The Impact of Cell Dropping Policies in ATM Nodes

Zhen Liu
INRIA, Centre Sophia Antipolis
2004 Route des Lucioles
B.P. 93, 06902 Sophia-Antipolis
FRANCE

Rhonda Righter*
Department of Operations and Management Information Systems
Santa Clara University, Santa Clara, CA 95053
USA

October 1996, revised June 1998
Submitted to Operations Research

Abstract

We consider policies for deciding which cells will be lost or dropped when losses occur at a finite buffer ATM node. The performance criteria of interest are the delay of transmitted (non-lost) cells, the jitter (or variability in the delay of transmitted cells), and the burstiness of lost cells. We analyze the performance tradeoffs for various cell dropping policies. We show the usual the "rear dropping" in which cells that arrive to a full buffer are lost stochastically maximizes delay, while "front dropping," in which cells at the front of the buffer are lost, stochastically minimizes delay. On the other hand, rear dropping stochastically minimizes the jitter. We also propose policies that have both stochastically smaller delay and less lost cell burstiness in a stochastic majorization sense than the rear dropping policy.

JITTER, ATM NETWORKS, STOCHASTIC MAJORIZATION, STOCHASTIC ORDERINGS, FINITE BUFFER QUEUES

*This work was done while Rhonda Righter was visiting INRIA.

We consider a single node of an ATM (Asynchronous Transfer Mode) network, in which messages are sent in packets or cells of a constant size. We study the delay of transmitted (unlost) cells, the jitter, or variability, in the delay, and the burstiness of lost cells. All of these issues are important with voice traffic for example. Voice traffic cannot tolerate long delays or much jitter. Also, voice messages can tolerate some losses, but will be more intelligible if the losses do not occur in groups of consecutive cells.

On the other hand, there are cases when it may be desirable to have losses occur in bursts. For example, when cells of data are grouped into larger packets, under some protocols if one cell in the packet is lost the entire packet must be retransmitted. Several cell discard algorithms have been proposed in the literature, see e.g. Kawahara et al. (1996).

Less burstiness in cell losses may again be preferred if there are some redundant cells in the packet. Indeed, in this case one may often be able to reconstruct lost cells and thus reduce packet losses. The reader is referred to Cidon et al. (1993) and references therein for details on how redundancy is used for error recovery.

Since cells have a fixed size, and the transmission rate is constant, we can assume service times are deterministic. We thus consider a discrete time *G/D/C(t)/b* queue, where the time slot equals the service time, which we take equal to 1 without loss of generality, and where b is the buffer size including the cell in service. The number of arrivals in each time slot is an arbitrary stochastic process that is independent of the identities of the cells in the buffer, though it may depend on the queue length. We assume that the service discipline is FCFS (first-come-first-served). The number of available servers at any time t, $C(t)$, may be a random variable representing for example the bandwidth of an ATM node allocated to a particular connection at time t. The process $C(t)$ may depend on the number of cells in the buffer (queue length), but is otherwise independent of the identities (indices) of the cells in the buffer. We assume that $C(t) \geq 1$ with probability 1 whenever the queue length is non-zero.

It is generally assumed in *G/G/s/b* loss systems that arrivals that find the buffer full are lost. Such a policy is called "rear dropping." An alternative policy is "front dropping", where cells at the front of the buffer (the ones that arrived earliest) are lost and the cells remaining in the buffer move up to the front, making room for new arrivals at the rear.

Some results exist in the literature for the single-server case (the *G/D/1/b* queue). It is easy to show, see Yin and Hluchyj (1993), that all dropping policies yield identical queue length distributions, provided dropping occurs only when there is buffer overflow, and that front dropping results in smaller delays for transmitted cells than rear dropping (or any other dropping, or "push-out," policy). Yin and Hluchyj (1993) also provide the delay distributions for both front and rear dropping under FCFS assuming a talk spurt model for arrivals. Schulzrinne (1993) shows that the same cells (as identified by their order of generation) will be dropped for all work conserving service policies and rear dropping, and that the distribution of loss runs under FCFS is the same for both rear and front dropping. Clare and Rubin (1986) give the delay distributions of transmitted cells for FCFS with front dropping and LCFS (last-come-first-served) with front dropping when the distribution of arrivals in each time slot is independent and identically distributed (i.i.d.). They show that rear dropping maximizes the mean waiting time for transmitted cells under any work conserving service discipline, and that LCFS with front dropping minimizes the mean waiting time over all service disciplines and dropping policies. However, FCFS has the desirable properties that waiting times are strictly bounded and the order of transmitted cells is preserved.

Takine et al. [8] analyze the distributions of queue lengths and loss periods, and the probability of consecutive losses, for a system with correlated arrivals, FCFS service, and rear dropping. See the references in that paper for other analyses of queueing models for ATM nodes with fixed dropping policies.

Lakshman, Neidhardt, and Ott (1996) show that for networks using TCP, the Internet transfer protocol, a front dropping policy results in better performance and allows the use of smaller buffers than rear dropping. During congestion episodes when buffers are full, front dropping causes the destination to "see" missing cells approximately one buffer drain time earlier than under rear dropping. The sources correspondingly receive earlier duplicate acknowledgments and can adjust accordingly.

In this paper we consider the $G/D/C(t)/b$ queueing model under the above mentioned assumptions. Such a model can be used for the analysis of either a single dedicated ATM node or a particular source with its own input buffer in an ATM node with multiple sources sharing the channels. We first consider the standard front and rear dropping policies. For both policies, if more than one cell is lost in a time slot, a "burst" of consecutive cells will be lost. We show that the loss burstiness is the same for both front and rear dropping, assuming cells are only lost when the buffer is full. We give a bound on the difference in mean delay for front and rear dropping and show that rear dropping minimizes jitter (defined later) over all dropping policies.

Since in many applications we want to minimize loss burstiness, we propose new policies, that we call *splitting* policies, and show that they have less bursty losses in a majorization sense than front or rear dropping. The delays of these policies are between that of front dropping and rear dropping. We provide specific bounds on their mean delay relative to the front and rear dropping policies. The effect of such policies on the loss burstiness and jitter are evaluated using simulation and compared to a random splitting policy.

The paper is organized as follows. We define our model and assumptions in section 1. We also present some basic properties of majorization and stochastic orderings. In section 2 we compare the two simplest dropping policies: front dropping and rear dropping. We consider different performance measures such as delay, jitter, and loss burstiness. In section 3 we propose two new classes of policies, parametrizable by a threshold, which we refer to as split and batch policies. The front and rear dropping policies turn out to be special cases of these new policies. We present properties of split and batch policies with respect to the performance measures of interest, and show that the split dropping policy has less loss burstiness than front or rear dropping. We then give the policy that maximizes loss burstiness in section 4. In section 5 we present simulation results in order to illustrate the effect of the threshold on these policies. We also compare our polices with a random-dropping policy. The comparison has been performed for two (somewhat extreme) input traffic patterns: Poisson arrivals and a process with long-range dependence. Finally, in section 6 we present our conclusions.

1 Preliminaries

1.1 Basic Model Assumptions and Definitions

We use smaller, increasing, etc. in the nonstrict sense.

We will consider a discrete time G/D/C(t)/b queue, where the time slot equals the service time, which we take equal to 1. There is a finite-capacity buffer of size b, including the cell in service. The number of servers $C(t)$ available at time t is a random variable. The service process $C(t)$ can depend on the number of cells in the buffer (queue length), but is otherwise independent of the identities (indices) of the cells in the buffer. We assume that $C(t) \geq 1$ with probability 1 whenever the queue length is non-zero. The number of arrivals in each time slot is an arbitrary stochastic process that is independent of the identities of the cells in the buffer, though it may depend on the queue length. The arrival process is arbitrary and can depend on the queue length, but not on the identities of cells in the queue. Moreover, the arrival process and the service process $C(t)$ can be mutually dependent.

We consider dropping policies in which cells are dropped only when the buffer has overflowed, and we never drop more cells than we have to. We also assume the service discipline is FCFS, and that cells are always ordered by their arrival times in the buffer. Then since we only drop cells when the buffer overflows, and since all cells have a service time equal to 1, the queue size, the number of transmissions $(0, 1, \ldots, c(t))$, and the number of losses in any time slot is the same under any dropping policy for any fixed arrival process and service process. Therefore, dropping policies affect only the identities of lost cells, not the times at which they are lost, and similarly, they affect only the identities of transmitted cells, and not the transmission times.

In our study, we shall assume that the processes for arrivals and number of available servers are fixed arbitrarily. We assume there will be a total of $N < \infty$ arrivals. Let L and T be the total number of losses and transmitted cells respectively in the N arrivals. Thus, $T + L = N$. Cells are indexed according to their order of arrival, and we assume them to be ordered within batches of arrivals (arrivals that occur during the same time slot). Suppose there is a "virtual" buffer of infinite size that includes the "real" buffer of size b, such that at the end of a time slot all arrivals during that time slot are placed in the buffer in order after the cells that are already present. If there are at most b cells in the virtual buffer, no cells are lost. If there are more than b cells, say $b + a$, where $a > 0$, then a cells must be dropped.

We assume that the arrival times of all cells in the buffer are known, and we permit policies that use this information. However, as we will see, optimal policies will depend only on the order of arrivals, through the buffer positions of the current cells.

Let a_i be the arrival time of cell i. If cell i is transmitted, we define its completion time, c_i, as the time at which it has finished its transmission at the ATM node, and we define its delay d_i as $c_i - a_i$. If cell i is lost, we let l_i be the time at which it is lost, and we define its delay as $l_i - a_i$. For any policy π let T^π and L^π be the set of transmitted and lost cells respectively and let $\bar{d}_T^\pi$, $\bar{d}_L^\pi$, and $\bar{d}^\pi$ be the average delay for transmitted, lost, and all cells respectively. Let D_T^π, D_L^π, and D^π be the corresponding total delays.

1.2 Stochastic Ordering and Majorization

We now give some brief definitions and useful facts relating to stochastic orderings and majorization. For further details and proofs, see Marshall and Olkin (1979).

