Villén-Altamirano and Villén-Altamirano (1991) and Villén-Altamirano et al. (1994). They call their version of this approach RESTART. Recently, a number of papers have dealt with the analysis (see, e.g., Glasserman et al. 1996a; Glasserman et al. 1997; Glasserman et al. 1996b), the refinement (e.g., Haraszti and Townsend 1998; Görg and Schreiber 1996; Villén-Altamirano 1998) and the application of splitting to practical rare-event problems (see, e.g., Haraszti and Townsend 1998; Haraszti et al. 1998; Görg and Fuß 1999).

Since splitting does not modify the probability measures of the underlying random processes in the system, it has the potential to be applicable to more complex systems. When applying splitting to a specific system, the two chief questions that need to be answered are: 1) *when* to split and 2) *how many* sub-trajectories to create when splitting. As we will discuss in the paper, the latter question can be answered in a rather problem-independent way, but the former question remains problem-specific.

To achieve efficient splitting, the splitting conditions have to be chosen in accordance with the rare event of interest and its prerequisite conditions, i.e., the dominant paths that lead to the rare event. In cases when such dominant paths are highly correlated with a *single* system parameter, the splitting conditions can be defined as the event when the system parameter exceeds (crosses) one of a set of pre-defined thresholds. For most existing splitting techniques it is assumed that 1) the rare event states reside in

the inner-most subset and 2) the system does not cross more than one threshold between two consecutive observations.

The splitting simulation technique derived from the theory of Direct Probability Redistribution (DPR) and introduced in Haraszti and Townsend (1998) removes two chief limitations of existing splitting techniques. First, DPR-based splitting allows arbitrary state-space partitions. The splitting condition is associated with the event of departing one subset and entering another, and transition is allowed between any two pairs of subsets. This feature of DPR allows for the case where the transitions that occur during system evolution skip one or more intermediate thresholds. This avoids the need for setting up the thresholds during problem formulation to ensure that no multiple crossing can occur. Second, under DPR-based splitting the rare event of interest can overlap multiple subsets. A similar extension to RESTART allowing this overlap was presented in Villén-Altamirano (1998).

Splitting has been shown to provide good speed-up and is straightforward to apply to queuing problems where excessive queue lengths or packet loss is the target event. This is because both the target event and the dominant paths are related to the queue length (see, e.g., Haraszti and Townsend 1998). Note that for these problems there is also a coincidence in time between the occurrence of the rare event and the prerequisite conditions that contribute to the rare event.

In this paper we discuss the problem of analyzing very low delay threshold probabilities that typically occur in ATM networks. Unfortunately, in contrast to the temporally coincident behavior of excessive queue length and packet loss described above, delay measurement has an inherent temporal behavior that complicates application of rare event simulation. In particular, there is a time-lag between the arrival of packets that have the potential for long delay and the departure of these same packets. Straightforward application of splitting to delay through queues with random service was presented in Görg and Fuß (1999). However, as we will show in this paper, the efficient application of splitting to more complicated queueing configurations and/or to queues with constant service times requires more complicated splitting control and the addition of auxiliary state variables, which calls for the application of DPR-based splitting.

In the remaining part of the paper, we first summarize the DPR-based splitting algorithm. Then we discuss how to construct the splitting conditions that result in computationally efficient splitting simulations of packet delay. We demonstrate the technique by using it to measure delay probabilities in three examples: a simple ATM multiplexer, a queueing system with multiple traffic classes, and a tandem queueing network with tagged and background traffic. Although ATM network examples are given in this paper, the simulation methodology for delay is directly applicable to any packet switching technology including TCP/IP.

2 OVERVIEW ON THE DPR-BASED SPLITTING ALGORITHM

Here we summarize the DPR-based splitting algorithm which we use to control splitting in the delay simulation; a more detailed description of DPR can be found in Haraszti and Townsend (1998). Let $s_i \in \mathcal{S}$, $i \in \{1, \ldots, n\}$ denote the (finite) *state-space* of the system and let the series $V_0, V_1, V_2, \ldots (V_k \in \{1, \ldots, n\})$ represent the evolution of the system, observed in successive *observation points*. It is assumed that the system model is a discrete-time Markov chain. We partition the state-space into m mutually exclusive, non-empty, indexed subsets, $\mathcal{S}_1, \ldots, \mathcal{S}_m$ ($\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_m = \mathcal{S}$). The partitioning is uniquely defined by the so-called *subset indicator function*

$$\Gamma(i) \in \{1, \ldots, m\}: \quad \Gamma(i) = j \iff s_i \in \mathcal{S}_j.$$

