Title
Exact emulation of a priority queue with a switch and delay lines

Journal
Queueing Systems, 53(3)

ISSN
0257-0130

Authors
Sarwate, A D
Anantharam, V

Publication Date
2006-07-01

Peer reviewed

Exact emulation of a priority queue with a switch and delay lines

A. D. Sarwate (asarwate@eecs.berkeley.edu)* and
V. Anantharam (ananth@eecs.berkeley.edu)†
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley

Abstract. All-optical packet switched networking is hampered by the problem of realizing viable queues for optical packets. Packets can be buffered in delay lines, but delay lines do not functionally emulate queues from an input-output point of view. In this paper we consider the problem of *exact* emulation of a priority queue of size K using a switching system comprised of a switch of size $(M+1)\times(M+1)$, which has one distinguished input for external arrivals, one distinguished output for external departures, and fixed-length delay lines of lengths $L_1, L_2, \ldots, L_M$ connecting the other inputs and outputs in pairs. We measure the complexity of such an emulation by $M+1$. We prove that $M \geq \lceil \log(K-1) \rceil$ and present a construction which works with $M = O(\sqrt{K})$; further, in our construction $\sum_{m=1}^{M} L_m = K + O(\sqrt{K})$. We also sketch an idea for an all-optical packet switched communication network architecture based on *approximate* emulation of priority queues of finite size using switches and delay lines, with erasure control coding at the packet level.

Keywords: erasure control coding, error control coding, networking, optical networking, priority queues, queueing, switching

1. Introduction

Packet switched communication networks need to resolve contention for links by buffering some of the contending packets. Optical networks with wavelength division multiplexing can already support data rates of several terabytes per second [15]. Unfortunately, there do not exist buffering strategies for optical packets that have the flexibility of their electronic counterparts [14]. As a result, currently deployed high speed optical packet switched networks have to transform optical packets into electronics for switching, and then convert the switched electronic packets back to optics for high speed transmission [13]. This so-called O-E-O bottleneck is one of the main stumbling blocks to the rapid deployment of all-optical networks.

* The work of A. D. Sarwate is supported by an NDSEG Graduate Research Fellowship which is sponsored by the U.S. Department of Defense.

† The work of V. Anantharam is supported by ONR grant No. N00014-1-0637, DARPA grant No. N66001-00-C-8062, and by NSF grant No. ECS 0123512.

There have been many schemes proposed for mimicking buffering for optical packets, almost all of which involve using a system comprised of an optical switch and delay lines to hold the packets. In this paper we will call such a system a *switching system*. A survey of this work is available in [10]; see also [14], and the more recent work [6] (as referenced in [7]). All of these schemes are ad-hoc in nature, and the extent to which they are successful in emulating buffering for arbitrary traffic patterns is rather unclear. Our purpose in this paper is to formulate and investigate from first principles the problem of emulating buffered queueing disciplines with a switching system.

2. Problem formulation

We first describe in detail the dynamics of the *priority queue* of size K with gated service that we seek to emulate. The queue is assumed to start empty at time 0[1]. At each time there may or may not be an arrival request and there may or may not be a departure request. Each arrival request has an associated integer called its *priority*. The queue operates as follows :

- If there is no arrival request and no departure request the packets that were in the queue continue to remain in the queue.
- If there is no arrival request and there is a departure request, then among the packets in the queue (if any) the oldest one among those with the highest priority is released from the queue to serve the departure request. The other packets remain in the queue.
- If there is an arrival request which has lower priority than j of the packets currently in the queue and there is no departure request, and if the total number of packets currently in the queue is $k < K$, the arriving packet is admitted into the queue. The packets that were in the queue remain in the queue.
- If there is an arrival request and there is no departure request and the total number of packets in the queue is K, the arriving packet is rejected, irrespective of its priority[2]. The packets that were in the queue remain in the queue.

[1] The modifications required to derive our results without this assumption are straightforward.