Recall that for two random vectors $\mathbf{X}$ and $\mathbf{Y}$, $\mathbf{X}$ is stochastically larger than $\mathbf{Y}$, $\mathbf{X} \geq_{st} \mathbf{Y}$, if and only if $E(g(\mathbf{X})) \geq E(g(\mathbf{Y}))$ for all component-wise increasing functions $g(\mathbf{t})$, provided the expectations exist. Also, $\mathbf{X} \geq_{st} \mathbf{Y}$ if and only if it is possible to construct two coupled random vectors, that is, two random vectors on the same probability space, $\widehat{\mathbf{X}}$ and $\widehat{\mathbf{Y}}$, such that $\widehat{\mathbf{X}} =_{st} \mathbf{X}$, $\widehat{\mathbf{Y}} =_{st} \mathbf{Y}$, and $\widehat{\mathbf{X}} \geq \widehat{\mathbf{Y}}$ with probability 1.

For two m dimensional real vectors $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{x}$ majorizes $\mathbf{y}$, $\mathbf{x} \succ \mathbf{y}$, if $\sum_{i=1}^k x_{[i]} \geq \sum_{i=1}^k y_{[i]}$, for $k = 1, \ldots, m-1$ and $\sum_{i=1}^m x_{[i]} = \sum_{i=1}^m y_{[i]}$, where $x_{[i]}$ is the i^{th} largest component of $\mathbf{x}$. Intuitively, $\mathbf{y}$ is better balanced than $\mathbf{x}$. A function f is said to be Schur-convex if $f(\mathbf{x}) \geq f(\mathbf{y})$ whenever $\mathbf{x} \succ \mathbf{y}$. For two random vectors $\mathbf{X}$ and $\mathbf{Y}$, $\mathbf{X}$ is larger than $\mathbf{Y}$ in the Schur-convex sense, $\mathbf{X} \geq_{scx} \mathbf{Y}$, if and only if $E(g(\mathbf{X})) \geq E(g(\mathbf{Y}))$ for all Schur-convex functions $g(\mathbf{t})$. Also, $\mathbf{X} \geq_{scx} \mathbf{Y}$ if and only if it is possible to construct two coupled random vectors, that is, two random vectors on the same probability space, $\widehat{\mathbf{X}}$ and $\widehat{\mathbf{Y}}$, such that $\widehat{\mathbf{X}} =_{st} \mathbf{X}$, $\widehat{\mathbf{Y}} =_{st} \mathbf{Y}$, and $\widehat{\mathbf{X}} \succ \widehat{\mathbf{Y}}$ with probability 1.

2 Front and Rear Dropping

Suppose in a particular time slot a cells must be dropped. Under rear dropping (denoted by R) the cells in virtual buffer positions $b+1, \ldots, b+a$ are lost. Under front dropping (denoted by F) the cells in buffer positions $1, \ldots, a$ are lost and the remaining cells are moved up to positions $1, \ldots, b$, into the real buffer. After the dropping policy is implemented the cell in position 1 is then served during the next time slot, and the process is repeated at the end of the next time slot with the arrivals that occur during that slot.

2.1 Delays for Front and Rear Dropping

Let $i^F(k)$ ($i^R(k)$) be the index of the k^{th} cell to be lost under front (rear) dropping. We have the following lemma that says that the indices of lost cells under front and rear dropping are just shifted versions of each other.

Lemma 2.1 *For fixed arrival and service processes,*

$$i^R(k) = i^F(k) + b.$$

Proof. Consider any time slot in which there are losses, say a of them. Note that under FCFS with front dropping, the cells in the virtual buffer always have consecutive indices, and under both front and rear dropping any cells that are in the virtual buffer but not in the real buffer are labeled consecutively and are the most recent arrivals. Therefore, let l be such that the indices of the cells in positions $b+1, b+2, \ldots, b+a$ are $l+1, l+2, \ldots, l+a$ under both policies. Then cells $l+1, \ldots, l+a$ are lost under rear dropping, and cells $l+1-b, \ldots, l+a-b$ are lost under front dropping since under front dropping the cells in the virtual buffer are consecutive. $\square$

Using lemma 2.1 we can bound the difference in the average delay for transmitted cells under front and rear dropping. That is,

$$\bar{d}_T^\pi = \frac{1}{T} D_T^\pi = \frac{1}{T} \sum_{i \in T^\pi} (c_i - a_i), \qquad \bar{d}_L^\pi = \frac{1}{L} D_L^\pi = \frac{1}{L} \sum_{i \in L^\pi} (l_i - a_i),$$

and

$$\bar{d}^\pi = D^\pi / N = (D_T^\pi + D_L^\pi)/N = \frac{1}{N} (\sum_{i \in T^\pi} c_i + \sum_{i \in L^\pi} l_i - \sum_{i=1}^{N} a_i),$$

where the variables are as defined in section 1.1.

Corollary 2.2

$$0 \leq \bar{d}_T^R - \bar{d}_T^F \leq (b-1)\frac{L}{N}.$$

Proof. Note that, as we observed earlier, since we assume arrivals occur at the same times under both front and rear dropping, completions and losses also occur at the same time under the two policies, i.e.,

$$\sum_{i \in T^F} c_i = \sum_{i \in T^R} c_i, \qquad \sum_{i \in L^F} l_i = \sum_{i \in L^R} l_i.$$

Therefore,

$$D^F = D^R, \qquad D_T^R - D_T^F = D_L^F - D_L^R = \sum_{i \in L^R} a_i - \sum_{i \in L^F} a_i.$$

By lemma 2.1, if the k^{th} cell to be lost under front dropping is cell i, then the k^{th} cell to be lost under rear dropping is cell $i+b$, so

$$\sum_{i \in L^R} a_i = \sum_{i \in L^F} a_{i+b}.$$

Because of the constant service time and FCFS discipline, if a cell is served, it will be served within b time units of its arrival under both front and rear dropping. Thus, under front dropping, if cell i is lost it must be the case that cell $i+b$ arrived before cell i could be served, i.e., $a_{i+b} \leq a_i + b - 1$. Also, by definition, $a_i \leq a_{i+b}$. Therefore

$$\sum_{i \in L^F} a_i \leq \sum_{i \in L^F} a_{i+b} \leq \sum_{i \in L^F} (a_i + b - 1) = \sum_{i \in L^F} a_i + L(b-1),$$

and the result follows. □

In fact, the lower bound can be strengthened to $\mathbf{d}_T^R \geq_{st} \mathbf{d}_T^F$, where $\mathbf{d}_T^\pi$ is the vector of delays of transmitted cells under policy π, $\pi = F, R$. From lemma 2.1 it is easy to show that $j^R(k) \leq j^F(k)$, where $j^\pi(k)$ is the k^{th} cell to be transmitted under policy π. That is, the k^{th} cell to be transmitted under R has an earlier arrival time than the k^{th} cell to be transmitted under F, but the departure time is the same under both policies. The same ideas can be used to show that for any dropping policy π that only drops cells when the buffer is full, $i^R(k) \geq i^\pi(k) \geq i^F(k)$, so $j^R(k) \leq j^\pi(k) \leq j^F(k)$, and therefore $\mathbf{d}_T^R \geq \mathbf{d}_T^\pi \geq \mathbf{d}_T^F$. Indeed, this is partially a corollary of lemmas 3.1 and 3.2 that appear later. Summarizing,

Corollary 2.3 *For any arrival and service processes and any policy π, with probability 1,*

$$\mathbf{d}_T^F \leq \mathbf{d}_T^\pi \leq \mathbf{d}_T^R.$$

Therefore, $\mathbf{d}_T^F \leq_{st} \mathbf{d}_T^\pi \leq_{st} \mathbf{d}_T^R$.

2.2 Jitter for Front and Rear Dropping

Now let us consider the jitter, or variability of the delay. We will show that rear dropping minimizes the jitter. Let $s(\pi)$ and $t(\pi)$ be the first and last transmitted cells respectively under π for the given arrival and service processes. Let $\widehat{T}^\pi = T^\pi - \{s(\pi)\}$. Also, if cell i is transmitted under π, we define $e(i)$ to be the index of the last cell before cell i to be transmitted under π (the dependency on π is suppressed). We assume the system is initially empty.

The jitter under an arbitrary policy π is defined to be the sum of absolute differences in consecutive delays, i.e.,

$$J^\pi = \sum_{i \in \widehat{T}^\pi} |(c_i - a_i) - (c_{e(i)} - a_{e(i)})|.$$

Note that although $s(\pi)$ may depend on π, $a_{s(\pi)}$ and $c_{s(\pi)}$ do not. This is because of the assumption that $C(t) \geq 1$ whenever there are cells in the buffer so that at least one cell in the first batch is transmitted. That is, $a_{s(\pi)}$ is the time of the first batch of arrivals and $c_{s(\pi)} = a_{s(\pi)} + 1$. Similarly, $a_{t(\pi)}$ and $c_{t(\pi)}$ do not depend on π although $t(\pi)$ may. These arguments are in fact valid for the first and last cells transmitted within each busy period. Indeed, the busy periods do not depend on the dropping policy π. Note that the jitter of π for the whole set of transmitted cells is the sum of jitters within each of the busy periods, plus the delay differences between the first transmitted cell in each busy period and the last transmitted cell in the previous busy period. Since these last delay differences are independent of π, when we compare the jitters resulting from different dropping policies, it suffices to consider a single busy period.

Hence in the remaining part of this subsection we *redefine* N and L to be the number of arrivals and losses during an arbitrary busy period for arbitrary arrival and service processes, and we index the cells starting from the first cell to arrive in the busy period, i.e., $i = 1, \ldots, N$.

Since there is at least one departure in every time slot during a busy period (recall that $C(t) \geq 1$), we have $c_i - c_{e(i)} \leq 1$, for $i \in T^\pi$. We assume, without loss of generality, that servers are numbered $1, 2, \ldots$ and that server 1 is always available during the busy period, and that when there are multiple servers, the cell served by server 1 has the smallest index. Let $\widetilde{T}^\pi$ be the subset of $\widehat{T}^\pi$ containing the cells which are transmitted by server 1. Let $\widetilde{N} = |\widetilde{T}^\pi|$ be the number of cells in $\widetilde{T}^\pi$. Note that $\widetilde{N} + 1$ equals the length of the busy period and is independent of π.