DPR assigns a so-called *oversampling factor*, μ_j, to each subset $\mathcal{S}_j$, $j \in \{1, \ldots, m\}$, and re-scales the steady state probability mass such that the following equation holds for every state:

$$\pi_i^{(m)} = \Phi \mu_{\Gamma(i)} \pi_i, \tag{1}$$

where $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_n\}$ and $\boldsymbol{\pi}^{(m)} = \{\pi_1^{(m)}, \ldots, \pi_n^{(m)}\}$ are the equilibrium probabilities of the original and of the m-partitioned system under DPR, respectively, and Φ is a re-normalization constant, $\Phi = 1/\sum_{i=1}^{n} \mu_{\Gamma(i)} \pi_i$. Thus, DPR ensures that states in subset $\mathcal{S}_j$ are visited relatively μ_j times more often than in the original system. By assigning high oversampling factors to those subsets that contain rare event states, relative occurrence of the rare events can be artificially increased. Expectedly, the price of increasing the steady state probabilities in some subset(s) is that the probabilities drop in other subsets, that is, the probability is being "redistributed", hence the name DPR. We call the vector $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_m\}$ the *oversampling vector*. We assume that $\mu_1 = 1$ and $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_m$. These conditions are needed for the formulation of the splitting algorithm; they can be satisfied for arbitrary partitioning without loss of generality, i.e., by reordering subset indexes according to $\boldsymbol{\mu}$, and normalizing the oversampling vector to μ_1.

DPR can be implemented using a trajectory splitting mechanism (very similar to the multi-threshold RESTART algorithm in Villén-Altamirano et al. 1994). DPR allows state transitions to arbitrary subsets and handles the case where the rare event set overlaps more than one subset. Splitting takes place whenever a transition to a higher indexed

subset occurs, i.e., whenever the system makes a transition from subset $\mathcal{S}_i$ to subset $\mathcal{S}_j$ such that $i < j$. The number of *new* sub-trajectories generated upon such a transition, Y_{ij}, is a random variable (RV) with expected value $\mathrm{E}[Y_{ij}] = \mu_j/\mu_i - 1$. When $j > i+1$, special care must be taken to maintain the proper trajectory density in the intermediate subsets, which is needed to preserve the unbiasedness of the simulation estimates. DPR addresses this issue by employing the following "*ticketing*" mechanism: At the $\mathcal{S}_i \rightarrow \mathcal{S}_j$ transition, each new sub-trajectory is assigned a *ticket*, Ω. The ticket $i < \Omega \leq j$ is the index of the "lowest" subset which the sub-trajectory is allowed to visit. Ω is selected according to the distribution:

$$\Pr\left\{\Omega = l \mid \mathcal{S}_i \rightarrow \mathcal{S}_j\right\} = \begin{cases} \frac{\mu_l - \mu_{l-1}}{\mu_j - \mu_i} & \text{if } i < l \leq j, \\ 0 & \text{otherwise.} \end{cases}$$

A sub-trajectory is terminated when it attempts to make a transition to a subset $\mathcal{S}_l$ for which $l < \Omega$. The state upon which a sub-trajectory terminates should be considered an invalid sample.

Under the above splitting scenario, every state in a subset $\mathcal{S}_j$ is visited, in the asymptotic sense, μ_j times more often than in an "unstressed" simulation. Unbiased estimates can be obtained when events are counted during the simulation with weight $1/\mu_{\Gamma(V_k)}$ in the kth observation point.

As motivated in Haraszti and Townsend (1998), the (near) optimal $\boldsymbol{\mu}$ vector is shown to be when the probabilities are redistributed such that the subset probability masses are "balanced" (i.e., nearly evenly distributed) among the subsets. To achieve this, the values in $\boldsymbol{\mu}$ have to be inversely proportional to the stationary subset probability masses of the original system. This (near) optimal $\boldsymbol{\mu}$ can be found by a simple iterative process using short, repetitive DPR simulation experiments (see, e.g., Akyamac et al. 1999). We have found that the overhead of such an exploration is a negligible fraction of the effective simulation time required to obtain the DPR estimates. The main challenge in applying DPR-based splitting to new problems is to find the appropriate $\Gamma(\cdot)$ function, which remains problem specific.

3 DELAY THRESHOLD SIMULATION USING SPLITTING

Delay is defined as the time a given packet spends in the system while passing between two reference points along its route. A generic method of measuring the delay requires that the packet is tagged with a time-stamp when it passes the first reference point, and the time-stamp is compared to the actual time when the packet reaches the second reference point. Let RV δ describe the delay observations. Typically, we are interested in the complementary cumulative distribution function (CCDF) of the delay, defined as

$$D(\tau) \stackrel{\triangle}{=} \Pr\{\delta > \tau\}.$$

For a given τ_0, $D(\tau = \tau_0)$ is called the *delay threshold probability*; a common performance parameter of communication networks which is required to be very small in real-time critical systems.

To achieve efficient simulation of delay, the subset indicator function, $\Gamma(\cdot)$, should be selected with the following two goals in mind. First, the choice of $\Gamma(\cdot)$ should produce the prerequisite conditions for rare delay events by forcing the system into longer queue lengths. Second, it should keep the artificially generated sub-trajectories "alive" long enough for the packets with high potential to exit the system and thus generate the accountable target delay events. Unfortunately, in contrast to the temporally coincident behavior of queue length and packet loss, delay measurement has an inherent temporal behavior that complicates application of splitting.

We illustrate the problem of choosing the subset indicator function for delay via an example. The example system is a discrete time, finite queue with batch arrivals and deterministic, constant service time. Constant service time makes rare event simulation of the delay tail especially challenging. Nevertheless, the solution is directly applicable to continuous time problems, systems with random service times, and also to queueing networks.