[2] It is possible to modify our results to deal with the case that the arriving packet displaces the lowest priority packet. See Section 5 for this extension.

– If there is an arrival request which has higher priority than all the packets currently in the queue and there is a departure request, the departure request is served by the arriving packet and the packets that were in the queue remain in the queue.

– If there is an arrival request which has lower priority than j of the packets currently in the queue, $j > 0$, and there is a departure request, the departure request is served by the packet in the queue that is the oldest among the highest priority packets in the queue. The arriving packet is admitted into the queue and the packets that were in the queue, other than the one that departed, remain in the queue.

It is not hard to check that both a finite buffer First-In-First-Out (FIFO) queue and a finite buffer Last-In-First-Out (LIFO) queue [3] are priority queues.

Consider a time t at which there are $0 \leq k \leq K$ packets in the queue. It is convenient to think of each packet, if any, as carrying a *tag* at time t from among the numbers 1 through k. Packets of higher priority have smaller tags than those of lower priority, and among among packets of the same priority older packets have smaller tags than younger packets. From the dynamics of the queue, as described above, one sees that the relative order of the tags of the packets in the queue never changes from one time to another. It is also convenient to define a *next-tag* at time t for each packet that is in the queue at time t and the arriving packet at time t (if any). For a packet that is present in the queue both at time t and at time $t + 1$ its next-tag at time t equals its tag at time $t + 1$. If a packet is released from the queue at time t to serve a departure request at time t (if any) its next-tag at time t equals 0. An arriving packet at time t (if any) is assigned next-tag 0 at time t if either it is rejected from a full queue or if it is the one that serves the departure request at time t (if any). An arriving packet at time t (if any) that is admitted into the queue has its next-tag at time t equal to its tag at time $t + 1$ (as a member of the queue).

We turn now to switching systems. The switching systems we consider in this paper are as shown in Figure 2. Packets arrive along a distinguished input to the switch (indexed by 0), called the arrival input; they exit along a distinguished output of the switch (indexed by 0), called the departure output. The remaining M inputs and outputs are connected in pairs via delay lines of lengths $L_1, L_2, \ldots, L_M$. We assume that the switching system is synchronous at the packet level,

[3] In the LIFO case, if there is an arrival request into a full queue when there is no departure request, it is the arriving packet that is rejected.

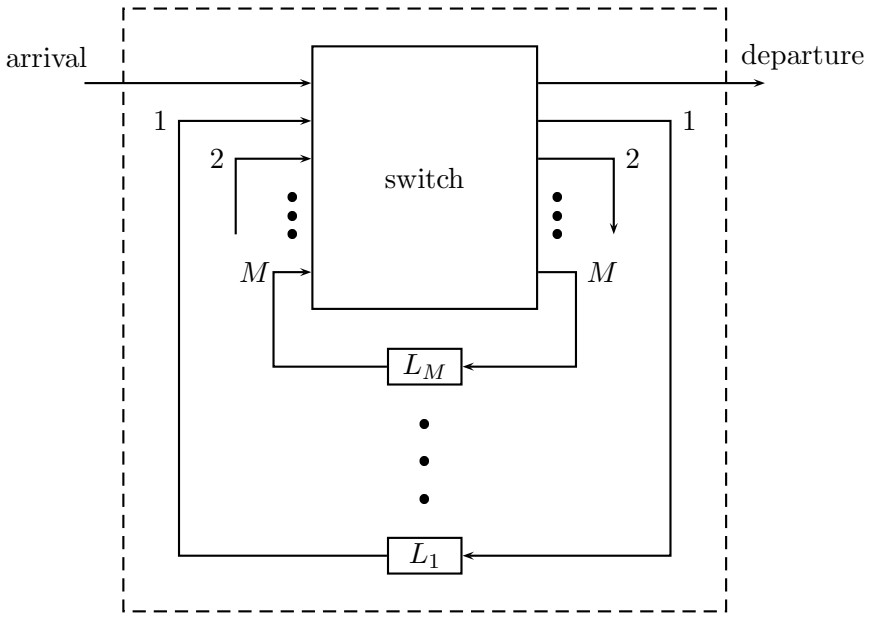

Figure 1. A queue implemented as a switch with delay lines. From the view of the input/output the system in the dashed box works like a queue.