Therefore, letting $I\{\bullet\}$ be the indicator function, the jitter can be rewritten as

$$J^\pi \;=\; \sum_{i \in \widehat{T}^\pi - \widetilde{T}^\pi} |a_i - a_{e(i)}| + \sum_{i \in \widetilde{T}^\pi} |a_i - a_{e(i)} - 1|$$

$$
\begin{aligned}
&= \sum_{i\in\widehat{T}^\pi-\widetilde{T}^\pi}(a_i-a_{e(i)})+\sum_{i\in\widetilde{T}^\pi}(a_i-a_{e(i)}-1)I\{a_i>a_{e(i)}\}+\sum_{i\in\widetilde{T}^\pi}I\{a_i=a_{e(i)}\}\\
&= \sum_{i\in\widehat{T}^\pi}(a_i-a_{e(i)})-\sum_{i\in\widetilde{T}^\pi}I\{a_i>a_{e(i)}\}+\sum_{i\in\widetilde{T}^\pi}I\{a_i=a_{e(i)}\}\\
&= a_{t(\pi)}-a_{s(\pi)}-(\widetilde{N}-\sum_{i\in\widetilde{T}^\pi}I\{a_i=a_{e(i)}\})+\sum_{i\in\widetilde{T}^\pi}I\{a_i=a_{e(i)}\}\\
&= -\widetilde{N}+a_{t(\pi)}-a_{s(\pi)}+2\sum_{i\in\widetilde{T}^\pi}I\{a_i=a_{e(i)}\}.
\end{aligned}
$$

Therefore, to minimize jitter, we must minimize $\sum_{i\in\widetilde{T}^\pi}I\{a_i=a_{e(i)}\}$, the sum of the number of time slots used for transmitting cells from the same batch minus the number of batches that are at least partially transmitted. Thus, we have the following dominance property for the policies that transmit at least one cell from each batch:

Theorem 2.4 *For a fixed arrival process and any policy π that drops at least one whole batch, there is a dropping policy $\hat{\pi}$ which transmitts at least one cell from each batch such that $J^{\hat{\pi}} \le J^\pi$ with probability 1.*

Proof. Suppose that π drops all the cells of a batch, say batch k. Let c be the cell that is transmitted by server 1 that has the least index greater than any of the indices of batch k. Under $\hat{\pi}$ let cell c be dropped and let server 1 transmit a cell, say cell j, from batch k instead. Let $\hat{\pi}$ otherwise agree with π. Then under $\hat{\pi}$, $a_j \neq a_{e(j)}$, so $\sum_{i\in\widetilde{T}^{\hat{\pi}}}I\{a_i=a_{e(i)}\} \le \sum_{i\in\widetilde{T}^\pi}I\{a_i=a_{e(i)}\}$ and $J^{\hat{\pi}} \le J^\pi$ with probability 1. □

If we confine ourselves to such dominant policies, the number of (at least partially) transmitted batches is simply the total number of batches that arrive. Thus, the jitter minimization is transformed into a combinatorial optimization problem, namely the minimization of the total number of time slots used for transmitting cells from the same batch minus the number of (at least partially) transmitted batches.

In the case there is only one server, $C(t)\equiv 1$, the total number of time slots used for transmitting cells from the same batch is equal to $N-L$, the number of transmitted cells, which is independent of the policy π. Hence, theorem 2.4 implies

Corollary 2.5 *For a fixed arrival process and $C(t)\equiv 1$, any dropping policy which transmitts at least one cell from the same batch minimizes the jitter with probability 1.*

Corollary 2.6 *For a fixed arrival process and $C(t)\equiv 1$, rear dropping minimizes the jitter with probability 1.*

Note that rear dropping will no longer minimize jitter when $C(t) \neq 1$. Suppose for example $C(t) \equiv 2$. Then if exactly two cells from the same batch were transmitted, $\sum_{i \in \widetilde{T}^\pi} I\{a_i = a_{e(i)}\}$ would not increase because only one of the two cells would be transmitted by server 1. Thus, with two servers, we would want to drop all but two cells from the same batch. The optimal policy for this case is no longer easily characterized. For example, if we could not drop all but two cells from the same batch, we would be indifferent to transmitting three and transmitting four cells from that batch.

2.3 Loss Burstiness for Front and Rear Dropping

We now mathematically characterize the burstiness of losses for a dropping policy and show that under this definition the loss burstiness for front and rear dropping is the same.

Let the loss burstiness of a dropping policy π be defined as a vector $LB^\pi = (LB_1, LB_2, \ldots LB_L)$ where LB_i is the number of consecutive losses in the i^{th} group of consecutive losses; LB_i is defined to be 0 if there are fewer than i groups of losses. More rigorously, we recursively define l_i and t_i, $i = 1, \ldots, L$, as follows, with $t_0 := 0$. Let l_i be the index of the first loss in the i^{th} group of consecutive losses, so $l_i = \min\{j : j \in L^\pi \text{ and } j > t_{i-1}\}$ if $\{j : j \in L^\pi \text{ and } j > t_{i-1}\}$ is non-empty; let $l_i = N + 1$ otherwise. Similarly, let t_i be the first cell to be transmitted after the i^{th} group of consecutive losses, so $t_i = \min\{j : j \in T^\pi \text{ and } j > l_i\}$ if $\{j : j \in T^\pi \text{ and } j > l_i\}$ is non-empty; let $t_i = N + 1$ otherwise. Then $LB_i = t_i - l_i$. We will refer to LB_i as the group size of the i^{th} group of losses. Note that if our objective is to minimize loss burstiness, as in voice applications, or when redundancy is encoded, the "ideal" loss burstiness is the vector with all components equal to 1, that is, where each loss is isolated. On the other hand, if we want to maximize loss burstiness, the "ideal" loss burstiness is the vector with one component equal to L and all other components equal to 0.

Now consider front and rear dropping for fixed arrival and service processes. From lemma 2.1 the group sizes for consecutive losses are the same for front and rear dropping since the indices of lost cells are just shifted. That is, we have the following corollary.

Corollary 2.7 *For fixed arrival and service processes, with probability 1,*

$$LB_i^F = LB_i^R = L_i.$$

where L_i is the number of losses the i^{th} time there are losses.

To see the last equality of the corollary, first recall that the number of transmissions, the number of losses, and the queue size is the same for all time slots for any dropping policy. If we have a set of losses in a particular time slot, under both front and rear dropping they will be consecutive losses. Also, because of our slotted system, at least one cell will be transmitted before the next set of losses occurs. Suppose in a particular time slot we have a set of losses, and let j $(j + b)$ be the index of the last lost cell in that set under policy F (R). Then under F cell $j + 1$ will be transmitted in the next time slot, and under R a cell will be transmitted in the next time slot so cell $j + b + 1$ will be able to enter the buffer, and will not be dropped. Thus, the set of lost cells in any time slot will be separated from the set of lost cells in any other time slot by at least one transmitted cell in both policies.

We next define the split dropping policy (S) that we prove to have less loss burstiness than front or rear dropping.

3 The Split Dropping Policy

As we saw in the previous section, a problem with both front and rear dropping is that every time there is a set of losses, they will be consecutive losses. Since in many applications we want to minimize loss burstiness, we propose the following split policy, with parameter θ, $\theta \in \{0, 1, \ldots, b\}$, which we call $S(\theta)$, or just S when we need not denote the dependency on θ. Suppose that in the current time slot we have losses, and that a cells must be dropped. Again we assume there is a "virtual" buffer that includes the "real buffer" such that all arrivals are placed in the buffer in order after the cells that are already present, so that in the current time slot there are cells in positions $1, \ldots, b + a$. If $a \leq \theta$ then we drop the first a cells in the buffer, as in front dropping, and move the remaining cells up into the "real" buffer. If $a > \theta$ then we drop the first θ cells in the buffer as well as the cells in positions $b + 1, b + 2, \ldots, b + a - \theta$, and then move the remaining cells into the real buffer. Thus the dropped cells of the current time slot are "split" or separated, though now, as we will see, consecutive losses can occur in more than one time slot. Note that $S(b) = F$ and $S(0) = R$.

It will be convenient to define another dropping policy, which we call the batch policy with parameter θ, $B(\theta)$. Under $B(\theta)$ if we have $b + a$ cells in our virtual buffer, we drop the cells in buffer positions $b - \theta + 1, \ldots b - \theta + a$. Thus $B(b) = F$ and $B(0) = R$.

Let $\Phi(\theta)$, $\theta \in \{0, 1, \ldots, b\}$, be the set of all policies defined as follows. Suppose that in a particular time slot a is the total number of cells that must be dropped (the total buffer overflow). Then a policy π is in $\Phi(\theta)$ if it drops $\min\{a, \theta\}$ cells from the real buffer, that is from among positions $1, \ldots, b$, and drops the remaining $\max\{a - \theta, 0\}$ from outside of the real buffer, that is, from among positions $b + 1, \ldots, b + a$. Thus, $S(\theta) \in \Phi(\theta)$ and $B(\theta) \in \Phi(\theta)$. We permit policies in $\Phi(\theta)$ that vary over time and depend on the state.

3.1 Delay for the Split Dropping Policy

We first show the following lemmas regarding the delays of the proposed policies.

Lemma 3.1 *For all $\theta \in \{0, 1, \ldots, b\}$, for all $\pi \in \Phi(\theta)$,*

$$\mathbf{d}_T^{S(\theta)} \leq \mathbf{d}_T^{\pi}$$

with probability 1, assuming $S(\theta)$ and π start in the same state and have the same arrival and service processes.