Let a_k and q_k denote the *number of arrivals* and the *queue length* at the kth observation point, respectively. Assuming the early-arrival model for slotted, discrete-time systems (according to Hunter 1983) q_k is defined as

$$q_k = \{q_{k-1} + (a_k - l_k) - 1\}^+,$$

where l_k is the number of *lost packets* in the kth slot,

$$l_k = \{q_{k-1} + a_k - K\}^+,$$

K is the size of the buffer and $\{x\}^+ = x$ if $x \geq 0$, 0 otherwise (see Figure 1). The *time-stamp* of the ith packet

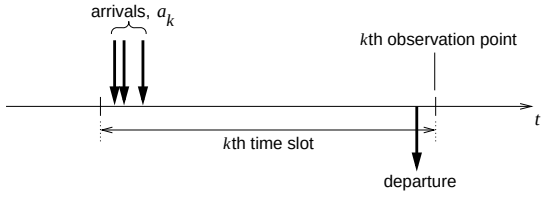

Figure 1: Relative order of arrivals, departures and observation points in an early-arrival discrete time queue.

in the queue, $\rho_{k,i}$, $i = 1, \ldots, q_k$, $q_k > 0$, can be expressed as

$$\rho_{k,i} = \begin{cases} \rho_{k-1,i+1} & \text{if } 1 \le i \le \{q_{k-1}-1\}^+ \\ k & \text{if } \{q_{k-1}-1\}^+ < i \le q_k. \end{cases} \quad (2)$$

The second condition in (2) is for newly arriving packets. A departure from the server in the kth time-slot results in a delay sample, δ_k. A departure occurs if either $q_{k-1} > 0$ (non-empty queue at the end of the previous slot) or if $q_{k-1} = 0$ but $a_k > 0$ (at least one new arrival). In the former case, the observed delay is $\delta_k = k - \rho_{k-1,1} + 1$; in the latter case $\delta_k = 1$. Figure 3 shows a sample trace of a_k, q_k and δ_k, obtained from simulation.

There is a significant time-lag between the rare event (e.g., a high-delay sample in time instant B in Figure 3) and its prerequisite condition (an earlier packet arrival when the queue was long at point A). This effect makes the simulation inefficient when indicator functions are based on either queue length or delay only. More specifically, if we use the queue length as the indicator function, i.e., $\Gamma(V_k) = q_k + 1$: sub-trajectories with large potential delay can be produced successfully, but the q_k typically drops by the time these packets would depart (see point B in Figure 3), thus terminating most of the potential sub-trajectories. Another obvious choice is $\Gamma(V_k) = \delta(k)$ (in slots without departure $\Gamma(V_k) = 1$), which has the problem that by the time the long delay is observed, the queue is typically very short. In the deterministic service time case, splitting at this point does not produce more packets which have the potential of experiencing large delay. This choice for an indicator function can also be very inefficient even if the service time is stochastic. We demonstrate the efficiency of these two choices for $\Gamma(\cdot)$ in our first numerical example, presented below.

A more efficient subset indicator function that maintains both goals simultaneously can be defined as follows. We introduce the notion of the *primary indicator function* (PIC), denoted by $T(V_k)$. The PIC is responsible for increasing the production of packets that have the potential to experience long delay. As illustrated in the example, $T(V_k) = q_k + 1$ is an efficient PIF. Since excessive queue length is the prerequisite condition of long delay, the majority of the potential packets would occur in sub-trajectories that would be the result of a splitting procedure where $\Gamma(V_k) = T(V_k)$.

Our secondary objective is to keep sub-trajectories that carry packet(s) with the potential for large delay alive until the packet(s) reach their destination (i.e., the second reference point). In our example this occurs when the packet is serviced. To artificially prolong the life of potential sub-trajectories, we introduce the following *auxiliary state variables*. Upon entering the system, let each new packet be tagged by a label, ϕ, called the *potential*, which equals the value of the current primary indicator function, defined by, $\phi = T(V_k)$. Let C_k denote the number of labeled packets in the system at the kth observation point,