and that all packets have the fixed length 1. At each time step t, the following events happen in order:

1. A packet may arrive at the arrival input. Also, a departure request may arrive.

2. The packets in the delay lines move forward by one slot. The switch implements a one-to-one mapping (matching) from its $M+1$ inputs to its $M+1$ outputs. Thus the arriving packet, if any, and the packets at the ends of the M delay lines, if any, are switched into the departure output and the beginnings of the M delay lines. The matching used by the switch at any time t is in principle allowed to depend on the entire history of the arrival process up to and including time t and the history of the previous matchings used up to and including time $t-1$. A packet that is switched into the departure output leaves the switching system.

We now define what we mean by saying that a switching system *exactly* emulates a priority queue of size K. We start with an empty priority queue of size K at time 0 on the one hand, and an empty

switching system at time 0 on the other hand. We are given an arbitrary sequence of departure requests and an arbitrary sequence of arrival requests with associated priorities, which we think of as applied to the queue on the one hand and to the switching system on the other hand. We will say that the switching system exactly emulates the priority queue up to time t if for each time $0 \leq s \leq t$ the packets that are in the priority queue at time s are identical to those that are in the switching system at time s and the packet that is placed on the departure output of the switching system at time s is identical to the one that departs from the queue at time s or to the one that is rejected from the queue at time s[4]. We will say that the switching system exactly emulates the priority queue for all time if it emulates the priority queue up to time t for each $t \geq 0$.

We measure the complexity of the switching system for a target queue depth K by the size $M + 1$ of the switch. We are interested in how small we can make M as a function of K. In this paper, we will show that $M \geq \lceil \log(K-1) \rceil$ and provide a construction with $M = O(\sqrt{K})$.

3. A lower bound of $\lceil \log(K-1) \rceil$

We now establish a lower bound on the switch size in any exact emulation. In this section we will assume that the lengths of the delay lines are ordered so that $L_1 \leq L_2 \leq \cdots \leq L_M$[5]. There must be at least enough room in the delay lines for K packets, so we must have

$$\sum_{i=1}^{M} L_i \geq K \ . \tag{1}$$

We claim that a packet with next-tag k at time t cannot be switched at time t into a delay line with length more than k. Suppose to the contrary that at time t, the packet with next-tag k is switched into a delay line of length more that k. Suppose furthermore that at times $t+1, t+2, \ldots t+k$, there are departure requests and no arrivals. Then at time $t+k$, this packet will have next-tag 0, but will not be available to

[4] At the times when an arriving packet is rejected from the full priority queue when there is no departure request, in the emulation we treat it as being placed on the departure output of the switching system. This is just a convenient way of representing its rejection. In a practical implementation, in this case, the arriving packet would simply be erased, for instance by disconnecting the departure output from the rest of the world.

[5] This assumption applies only in this section; for instance, it does not apply in the next section.

be switched to the output line of the switching system, so the switching system cannot correctly emulate the priority queue.

This observation leads to the following lemma, which applies to any switching system that exactly emulates a priority queue of size K :

Lemma 1. *$L_1 = 1$. If $K \geq 2$, then for all $m \geq 1$ such that*

$$\sum_{i=1}^{m} L_i \leq K - 1, \tag{2}$$

we have

$$L_{m+1} \leq 1 + \sum_{i=1}^{m} L_i. \tag{3}$$

Proof. First note that the statement of the lemma makes sense, since if (2) holds then, by (1), line $m+1$ must exist. We must have $L_1 = 1$ because if the switching system is empty at time t and there is an arrival request and no departure request at this time, then the arriving packet must, by our observation above, be assigned to a delay line of length 1. Thus there has to be at least one delay line of length 1, so $L_1 = 1$.