Proof. Let π be an arbitrary dropping policy in $\Phi(\theta)$, and let $p^{\pi}(i)$ ($p^S(i)$), $i = 1, \ldots, n$ be the index of the cell in position i of the (real) buffer at the beginning of a time slot (just before a transmission and after implementing the loss policy), where $0 \leq n \leq b$ is the number of cells in the buffer (the queue length). As we noted before, the queue lengths, departure times, and loss times will be the same under any policy with the same arrival process, so n does not depend on the policy. We will show by induction on the time slots that for each slot,

$$p^{\pi}(i) \leq p^S(i)$$

for $i = 1, \ldots, n$. The lemma will follow from this, since the cell that is transmitted in a time slot is in the first position, i.e., cell $p(1)$. A smaller index for a transmitted cell means an earlier arrival time, which means for the same transmission time the delay is greater.

Note that because cells always enter the buffer in order and leave in order (because of the FCFS service discipline, we have for $i = 1, \ldots, n-1$,

$$p^{\pi}(i) \leq p^{\pi}(i+1), \qquad p^S(i) \leq p^S(i+1).$$

For the first time slot $p^{\pi}(i) \leq p^S(i)$ because both policies start in the same state. Let us suppose $p^{\pi}(i) \leq p^S(i)$ at the beginning of time slot t and consider time slot $t+1$. The cell in position 1 (if there is one) will have been transmitted and there will be a', say, new arrivals. If $a' + n - 1 \leq b$ then all the new arrivals will enter the buffer in order, and the inequality will be preserved. Suppose $a := a' + n - 1 - b > 0$. Then the new arrivals enter the virtual buffer in order and $p^{\pi}(i) \leq p^S(i)$ still holds for the vector of cells occupying positions in the virtual buffer, i.e., when n, $p^{\pi}(i)$, and $p^S(i)$, are redefined to be the number of cells in the virtual buffer and the index of the cell in position i in the virtual buffer at the beginning of time slot n, *before* implementation of the dropping policy. Let $r^{\pi} = (r_1^{\pi}, \ldots r_b^{\pi})$ ($r^S = (r_1^S, \ldots r_b^S)$) be the positions in the virtual buffer of the cells that remain after the dropping policy for time slot $t+1$ is implemented, but before the remaining cells are moved up into the virtual buffer. Then, if $a \geq \theta$, $r^S = (\theta+1, \theta+2, \ldots, b-1, b, b+a-\theta+1, \ldots, b+a)$, and if $a < \theta$, $r^S = (a+1, a+2, \ldots, b+a)$. Thus, in either case $r_i^\pi \leq r_i^S$, $i = 1, \ldots, b$, for all $\pi \in \Phi(\theta)$. Therefore, $p^\pi(r_i^\pi) \leq p^\pi(r_i^S) \leq p^S(r_i^S)$, so $p^\pi(i) \leq p^S(i)$ for time slot $t+1$ *after* implementing the dropping policy, and we are done. □

The same argument gives us the following.

Lemma 3.2 *For any fixed arrival and service processes, for all* $\theta \in \{0, 1, \ldots, b-1\}$,

$$\mathbf{d}_T^{S(\theta+1)} \leq \mathbf{d}_T^{S(\theta)}, \qquad \mathbf{d}_T^{B(\theta+1)} \leq \mathbf{d}_T^{B(\theta)}.$$

The following lemma corresponds to lemma 2.1. There is no corresponding shifted relationship between lost cells under policies S(θ) and F.

Lemma 3.3 *For all* $0 \leq \theta \leq b$,

$$i^{B(\theta)}(k) = i^F(k) + b - \theta.$$

Proof. The argument is similar to that of lemma 2.1. Consider any time slot in which there are losses, say a of them. As we noted before, under FCFS with front dropping, the cells in the virtual buffer always have consecutive indices. Also, under $B(\theta)$, the cells in positions $b-\theta+1, \ldots b+a$ must all have indices after the index of the cell dropped the last time any cells were dropped, and they therefore must have consecutive indices. In other words, under $B(\theta)$ we can think of the virtual buffer as being split into two buffers, one with buffer positions $1, \ldots, b-\theta$ from which cells are never dropped, and the other with buffer positions $b-\theta+1, \ldots, b+a$ in which the policy is front dropping and which feeds into the first sub-buffer. Cells in the second sub-buffer always have consecutive indices. Therefore, let l be such that the indices of the cells in positions $b-\theta+1, b-\theta+2, \ldots, b-\theta+a$ are $l+1, l+2, \ldots, l+a$ under both policies. Then cells $l+1, \ldots, l+a$ are lost under $B(\theta)$ and cells $l+1-(b-\theta), \ldots, l+a-(b-\theta)$ are lost under F. □

We also have the following corollary, which is analogous to corollary 2.2. The first inequality follows from lemmas 3.1 and 3.2 and the fact that $F = S(b)$, the second follows from lemma 3.1, and the third follows from lemma 3.3 in the same way as corollary 2.2 follows from lemma 2.1.

Corollary 3.4

$$0 \leq \bar{d}_T^{S(\theta)} - \bar{d}_T^F \leq \bar{d}_T^{B(\theta)} - \bar{d}_T^F \leq (b-\theta-1)\frac{L}{N}.$$

3.2 Loss Burstiness for the Split Dropping Policy

Now let us consider the loss burstiness of our proposed policies. From lemma 3.3 we have the following generalization corollary 2.7.

Corollary 3.5 *For any fixed arrival and service processes, for all θ,*

$$LB^{B(\theta)} = LB^F = LB^R.$$

We first recall a basic majorization lemma for integer vectors. Let $\mathbf{a}$ and $\mathbf{b}$ be L dimensional integer vectors (that is, all their components are integers). The transformation from $\mathbf{b}$ to $\mathbf{b}'$ is called a transfer if, for $b_i > b_j$, $b'_i = b_i - 1$, $b'_j = b_j + 1$, and $b'_k = b_k$, $k \neq i, j$. We have the following lemma which is due to Muirhead (see Marshall and Olkin, 1979, Lemma 5.D.1).

Lemma 3.6 *If $\mathbf{a}$ and $\mathbf{b}$ are integer vectors and $\mathbf{a} \prec \mathbf{b}$, then $\mathbf{a}$ can be derived from $\mathbf{b}$ by successive applications of a finite number of transfers.*

Our main result for this section is the following, which, along with the corollary above, shows that our split dropping policy is better than front or rear dropping in terms of minimizing loss burstiness in the sense that the losses are grouped into more groups of smaller size under the split policy.

Theorem 3.7 *For any fixed arrival and service processes, for all θ,*

$$LB^{S(\theta)} \prec LB^{B(\theta)} = LB^F = LB^R.$$

Therefore, for stochastic arrival and service processes, for all θ,

$$LB^{S(\theta)} \leq_{scx} LB^{B(\theta)} =_{st} LB^F =_{st} LB^R.$$

Proof. Let us fix θ, $1 \leq \theta \leq b-1$, and let $B = B(\theta)$ and $S = S(\theta)$. First consider LB^B. As we noted in section 2.2, for front and rear dropping, and hence for B by corollary 3.5, the i^{th} group size of losses is just L_i, the number of losses the i^{th} time there are losses. Suppose losses occur M times during the first N arrivals. Then, $LB^B = (L_1, \ldots, L_M, 0, 0, \ldots 0)$, where the number of 0's is $L - M$. Note that a lower bound, in the sense of majorization, for LB for any dropping policy is $(1, 1, \ldots, 1)$, where there are L 1's. Let x_i (y_i) be the number of cells that are dropped from within the real buffer (from outside of the real buffer) the i^{th} time there are losses under both policies B and S. Note that for all $i = 1, \ldots, M$, $L_i = x_i + y_i$, $0 \leq x_i \leq \theta$, and if $y_i > 0$ then

$x_i = \theta$. Therefore, since the order in which components appear in a vector does not matter for majorization comparisons, we can write $LB^B = (x_{[1]} + y_{[1]}, \ldots, x_{[M]} + y_{[M]}, 0, 0, \ldots 0)$, where $x_{[i]}$ (resp. $y_{[i]}$) is the i^{th} largest component in the vector $\mathbf{x} = (x_1, \ldots, x_M)$ (resp. $\mathbf{y} = (y_1, \ldots, y_M)$).

Let X_i^π (Y_i^π) be the sets of indices of the cells dropped from within the real buffer (from outside of the real buffer) the i^{th} time there are losses, $\pi = S, B$, $i = 1, \ldots,, M$. (Y_i^π may be empty.) Then Y_i^π (if it is nonempty) contains consecutive indices under both policies, and under B, X_i^B contains consecutive indices, and Y_i^B (if it is nonempty) will be consecutive with X_i^B. We will also speak generically of "X groups" and "Y groups." We say that two groups "group together" if the union of the two groups is a set of consecutive indices. So, for example under B, X_i^B groups together with Y_i^B (if it is nonempty) for each i.

Now consider the split policy, S. It is possible under S that X_i^S does not contain consecutive indices. When this happens, there is a group Y_j^S such that X_i^S and Y_j^S are grouped together in the sense that their union is a set of consecutive indices. As a specific example, consider a system with one server, $b = 4$, $\theta = 2$, cells 1,2,$\ldots$,7 arrive in the first time slot, cells 8, 9, and 10 arrive in the second time slot, and there are no further arrivals. Then under policy S, cells 1, 2, and 5 will be dropped ($X_1^S = \{1, 2\}$, $Y_1^S = \{5\}$) and cell 3 will be transmitted in the first time slot, cells 4 and 6 will be dropped ($X_2^S = \{4, 6\}$, $Y_2^S = \emptyset$) and cell 7 will be transmitted in the second time slot. Cells 8, 9, and 10 will be transmitted in the following time slots. Hence, cells 4, 5, and 6 form a group of consecutive losses, and $LB^S = (2, 3, 0, 0, 0)$. (Under policy B, cells 3, 4, 5, 7, and 8 will be dropped, and $LB^B = (3, 2, 0, 0, 0)$ which equals LB^S in the majorization sense.)