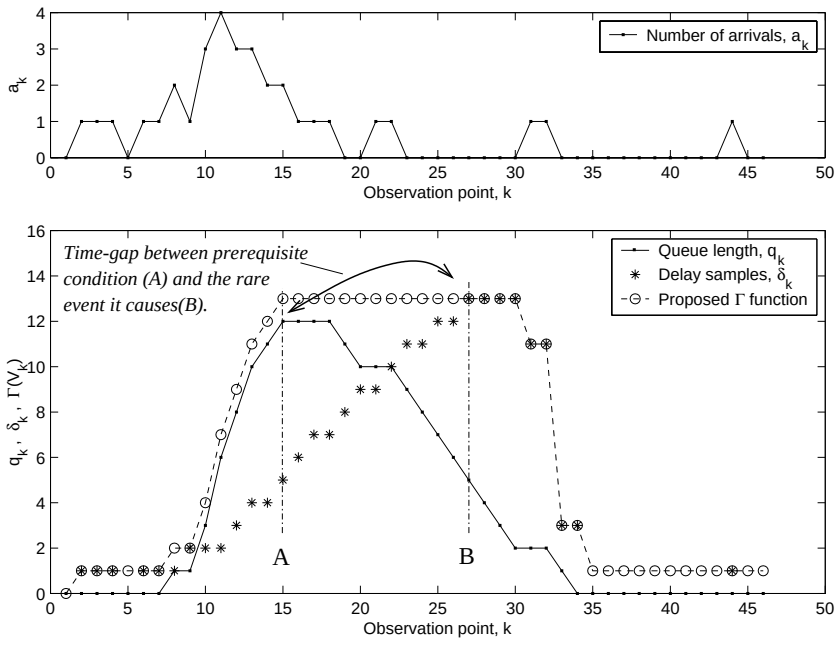

Figure 2: Sample trace of a_k, q_k and δ_k for a discrete time queue with constant 1 slot service time. The proposed splitting function is also shown.

and $\phi_{k,i}$, $i = 1, \ldots, C_k$ denote their labels. We formulate $\Gamma(V_k)$ from $T(V_k)$ and $\phi_{k,i}$ as

$$\Gamma(V_k) = \max\left(T(V_k), \bar{\phi_k}\right), \tag{3}$$

where

$$\bar{\phi_k} = \max_{i \in \{1,\ldots,C_k\}} \phi_{k,i}$$

The above formulation of $\Gamma(V_k)$ ensures that if a new tagged packet passes the first reference point while the system is in subset $\mathcal{S}_j$, the subset index will not decrease until that packet passes the second reference point, thus keeping a large number of the potential sub-trajectories alive until the critical packets result in successful delay samples. In our example, $\phi_{k,i} = q_k + 1$ for newly arrived packets, i.e.,

$$\phi_{k,i} = \begin{cases} \phi_{k-1,i+1} & \text{if } 1 \le i \le \{q_{k-1} - 1\}^+ \\ q_k + 1 & \text{if } \{q_{k-1} - 1\}^+ < i \le q_k. \end{cases}$$

In Figure 3 we also plot the corresponding values of the proposed $\Gamma(V_k)$ function. It can be seen that after reaching a high value (at point A) the indicator function sustains its value until the last packet with high potential departs the system (three slots after point B).

The proper choice of the PIF remains problem-specific, but as we will demonstrate in the next section, a relatively simple function such as the total number of packets in the system, can provide satisfactory efficiency.

4 APPLICATION EXAMPLES

4.1 Cell Delay Analysis of a Simple ATM Multiplexer

Here we consider a discrete time N×ON-OFF/D/1/K queue. This is a specific case of the previously discussed example, where the batch arrivals are generated by N identical but independent ON-OFF sources. An ON-OFF source is a special two-state Markov Modulated Bernoulli Process (MMBP) which generates a packet in each time-slot while in the *active* state, and does not generate any packet in the *idle* state. The sources are parameterized by the triplet $\{N, \alpha, B\}$, where α is the aggregated load generated by the N sources, normalized to the service rate, and B is the average burst length of a source. The server serves exactly one cell at each time slot. Such a system can be regarded as a typical building block for modeling an ATM multiplexer stage.

To estimate the delay threshold probabilities, we consider the following three alternative indicator functions:

A. $\Gamma(V_k) = q_k + 1$ (queue length based splitting)
B. $\Gamma(V_k) = \delta_k$ if $q_{k-1} > 0$, 1 otherwise (delay based splitting)
C. As in (3), using $T(V_k) = q_k + 1$

For $N = 24$, $\alpha = 0.15$, $B = 3$, and $K = 64$, Figures 3 and 4 plot the $D(\tau)$ estimates and their normalized half-size confidence intervals (95 % confidence level) obtained by the three methods. For each method, 5 or 6 short repetitive simulations were used to obtain a balancing $\boldsymbol{\mu}$ vector, and then 50 independent retrials were executed, each simulating a total number of 10^6 slots (including the generated sub-trajectories) to obtain the plotted estimates. As a reference, the numerical solution of $D(\tau)$ is also plotted in Figure 3.

The results confirm our expectation: the first two splitting strategies did not work. Their failure is visible in Figure 3: they differ considerably from the numerical result. In practical cases where numerical results are not available, a more reliable indication of failure is the excessive variance of the estimates (Akyamac et al. 1999). As can be seen in Figure 4, the confidence intervals for the queue length based and for the delay based simulation exceed 100 % for a considerable part of the tail, which is not acceptable. The splitting estimator based on our proposed splitting function coincides perfectly with the numerical results and yields very small confidence intervals (i.e., below 20 % in all cases).

Each of the three simulation experiments required approximately 1.5 minutes on a 466 MHz Pentium II PC using the FreeBSD Unix operating system (including the short iterations used for to obtain $\boldsymbol{\mu}$). In Figure 3 we also plotted the estimated speed-up for the proposed $\Gamma(V_k)$ when compared to brute-force simulation (estimated as described in Haraszti and Townsend 1998).