Now suppose that (2) holds but (3) does not hold, for some $m \geq 1$. Let j denote $\sum_{i=1}^{m} L_i$. Note that $1 \leq j \leq K-1$ under our assumptions. Suppose the switching system has j packets in it at time t and there are no arrivals and no departure requests at times $t, t+1, \ldots, t+j-1$. From our observation above, at time $t+j$ the switching system has j packets which are all in the delay lines numbered $1, 2, \ldots, m$. Thus these delay lines are "full". Suppose that at time $t+j$ there is an arrival request but no departure request, and that this arrival request has lower priority than all the packets in the switching system at time $t+j$. The new packet then has next-tag $j+1$. Since the lines $1, 2 \ldots, m$ are full at time $t+j$, the total number of packets to be switched back into the delay lines at time $t+j$ is $m+1$. Hence, at least one packet has to be switched into a line numbered $m+1$ or higher. The switching system is therefore forced to assign a packet with next-tag $j+1$ or less to a line with length $j+2$ or higher. But by our observation, this implies that the switching system cannot exactly emulate a priority queue of size K.

Using Lemma 1 we get our lower bound :

Proposition 1. $M \geq \lceil \log(K-1) \rceil$.

Proof. Let $K = 2^A + 1$. From Lemma 1, starting with $L_1 = 1$, we see that $L_2 \leq 2$, $L_3 \leq 4$, and so on, through to $L_A \leq 2^{A-1}$. This tells us that $\sum_{m=1}^{A} L_m \leq 2^A - 1$. However, we have $\sum_{m=1}^{M} L_m \geq K = 2^A + 1$; this is (1). Hence $M \geq A+1 \geq \log K$. Since the smallest possible M is a nondecreasing function of K – any switching system that can emulate of priority queue of a given size can emulate one of smaller size – it follows that $M \geq \lceil \log(K-1) \rceil$.

4. An upper bound of $O(\sqrt{K})$

In this section we describe an algorithm to emulate a priority queue of size K using a switching system with a switch of size $M = O(\sqrt{K})$. Since a switching system emulating a priority queue of a given size can also emulate one of any smaller size, it suffice to consider only the case where K is of the form $\frac{1}{2}N(N+1)$ for some $N \geq 1$ and to show that a priority queue of size K can be emulated by a switching system with an $(M+1) \times (M+1)$ switch with $M = 2N-1$. The architecture for the switching system we construct is given in Figure 2 for the case $N = 5$. This figure shows the lengths of the 9 delays lines; the switch itself is a 10×10 switch, since it also has an arrival input and a departure output. It emulates a priority queue of size 15.

For a more realistic assessment of our construction, note that a 128×128 switch can be used to emulate a priority queue of size 2080, which is bigger than 2048, a quite interesting size in practice[6]. The switch has one arrival input and one departure output. Of the 127 delay lines in the switching system, the first 64 have length equal to their index, and the next 63 are of length 1. The longest delay line used is of length 64. The total length needed for all the delay lines is 2143, so the overhead, in terms of this metric, as compared to the absolute lower bound of 2080, as given by (1), is negligible.

Let $\alpha(k)$ be defined as

$$\alpha(k) = -\frac{1}{2} + \frac{1}{2}\sqrt{1+8k}. \tag{4}$$

Then $\alpha(k)$ satisfies $k = \frac{1}{2}\alpha(k)(\alpha(k)+1)$, so if $K = \frac{1}{2}N(N+1)$, as we assume, then $N = \alpha(K)$. We will show how to emulate a priority queue of size K with a switching system that uses a switch of size $2N \times 2N$, so $M = 2N - 1$. We partition the set of delay lines into two sets of size N and $N-1$ respectively : the set of *regular lines*

[6] Commonly, 2048 is called "2K".