It is also possible under S that for some $i < j$, X_j^S is consecutive with Y_i^S, that is, the largest index in Y_i^S equals the smallest index in X_j^S minus 1. For example, suppose there is one server, $b = 2$, $\theta = 1$, cells 1,2,3, and 4 arrive in the first time slot, cells 5 and 6 arrive in the second time slot, and there are no further arrivals. Then under policy S, cells 1 and 3 will be dropped ($X_1^S = \{1\}$, $Y_1^S = \{3\}$) and cell 2 will be transmitted in the first time slot, cell 4 will be dropped ($X_2^S = \{4\}$, $Y_2^S = \emptyset$) and cell 5 will be transmitted in the second time slot, and cell 6 will be transmitted in the third time slot. Hence, cells 3 and 4 form a group of consecutive losses, and $LB^S = (1, 2, 0)$. (Under policy B, cells 2, 3, and 5 will be dropped, and $LB^B = (2, 1, 0)$ which equals LB^S in the majorization sense.)

Let us consider the possible ways under policy S that the X_i^S's and Y_j^S's can group together. Let $\hat{x}_i$ and $\hat{y}_j$ ($\bar{x}_i$ and $\bar{y}_j$) be the largest (smallest) index in X_i^S and Y_j^S, for non-empty Y_j^S's, respectively. We have the following properties.

P1 *For each $i = 1, \ldots, M-1$ there exists t such that $\hat{x}_i < t < \bar{x}_{i+1}$ and such that cell t is transmitted.* This is because a cell with an index larger than $\hat{x}_i$ be at the head of the queue after the cells in X_i^S and Y_i^S are dropped, and will be transmitted during the next time slot, before any cells in X_{i+1}^S are dropped. A consequence is that for $i \neq j$, X_i^S and X_j^S cannot group together, and, for $i \leq j$, X_i^S and Y_j^S cannot group together.

P2 $\bar{y}_j > \hat{y}_i + \theta + 1$ *for all $i < j$ such that Y_i^S and Y_j^S are nonempty.* This is because cells $\hat{y}_i + 1, \ldots, \hat{y}_i + \theta + 1$ will enter the real buffer after the cells in X_i^S (whose cardinality is θ as Y_i^S is nonempty) and Y_i^S are dropped. Note that there is at least one cell departure at each time slot. A consequence is that for $i \neq j$, Y_i^S and Y_j^S cannot group together.

Our properties imply that groups of consecutive losses under S will consist of either individual X or Y groups or groups of the form $Y_i^S \cup X_j^S$ with $i < j$. Indeed, due to P1, there can never be groups of the form $X_i^S \cup X_j^S$ for $i \neq j$, or $X_i^S \cup Y_j^S$ with $i < j$. Also, in view of P2, no two Y groups can group together, nor can they be bridged by an X group (whose cardinality is bounded above by θ), i.e. there can be no groups of consecutive losses of the form $Y_i^S \cup X_j^S \cup Y_k^S$ with $i \neq k$. Thus, the only possible merged groups have the form $Y_i^S \cup X_j^S$ with $i < j$.

Let $g \leq M$ be the number of groups of the form $X_i^S \cup Y_j^S \cup X_i^S$ under S, let $j_1, \ldots j_g$ and $i_1, \ldots, i_g$ be the indices of the corresponding Y_j^S's and X_i^S's respectively, and let $k_1, \ldots, k_{M-g}$ and $l_1, \ldots, l_{M-g}$ be the indices of the Y_j^S's and X_i^S's respectively that are unmatched (or empty) under S. Then $LB^S = (x_{i_1} + y_{j_1}, \ldots, x_{i_g} + y_{j_g}, x_{k_1}, \ldots, x_{k_{M-g}}, y_{l_1}, \ldots, y_{l_{M-g}}, 0, 0, \ldots 0)$. Let $LB' = (x_{i_1} + y_{j_1}, \ldots, x_{i_g} + y_{j_g}, x_{k_1} + y_{l_1}, \ldots, x_{k_{M-g}} + y_{l_{M-g}}, 0, 0, \ldots, 0)$, where the number of 0's in the vectors is such that they each have L components. Then, from our transfer lemma, lemma 3.6, $LB^S \prec LB'$. Moreover, using the rearrangement inequalities (Propositions 6.A.1 and 6.A.2, p.140) of Marshal and Olkin (1979), we obtain $LB' \prec LB^B = (x_{[1]} + y_{[1]}, \ldots, x_{[M]} + y_{[M]}, 0, 0, \ldots 0)$. Therefore, $LB^S \prec LB^B$. $\square$

Let us now consider the distribution of the size of groups of losses under our policies. Let P_k^π be the proportion of nonzero loss groups that have size at least k under policy π for the first N arrivals. We have the following.

Corollary 3.8 *For a fixed arrival process, for all θ, and for $k \geq \theta$,*

$$P_k^{S(\theta)} \leq P_k^{B(\theta)}.$$

That is, $S(\theta)$ is less likely to have large groups of consecutive losses than $B(\theta)$.

Proof. Fix θ and let S, B, M, and L_i be as defined in the previous proof. We have that

$$P_k^B = \frac{n_k^B}{M},$$

where

$$n_k^B = \sum_{i=1}^{M} I\{L_i \geq k\}$$

is the number of groups with size at least k under policy B, and I is the indicator function. Also,

$$P_k^S = \frac{n_k^S}{g'},$$

where n_k^S is the number of groups with size at least k under policy S, and where g' is the number of groups under S. From the previous proof it is easy to see that $g' \geq M$ and that for $k \geq \theta$, $n_k^S \leq n_k^B$ so that $P_k^S \leq P_k^B$. $\square$

4 Maximizing Loss Burstiness

Now suppose we want to maximize loss burstiness, that is, we want large groups of cells with consecutive indices to be dropped. This might be the case when we are transmitting large packets of consecutive cells, where if one cell of the packet is lost, the entire packet is corrupted and must eventually be retransmitted. Whenever the buffer overflows, if there are no cells in the buffer that are consecutive to cells that have been dropped earlier, then we should drop from the rear, otherwise we should either drop from the rear or drop those cells that are consecutive to earlier dropped cells. More rigorously, we have the following.

Partial Characterization of the Optimal Policy

1. Set $j = -1$ at time 0. (As the algorithm progresses j will be the cell dropped the last time there was a buffer overflow.)

2. When there is an overflow, of size a say, with $k > j$ being the index of the last cell to arrive,

   a. if cell $j + 1$ is in the buffer then

      either drop cells $j + 1, \ldots, j + a$ and reset j to $j + a$,
      or drop cells $k - a + 1, \ldots, k$ and reset j to k;

   b. else, drop cells $k - a + 1, \ldots, k$ and reset j to k.

Theorem 4.1 *The policy that maximizes the loss burstiness in the Schur-convex sense has the properties characterized above.*

Proof. It is easy to see that for any policy that does not drop the last cells (rear dropping) when there are no cells in the buffer that are consecutive with earlier dropped cells, a policy that does drop the last cells will have larger loss burstiness in the Schur-convex sense. This is because there is a greater chance that these cells could be consecutive with later dropped cells. Thus, rear dropping is optimal the first time there is a buffer overflow. Suppose that a policy agrees with the characterization above for the first m buffer overflows, and consider the $m+1$'st buffer overflow, with overflow size a. If there are no cells in the buffer consecutive with earlier cells at this time, then again rear dropping will dominate any other policy.

Therefore, suppose that at the $m+1$'st buffer overflow time there are cells in the buffer that are consecutive with earlier dropped cells, i.e., there is a cell $j+1$ in the buffer such that cells $j, j-1, \ldots, j-\hat{a}$ have been dropped for some $\hat{a}$. Then again it is not hard to see that any policy that does not drop either the last cells in the buffer or cells $j+1, \ldots, j+a$ will be dominated by a policy that does do one of these two things. $\square$

5 Comparison of Dropping Policies by Simulation

We now investigate the effects of the threshold on different dropping policies. We further compare the policies $S(\theta)$ and $B(\theta)$ with a random dropping policy $U(\theta)$. The policy $U(\theta)$ is similar to $S(\theta)$ and $B(\theta)$ except that the $\min(\theta, a)$ cells to be dropped from the real buffer are chosen randomly according to a uniform distribution. If $a > \theta$ the cells in positions $b+1, b+2, \ldots, b+a-\theta$ in the virtual buffer are also dropped.

We used the simulator developed by N. Niclausse at INRIA to generate different curves of the average delay, the sample variance of the sizes of groups of consecutive losses, and the jitter. In our experiments, we have chosen the buffer size as 10. There is a single server: $C(t) \equiv 1$.

We consider two (extreme) input traffic processes. In the first one the number of arriving cells in each time slot has a Poisson distribution. The simulation results are reported in Figure 1 and 2 which are obtained with input rates $\rho = 0.95$ and $\rho = 3.0$, respectively.