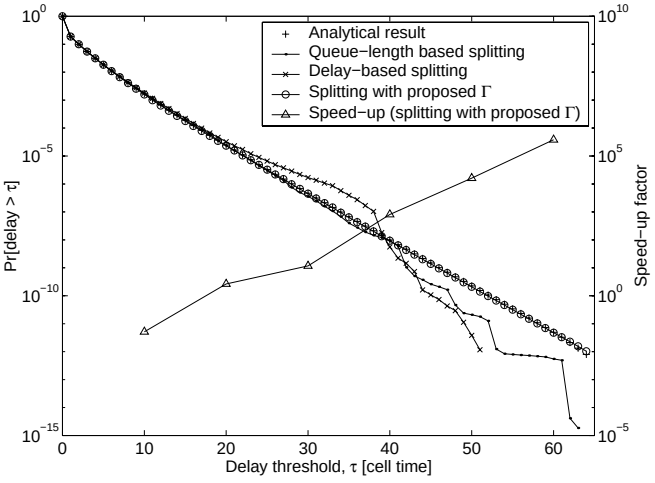

Figure 3: Delay threshold probabilities for the N×ON-OFF/D/1/K queue estimated using three alternative splitting strategies.

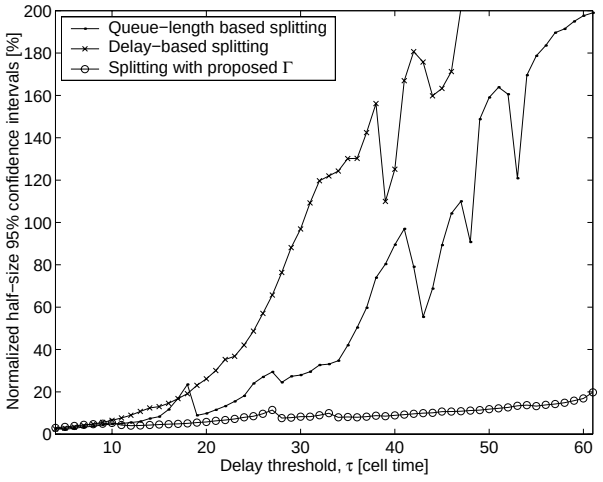

Figure 4: Estimated normalized half-size confidence intervals (95%) for the delay threshold estimation in the N×ON-OFF/D/1/K example.

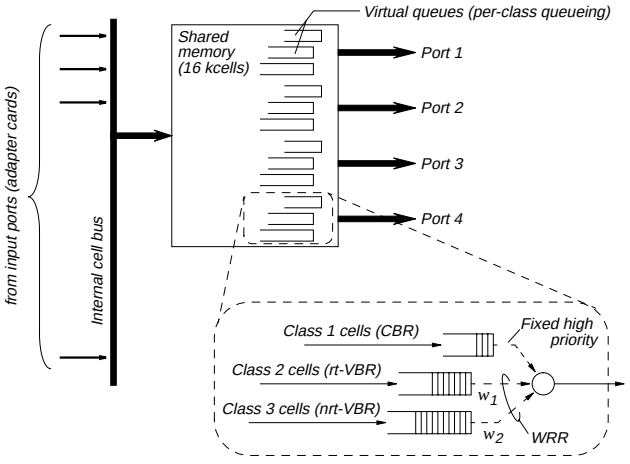

Figure 5: A portion of the shared memory ATM switch with three priority classes. Four ports share one memory board (16 kcells). CBR traffic (Class 1) has fixed *high* priority, rt-VBR and nrt-VBR (Class 2 and 3) share the remaining bandwidth using Weighted Round Robin (WRR) scheduling.

4.2 A Shared Memory LAN Switch with Multiple Priorities Using Static Priority and Weighted Round Robin Scheduling

Our second example represents a more complicated system, although in terms of delay simulation it differs only from the previous case in the sense that the service time is not deterministic.

The system is a model of an off-the-shelf *shared-memory* ATM LAN switch that uses static priority and *Weighted Round-Robin (WRR)* scheduling to provide services for three distinct service classes. The three service classes are: (Class-1) constant bit-rate (CBR) traffic with real-time requirements, (Class-2) variable bit-rate traffic with real-time requirements (rt-VBR) and (Class-3) variable bit-rate data traffic with no real-time requirements (nrt-VBR).

The switch works as follows (see Figure 5). Input sources are connected via a number of adapter cards to an internal cell bus that propagates the cells to the output ports. Four output ports share the same shared memory segment which can store up to 16 kcells (16384 cells). From the shared memory, there are three virtual queues formed for each output port, one queue for each of the three service classes. Cells in the three virtual queues compete for the link capacity of the output port in the following way. The CBR queue enjoys static high priority over the other two queues, that is, every slot when there is at least one cell in the CBR queue, the head-of-line CBR cell is forwarded to the link. The rest of the link bandwidth is shared by the rt-VBR and nrt-VBR cells according to the WRR discipline (see, e.g., Katavenis et al. 1991 for an introduction to WRR scheduling). WRR is an extension of Round-Robin which works in the following way. A counter and a weighting factor is assigned to each queue. Let us denote the former by C_2 and C_3 and the latter by w_2 and w_3, respectively. Initially, the counters are set to $C_2 = C \times w_2$ and $C_3 = C \times w_3$, where C is a cycle constant. When both counters are greater than zero, cells from the two classes can be sent in an alternating fashion, servicing at most one cell at each slot when there is no CBR cell. After sending a cell, the counter for that class is decremented by one. If the counter of one of the classes reaches zero, there can be no further cells sent from that queue, only the other queue can be serviced. When the counter value and/or the queue length has reached zero in both classes, both counters are reset to their initial values, and a new cycle begins.

We modeled the CBR traffic as N_1 independent Bernoulli cell generators. The total generated CBR load is α_1. The rt-VBR traffic is modeled by N_2 independent ON-OFF sources, with total normalized load α_2 and mean burst duration of B_2 [cells]. The same model is used for the nrt-VBR traffic, with the respective parameters N_3, α_3 and B_3. The above load values are defined as the total load of the given class normalized to the total maximum throughput of the four output ports, and all source models use the same slot size as the output ports.

We measured the delay threshold probabilities experienced by the rt-VBR and the nrt-VBR cells, respectively, as well as the occupancy distribution (probability mass function) of the shared memory segment. Figure 6 presents both results for the sample traffic configuration $N_1 = N_2 = N_3 = 8$, $\alpha_1 = 0.2$, $\alpha_2 = 0.3$, $B_2 = 3$, $\alpha_3 = 0.4$, $B_3 = 20$. For the WRR scheduler we used cycle length $C = 50$ and weights $w_2 = 60\%$ and $w_3 = 40\%$.

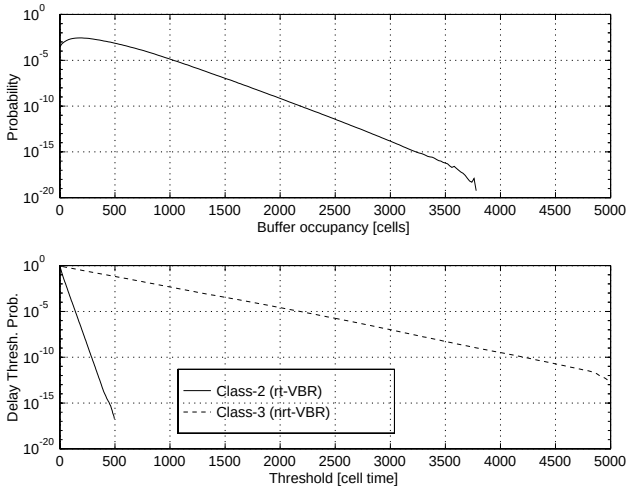

Figure 6: Shared memory occupancy (top plot), and Delay Threshold Probabilities (DTP) for Class 2 and Class 3 traffic (lower plot). ($N_1 = N_2 = N_3 = 8$, $\alpha_1 = 0.2$, $\alpha_2 = 0.3$, $B_2 = 3$, $\alpha_3 = 0.4$, $B_3 = 20$, $w_2 = 0.6$, $w_3 = 0.4$ and $C = 50$ slots).

For the memory occupancy simulation, the subset indicator function has been defined as $\Gamma(V_k) = \lfloor X_k/20 \rfloor + 1$, where X_k represents the the total number of cells in the shared memory. The rescaling was necessary to keep the number of subsets below 200. Although not presented in the plot, the relative 95% confidence interval was below 60% even at the tail of the occupancy curve.

The delay threshold measurements for the two classes were performed separately, since they required different subset indicator functions. In both cases, the number of cells in the queue of the given class has been used as a primary subset indicator function. Those cells were labeled according to the method presented in Section 3, and the extended subset indicator of (3) was applied. In all cases, 50 independent replications were executed after balancing the subset probabilities. The execution time for each of the three experiments was less than 20 minutes using a conventional UNIX workstation. The accuracy of the estimated values were very high, with the poorest accuracy in the tails of the plots. These tail values were still within 60% relative confidence interval widths (with 95% confidence level).

4.3 Delay Threshold Probabilities over Tandem Queues

Our third example extends the delay measurement technique to a tandem queuing system, presented in Figure 7. A "tagged" cell source, modeled by an ON-OFF source with normalized load, α_f, and mean burst length, B_f [cells], sends cells to the first queue. These tagged cells travel through a tandem connection of M queues. The metric of interest is the probability mass function of the total delay over the M queues. Background traffic of the queues is modeled by N additional independent ON-OFF sources, attached to each queue, as shown in the figure. Cells of the background traffic do not propagate to the next queue in the tandem chain.

We only consider the homogeneous case where each queue has the same buffer size of K cell slots and the same background load. The background load is parameterized by N, the total background load, α_b (normalized to the link capacity), and the mean burst duration of the background sources, B_b.

We applied the subset indicator strategy presented in Section 3, and used the total number of cells in all the queues as the basis for our primary subset indicator function. Thus,

$$T(V_k) = 1 + \sum_{i=1}^{M} X_k^i,$$

where X_k^i represents the length of the ith queue at time t_k. It can easily be seen that the above $T(V_k)$ becomes a weaker estimate of the potential delay of the tagged cells as M increases. This is due to the temporal shift in the contribution of each queue length to the total delay of any tagged cell. Therefore, we expect a decrease in the efficiency of DPR simulation using the above subset indicator strategy as M increases. This expectation is indeed supported by the numerical results, as presented below.