The second input traffic process has long-range dependence. It is defined as the number of busy servers in the $M/G/\infty$ model of Cox (1984). The arrivals to the $M/G/\infty$ queue follow a Poisson process with parameter λ, and service times are i.i.d. with a Pareto distribution with parameter $\alpha > 1$. Let $\overline{F}(x)$ be the tail distribution of the service times, and σ be a strictly

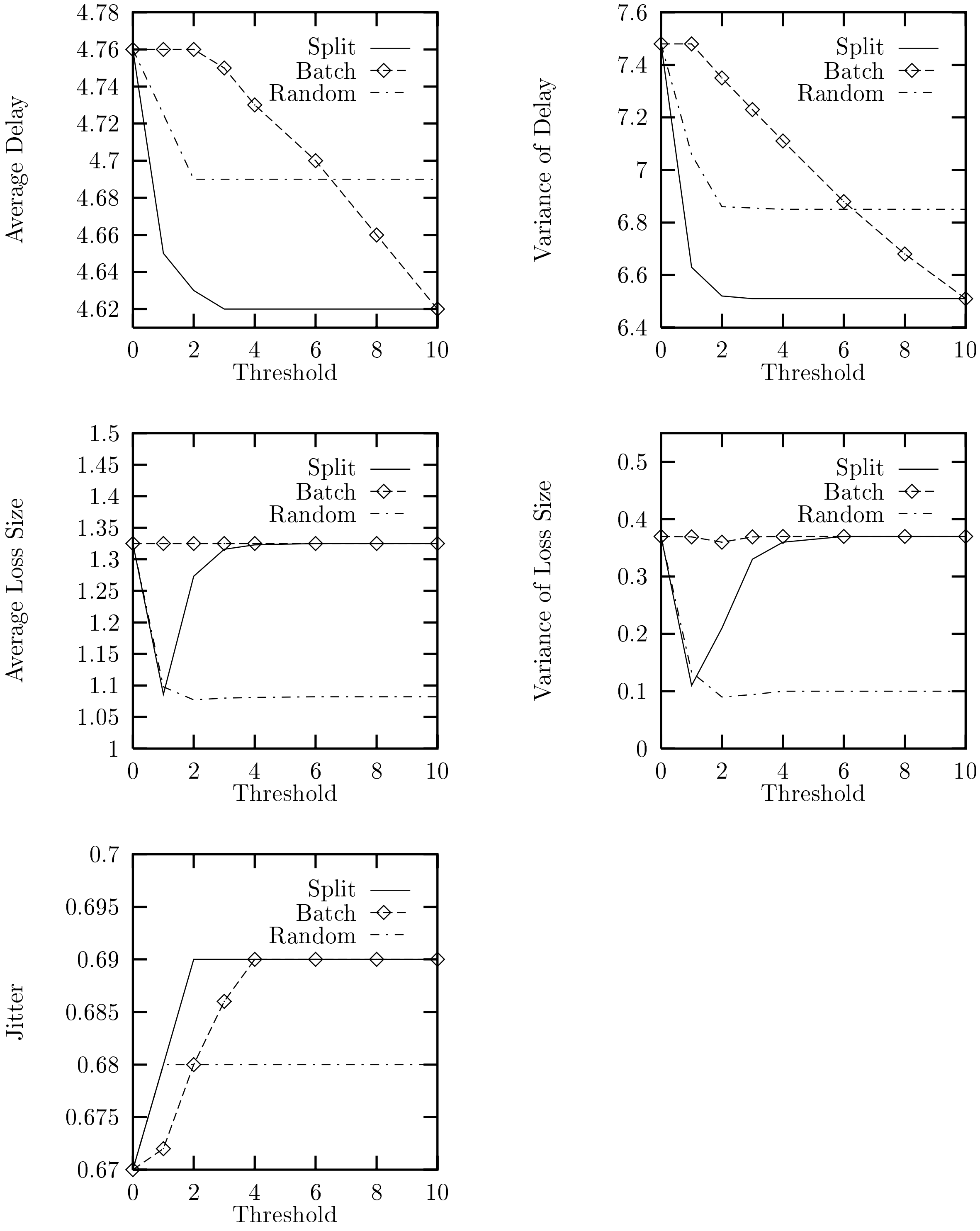

Figure 1: Performance Comparison with Poisson Arrivals, $\rho = 0.95$

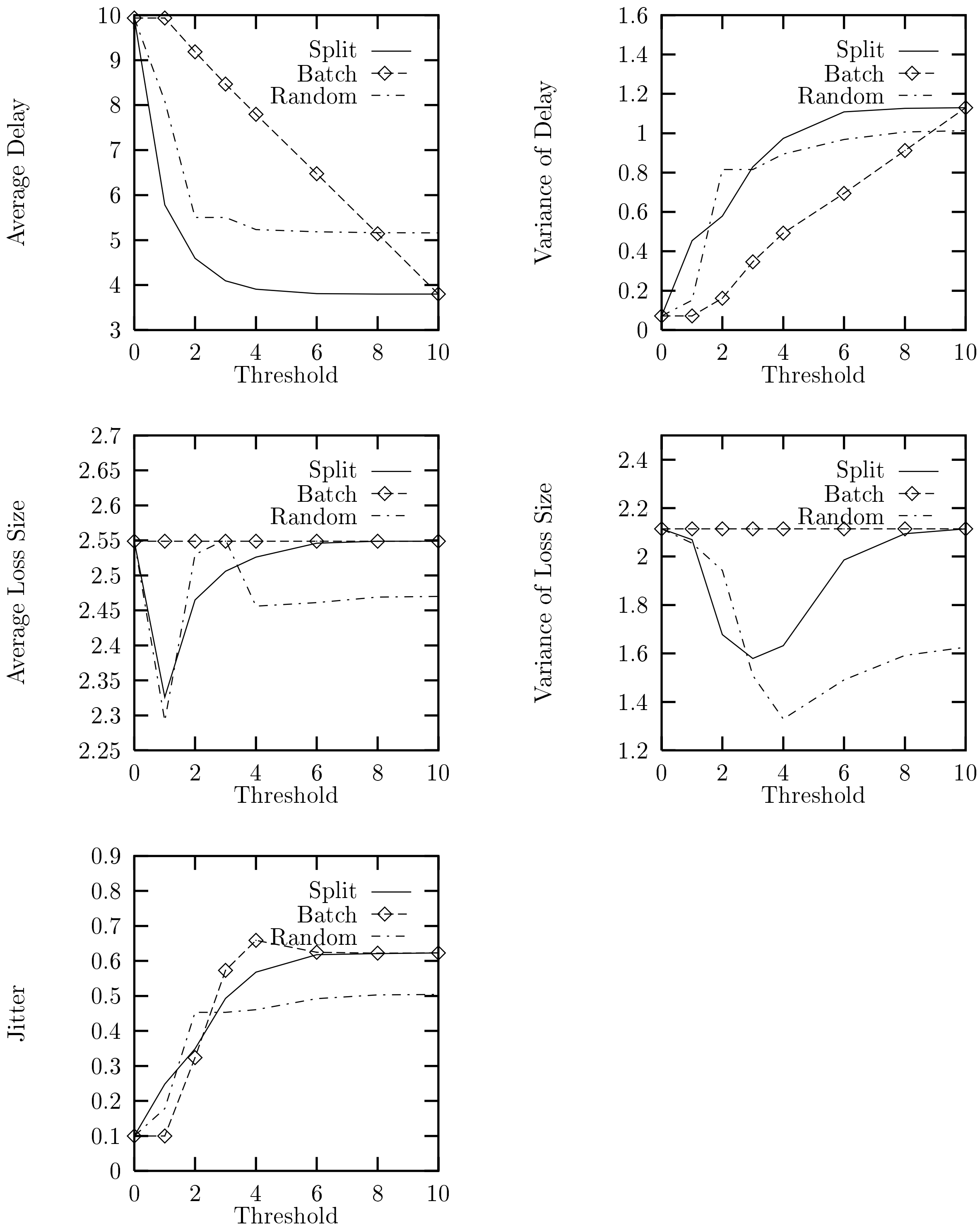

Figure 2: Performance Comparison with Poisson Arrivals, $\rho = 3.0$

positive real number, so

$$\overline{F}(x) = \begin{cases} 1, & x < \sigma \\ (\frac{x}{\sigma})^{-\alpha} & \end{cases}.$$

The expectation of the service times is $\sigma\alpha/(\alpha-1)$, and the average number of busy servers in the system is $\lambda\sigma\alpha/(\alpha-1)$. Assume that $1 < \alpha < 2$. It is known that the process of the number of busy servers in the M/G/∞ model has long-range dependence and is asymptotically self-similar with Hurst parameter $H = (3-\alpha)/2$. This process is used as our input traffic process.

More specifically, we first simulate the $M/G/\infty$ queueing system. We take the *number of busy servers* at integer time instants in this queueing system as the *number of cells* to arrive in each time slot in the discrete-time *G/D/C(t)/b* queueing system.

The simulation results are reported in Figure 3 (with $\lambda = 0.3125$, $\alpha = 1.2$, $\sigma = 0.5$ so that the input rate $\rho = 0.9375$) and Figure 4 (with $\lambda = 1.0$, $\alpha = 1.2$, $\sigma = 0.5$ so that the input rate $\rho = 3.0$). In both cases, the Hurst parameter is 0.9.

It is not surprising to see that the average delays under all these policies are decreasing in the threshold θ. This confirms the theoretical monotonicity properties established for split and batch policies (lemma 3.2). The delay under the batch policy decreases almost linearly in θ, whereas under the split and random policies the effect of the parameter θ on the delay vanishes quickly.

The experimental results illustrate that the variance of the delay is monotone in the threshold θ as well. However, it is decreasing when the input rate $\rho < 1$, and increasing when $\rho > 1$.

The loss burstiness, measured here by the sample average size of loss groups, is independent of θ and equals the average overflow size under the batch policy, as predicted by corollary 2.7 and theorem 3.7. However, under the other two policies, the effect of the parameter θ is quite important for small values of θ. Note that there is no monotonicity of the loss burstiness with respect to θ, at least not under the split policy. The variance of the loss sizes has similar behavior. Under the split policy, the loss burstiness is minimized with $\theta = 1$ in all our experiments.

The simulation results also indicate that the jitter increases in θ under all the three policies. This is consistent with the fact that when $\theta = 0$, all three policies reduce to rear dropping which was shown to minimize the jitter (corollary 2.6).

Observe that the behaviors of the split and batch policies are quite similar for the two input traffic processes. The delay and loss burstiness (resp. jitter) are smaller (resp. larger) under the split policy than under the batch policy. The random policy, however, behaves differently in comparison with the other two policies with respect to loss burstiness and jitter. For the Poisson arrival process with input rate $\rho < 1$, the random policy has the smallest loss burstiness and

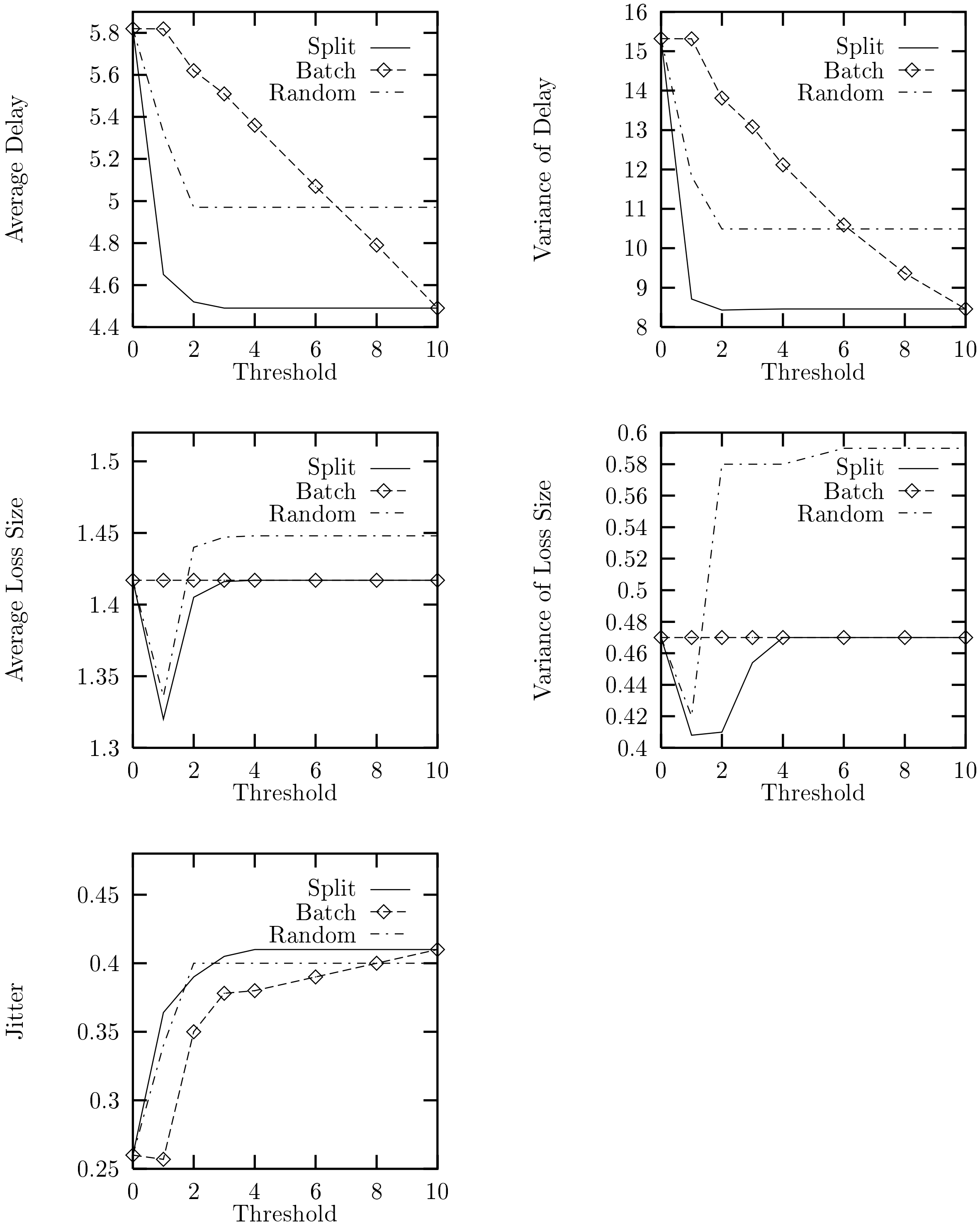

Figure 3: Performance Comparison with Cox Arrivals, $\rho = 0.9375$

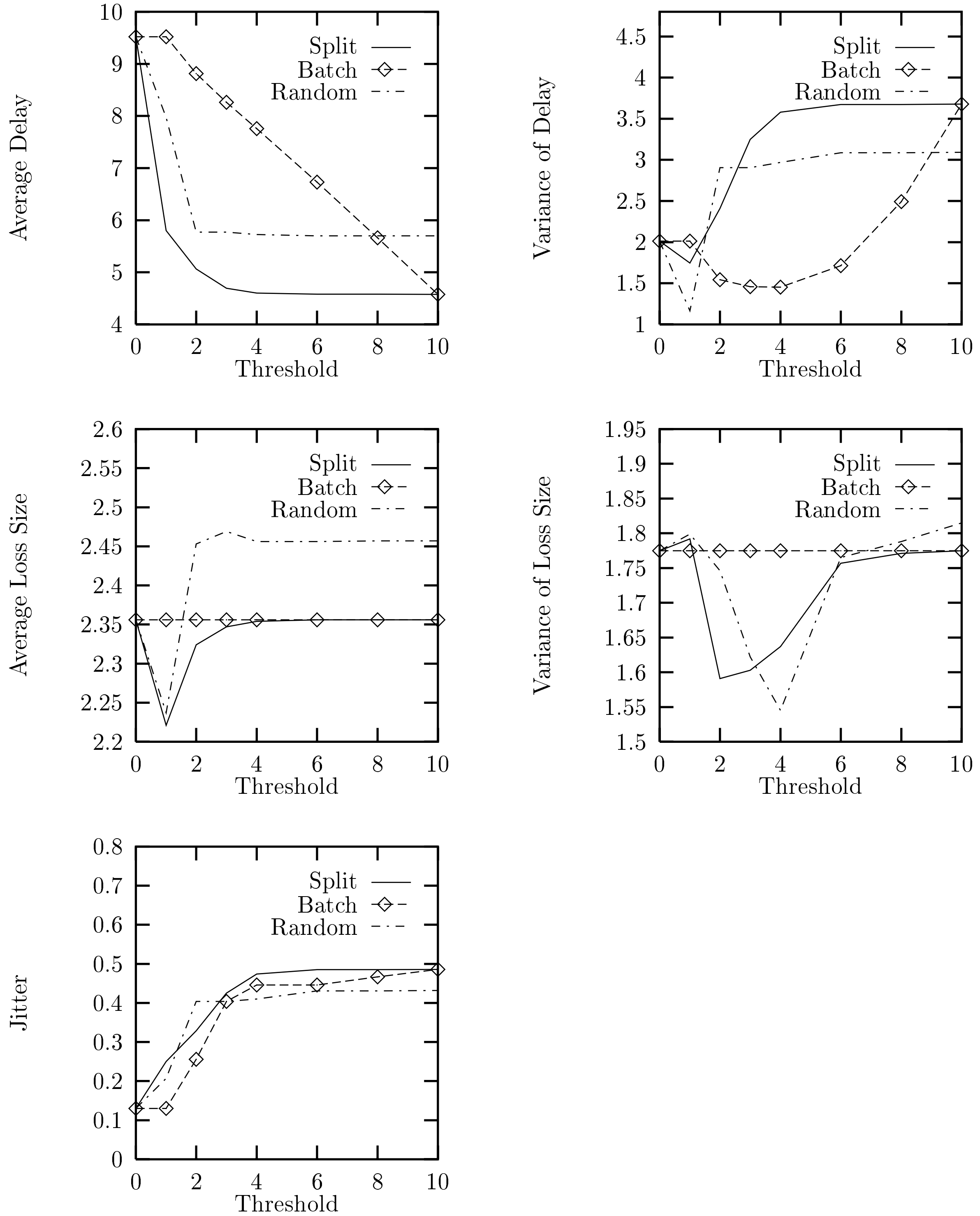

Figure 4: Performance Comparison with Cox Arrivals, $\rho = 3.0$

jitter, whereas for the other cases (Poisson arrival process with $\rho > 1$ or long-range dependent arrival processes), it has the largest loss burstiness, and generally larger jitter.

Finally, we present some performance comparisons between Cox arrivals and Poisson arrivals, see Figures 5 and 6. We illustrate the results for the split policy alone. The other policies have similar comparison results. The curves in Figure 5 for the Cox arrivals are the same as those in Figure 3. In Figure 5, the Poisson arrivals have rate 0.9375, the same as that of the Cox arrivals. The curves in Figure 6 are extracted from Figures 2 and 4.

One observes that for the same traffic load, Cox arrivals exhibit larger mean delay. This is consistent with known results in the queueing literature.

For the comparison of loss burstiness, there does not seem to be simple arguments to explain the fact that Cox arrivals exhibit larger loss burstiness when $\rho < 1$, and smaller loss burstiness when $\rho > 1$.

The jitter is smaller under Cox arrivals than under Poisson arrivals for both cases $\rho < 1$ and $\rho > 1$. It seems to be difficult to theoretically justify such a result. Intuitively, as the burstiness of the arrivals, and so of losses, is larger under Cox process, the total number of time slots used for transmitting cells from the same batch is smaller as well. Thus, the jitter becomes smaller, in accordance with the remark preceding theorem 2.4.

6 Conclusions

In this paper we have analyzed the impact of different cell dropping policies on the performance measures like cell delay, jitter, and loss burstiness. We have considered a discrete-time queueing system with a single finite-capacity queue and a time-varying number of deterministic servers. We have proposed split dropping and batch dropping policies, which are parametrizable policies and have as special cases the usual rear and front dropping policies when the parameter (threshold) is set to 0 and the buffer size, respectively. We have shown that the delay is stochastically decreasing in the threshold, and that front (rear) dropping minimizes (maximizes) the delay. With respect to the loss burstiness, we have shown that these last two policies are identical. We have also shown that split dropping yields smaller loss burstiness and delay than the rear dropping. For the minimization of jitter, however, we have proved that rear dropping is optimal.