Figure 8 displays the D-PMF for $M = 1, 2, 5$ and 10, for the traffic configuration $\alpha_f = 0.1$, $B_f = 10$, $N = 7$, $\alpha_b = 0.4$, $B_b = 10$, and for buffer sizes $K = 500$ cell slots. After obtaining a quasi-balanced $\boldsymbol{\mu}$ vector for each of the four cases, 100 independent replications were executed, each with approximately 10^6 simulation slots for the $M = 1, 2$

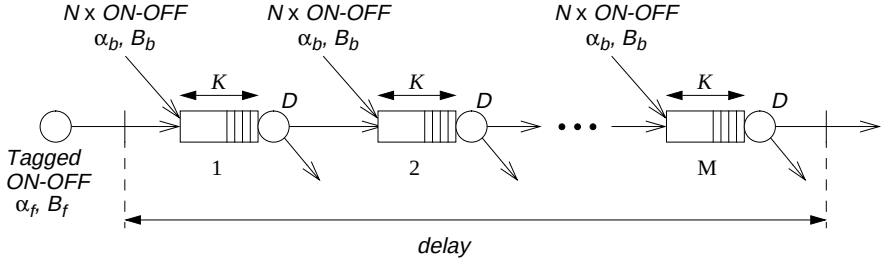

Figure 7: End-to-end delay measurement of a bursty connection over tandem ATM queues with bursty background traffic.

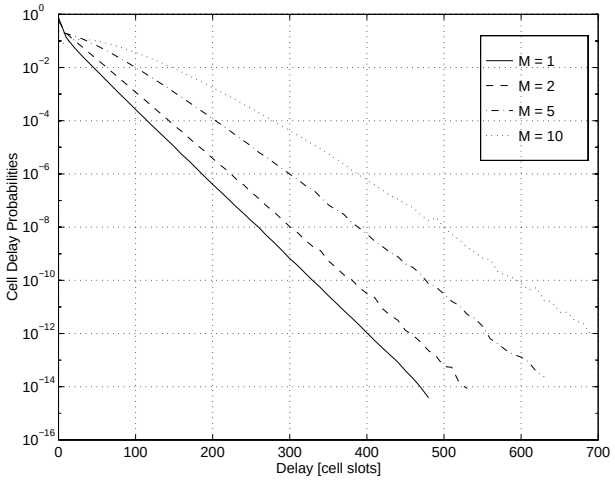

Figure 8: Cell Delay Probabilities for the tandem system for $M = 1, 2, 5$ and 10, estimated by DPR simulation.

cases, and with approximately 10^7 slots for the $M = 5$ and 10 cases. The required simulation times varied from 5 minutes (for the $M = 1$ case) to 8.5 hours (for the $M = 10$ case).

To compare the efficiency of DPR for the four cases, we focused on the delay probability estimate that was nearest to the 10^{-12} value, and used its estimator variance to estimate the simulation speed-up for each of the four cases. For values of $M = 1$, 2, 5, and 10, the respective speed-up factors obtained are 7.5×10^5, 1.1×10^5, 1.2×10^4, and 3.2×10^3. It can be concluded that the speed-up factor decays severely as M increases. This example shows that a more refined subset indicator strategy would have to be developed to keep DPR efficient if simulating larger numbers of tandem queues.

5 CONCLUDING REMARKS

In this paper we proposed a splitting rule to estimate very low delay threshold probabilities through a network using DPR-based splitting simulation. The method presented produces the prerequisite conditions for the rare target events. In the case of rare delay events, this involves forcing the system into longer queue lengths. However, unlike cell or packet loss, for delay events the method keeps the artificially generated sub-trajectories "alive" long enough for the cells with high potential to exit the system and thus generate the accountable target delay events.

We showed the efficiency of the technique by using it to measure delay probabilities in three different examples: a simple ATM multiplexer, a queueing system with multiple traffic classes, and a tandem queuing system with tagged and background traffic. For the first two examples, the factor of speed-up over straight-forward simulation was experimentally determined to be nearly inversely proportional to the probability estimate. In the third example, the improvement factor was very large but decreased as the number of tandem queues increased.

ACKNOWLEDGEMENTS

This work was supported in part by the Center for Advanced Computing and Communication (CACC), Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, North Carolina, U.S.A. Part of this work was carried out while Zsolt Haraszti was on a leave of absence to CACC, supported by Ericsson Telecom AB, Stockholm, Sweden.

References

Akyamac, A. A., Haraszti, Z., and Townsend, J. K. 1999. Efficient Rare Event Simulation Using DPR for Multidimensional Parameter Spaces. In *Proc. IEE International Teletraffic Congress, ITC16 (Edinburgh)*. Elsevier, Amsterdam, 767–776.

Bayes, B. A. 1970. Statistical Techniques for Simulation Models. *The Australian Computer Journal 2*, 4 (November), 180–184.

Glasserman, P., Heidelberger, P., Shahabuddin, P., and Zajic, T. 1996b. Multilevel Splitting for Estimating Rare Event Probabilities. Tech. Rep. RC 20478, IBM Research Division, T. J. Watson Research Center. June. To appear in *Operations Research* in 1999.