In addition to these theoretical results, we have also used simulation to evaluate the impact of the parameter of the dropping policies on the performance measures. We have compared the above mentioned policies with a uniformly distributed random-dropping policy, assuming two (somewhat extreme) input traffic patterns, Poisson arrivals and a process with long-range-dependence. The behavior of the split and batch policies are quite similar for the two input traffic processes. The delay and loss burstiness (resp. jitter) are smaller (resp. larger) under

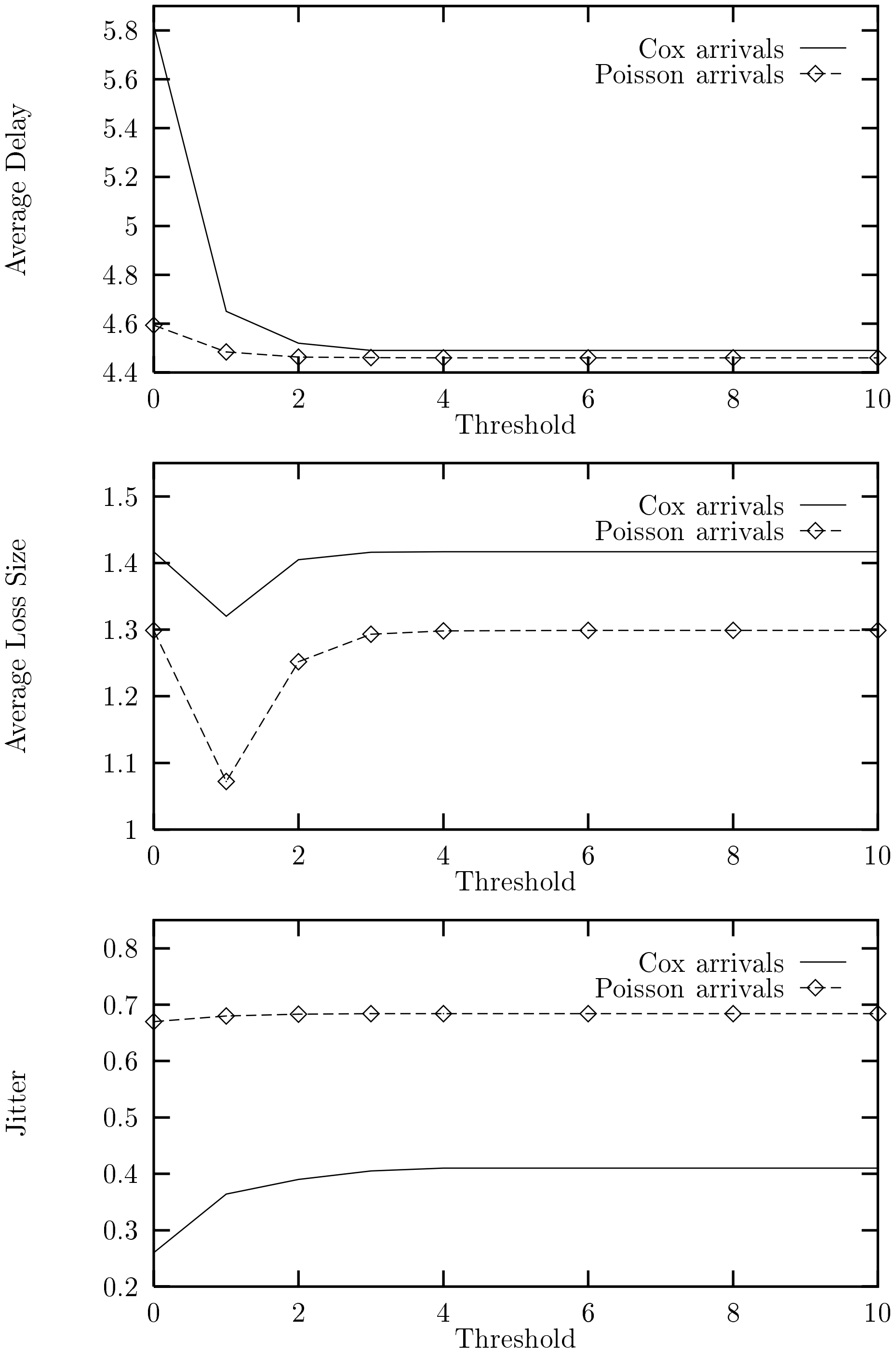

Figure 5: Performance of the Split Policy with Cox Arrivals and Poisson Arrivals, $\rho = 0.9375$

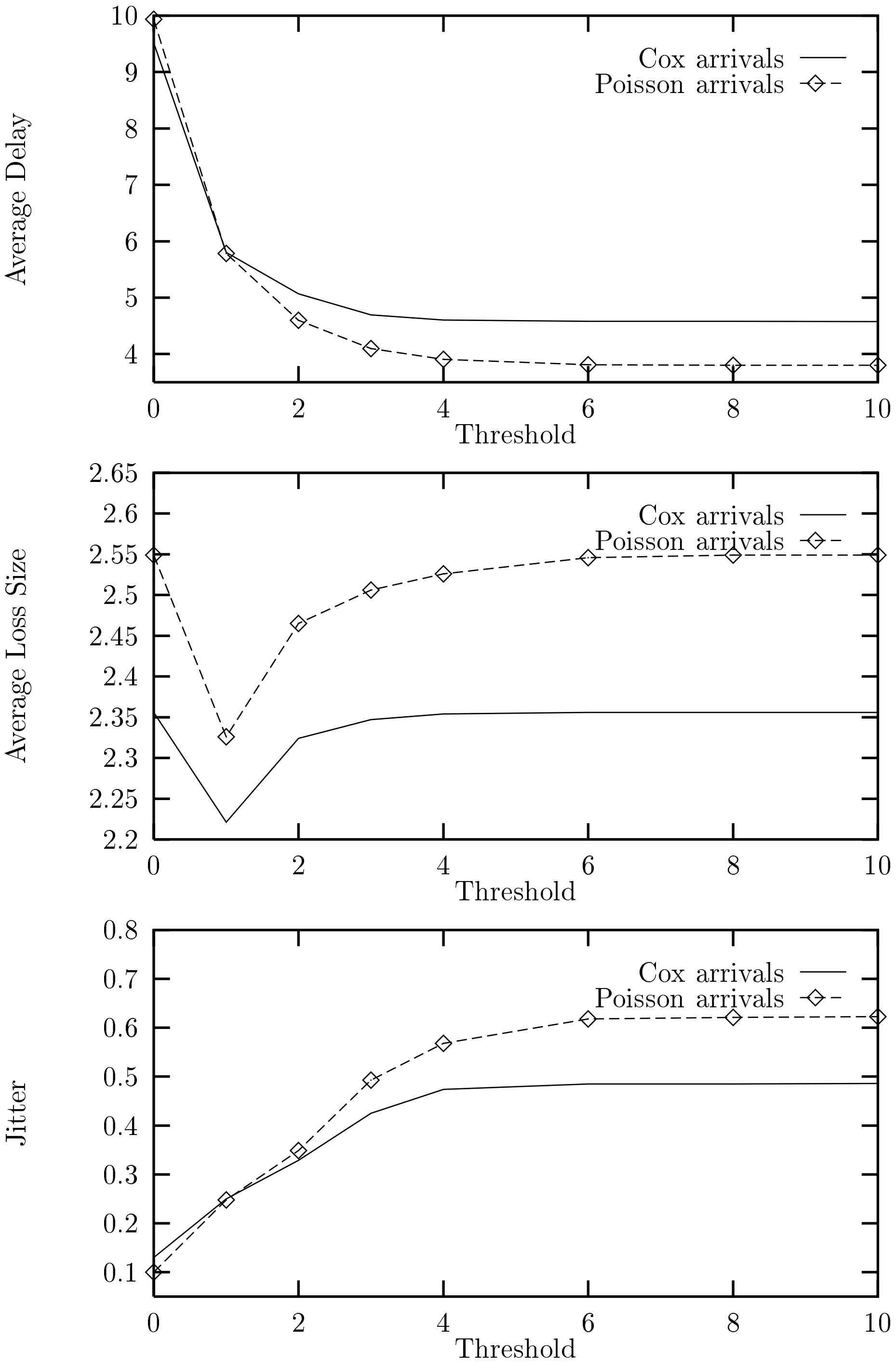

Figure 6: Performance of the Split Policy with Cox Arrivals and Poisson Arrivals, $\rho = 3.0$

the split policy than under the batch policy. The loss burstiness and jitter of the random policy relative to split and batch policies depends on the input traffic process.

We conjecture that the jitter is an increasing function of θ under all the three policies. This conjecture is supported by the simulation results, but we were unable to prove it.

Acknowledgments: We thank Nicolas Niclausse of INRIA for developing the simulator that generated the results presented in Section 5. This simulator uses the program developed by Jorn Migge of INRIA for the simulation of the M/G/∞ queueing system. We also thank the referees for helpful comments and pointing out an error in our original version of theorem 2.4.

References

[1] Cidon, I., A. Khamsy and M. Sidi (1993). Analysis of packet loss processes in high-speed networks. *IEEE Trans. on Information Theory*, 39: 98-108.

[2] Clare, L. P., and I. Rubin. (1986). Preemptive buffering disciplines for time-critical sensor communications *IEEE Int'l. Conf. on Comm.* 904-909.

[3] Cox, D. R. (1984). Long-range dependence: a review. *An Appraisal*, H. A. David and H. T. David (Eds.), The Iowa State University Press, Ames IA, pp. 55-74.

[4] Kawahara, K., K. Kitajima, T. Takine, Y. Oie (1996). Performance evaluation of selective cell discard schemes in ATM networks. *Proc. INFOCOM'96*, San Francisco, USA, 1054-1061.

[5] Lakshman, T. V., A. Neidhardt, T. J. Ott. (1996). The drop from front strategy in TCP and in TCP over ATM. *IEEE INFOCOM Proc.* 1242-1250.

[6] Marshall, A. W. and I. Olkin (1979). *Inequalities: Theory of Majorization and Its Applications*, Academic Press, NY.

[7] Schulzrinne, H. G. (1993). Reducing and characterizing packet loss for high-speed computer networks with real-time services. Ph.D. Dissertation (Chapter 5); EECS, U. Mass.

[8] Takine, T., T. Suda, and T. Hasegawa (1995). Cell loss and output process analyses of a finite-buffer discrete-time ATM queueing system with correlated arrivals. *IEEE Trans. on Comm.* 43: 1022-1037.

[9] Yin, N., and M. G. Hluchyj. (1993). Implication of dropping packets from the front of a queue *IEEE Trans. Comm.* 41: 846-851.