Glasserman, P., Heidelberger, P., Shahabuddin, P., and Zajic, T. 1996a. Splitting for Rare Event Simulation: Analysis of Simple Cases. In *1996 Winter Simulation Conference (San Diago)*. IEEE Press, Piscataway, NJ, 302–308.

Glasserman, P., Heidelberger, P., Shahabuddin, P., and Zajic, T. 1997. A Look at Multilevel Splitting. In *Monte Carlo and Quasi-Monte Carlo Methods 1996*, H. Niederreiter, P. Hellekalek, G. Larcher, and P. Zinterhof, Eds. Springer-Verlag, New York, 98–108.

Görg, C. and Fuß, O. 1999. Simulating Rare Event Details of ATM Delay Time Distributions with RESTART/LRE. In *Proc. IEE International Teletraffic Congress, ITC16 (Edinburgh)*. Elsevier, Amsterdam, 777–786.

Görg, C. and Schreiber, F. 1996. The RESTART/LRE Method for Rare Event Simulation. In *Proc. 1996 Winter Simulation Conference, WSC'96 (Coronado, California, USA)*. IEEE Press, Piscataway, NJ, 390–397.

Hammersley, J. M. and Handscomb, D. C. 1964. *Monte Carlo Methods*. Methuen and Co., Ltd., London.

Haraszti, Z. and Townsend, J. K. 1998. The Theory of Direct Probability Redistribution and its Application to Rare Event Simulation. In *Proc. IEEE Int. Conf. Commun., ICC '98 (Atlanta)*. IEEE Press, Piscataway, NJ, 1443–1450.

Haraszti, Z., Townsend, J. K., and Freebersyser, J. A. 1998. Efficient Simulation of TCP/IP for Mobile Wireless Communications Using Importance Sampling. In *Proc. IEEE MILCOM '98 (Boston)*. IEEE Press, Piscataway, NJ, 867–871.

Heidelberger, P. 1995. Fast Simulation of Rare Events in Queueing and Reliability Models. *ACM Trans. on Modeling and Comp. Sim. 5*, 1 (January), 43–85.

Hunter, J. J. 1983. *Mathematical Techniques of Applied Probability*. Vol. 2, Discrete Time Models: Techniques and Applications. Academic Press.

Kahn, H. and Harris, T. E. 1951. Estimation of Particle Transmission by Random Sampling. *National Bureau of Standards Applied Mathematics Series 12*, 27–30.

Katavenis, M., Sidiropoulos, S., and Courcoubetis, C. 1991. Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip. *IEEE J. Select. Areas Commun. 9*, 8 (Oct.), 1265–1279.

Villén-Altamirano, J. 1998. RESTART Method for the Case Where Rare Events Can Occur in Retrials From Any Threshold. *Int. J. Electron. Commun. (AEÜ) 52*, 3 (November), 183–189.

Villén-Altamirano, M., Martínez-Marrón, A., Gamo, J., and Fernández-Cuesta, F. 1994. Enhancement of the Accelerated Simulation Method RESTART by Considering Multiple Thresholds. In *Proc. 14th Int. Teletraffic Congress, ITC 14 (France)*. Vol. 1a. North-Holland, Amsterdam, 797–810.

Villén-Altamirano, M. and Villén-Altamirano, J. 1991. RESTART: A Method for Accelerating Rare Event Simulations. In *Proc. 13th Int. Teletraffic Congress, ITC 13 (Queueing, Performance and Control in ATM)*, P. C. Cohen J.W., Ed. North-Holland, Copenhagen, Denmark, 71–76.

AUTHOR BIOGRAPHIES

ZSOLT HARASZTI received the M.S. degree in Electrical Engineering from the Technical University of Budapest, Hungary, in 1993. He joined Ericsson Telecom AB in 1995 as a research fellow of the Traffic Analysis and Simulation Laboratory. In 1997 he was on a leave of absence to the Center for Advanced Computing and Communication (CACC), North Carolina State University, Raleigh, North Carolina, U.S.A. Since 1997 he works for Ericsson's Switching Laboratory (Switchlab) in Ericsson Radio Systems AB, Stockholm, Sweden. His main research interest is in communications system modeling, simulation and network traffic management. His Ph.D. research focus is on efficient simulation of rare events in packet switched networks. He is a member of the IEEE.

J. KEITH TOWNSEND received the B.S. degree in Electrical Engineering from the University of Missouri, Rolla, in 1981, and the M.S. and Ph.D. Degrees in Electrical Engineering in 1984 and 1988 respectively from the University of Kansas.

Before graduate study he was an Engineer in the Avionics Design Group at Bell Helicopter Textron, Ft. Worth, TX. While at the University of Kansas, his research activities included digital image processing, communications system modeling and analysis, and digital signal processing. He joined the faculty of the Electrical and Computer Engineering Department at North Carolina State University in July 1988, where he is now an Associate Professor. His current research interests include wireless communication systems, and techniques for simulation of rare events in communication links and networks.

Dr. Townsend is a member of Sigma Xi, Eta Kappa Nu, Tau Beta Pi, Phi Kappa Phi, and the IEEE. He has been a Guest Editor for the *IEEE Journal on Selected Areas in Communications*, and is currently an Editor for the *IEEE Transactions on Communications*.