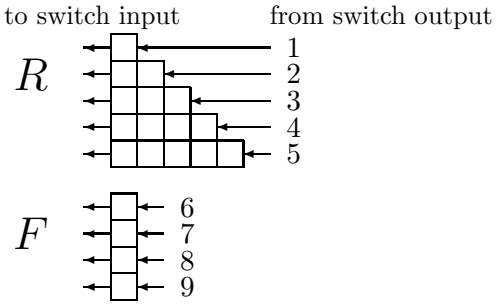

Figure 2. A queue of size K = 15. With reference to Figure 2, only the portion below the switch inside the dashed box is being shown. Here $M = 9$.

R and the set of *overflow lines* F. The lines in R have lengths $L_1 = 1, L_2 = 2, \ldots L_N = N$, and the lines in F all have length 1. That is, $L_{N+1} = 1, L_{N+2} = 1, \ldots L_{2N-1} = 1$. Note that, with this construction, the total length of the delay lines needed, for arbitrary K, is

$$\sum_{m=1}^{M} L_m = \frac{1}{2}\lceil\alpha(K)\rceil(\lceil\alpha(K)\rceil + 1) + \lceil\alpha(K)\rceil - 1 = K + O(\sqrt{K}) \ .$$

To start with we are going to consider a sloppy switching system (SSS), which uses a much bigger switch, with the regular lines as above, but with a total of $K - 1$ overflow lines, each of which is of length 1. With some abuse of notation, we will continue to use R for the set of regular lines and F for the set of overflow lines in the SSS. We consider only $K = \frac{1}{2}N(N+1)$ for some $N \geq 1$, so the SSS has N regular lines, of lengths $1, \ldots, N$ respectively, and $K - 1$ overflow lines, each of length 1, and it needs an $(N + K) \times (N + K)$ switch. We will describe a switching algorithm for the SSS and prove that it exactly emulates a priority queue of size K. We will then argue that at every time at most $N - 1$ overflow lines are used in the SSS, so it unnecessary to have so many overflow lines. Thus, the same algorithm, applied to the original switching system, will ensure that it also exactly emulates a priority queue of size K.

We now describe a class of algorithms for the SSS that will make it emulate a priority queue of size K. Assume that the SSS has been correctly emulating the priority queue up to time t, so we can talk about the tag and post-tag of packets at time t. The algorithms we consider are of the following type : at time t, the algorithm computes a number $1 \leq T(t) \leq N$. The regular line with this index is called the *trapping line*. At time t, the algorithm places the packet with next-tag 0, if any, in the departure line, and places the remaining packets that need to

be switched, if any, in order of increasing next-tag, in the regular lines of least index, up to and including the trapping line, and places any remaining packets that still need to be switched into the overflow lines, one to each overflow line.

Note that, in the SSS, *irrespective of how $T(t)$ is computed*, an algorithm of this type will continue to exactly emulate the priority queue up to time $t+1$, and the packet with tag 1 at time $t+1$ will be in the regular line of least index at time $t+1$. This follows from the observation that the packet with next-tag 0, if any, is always placed in the departure line, and any other packet is always placed in a line of length that is no bigger than its next-tag.

We will now build up to a particular way of computing $T(t)$, given in equation (9) below. Let $a(t)$ denote that indicator that there is an arrival request at time t. Let $d(t)$ denote the indicator that there is a departure request at time t. Let $\omega_Q(t)$ denote the total number of packets in the SSS at time t. This does not include the arrival request, if any. We have already proved that that the SSS exactly emulates the priority queue of size K, so we know that $\omega_Q(t)$ evolves like the queue size in the priority queue, i.e.

$$\omega_Q(t+1) = \min(K, [\omega_Q(t) + a(t) - d(t)]^+) \ . \tag{5}$$

For future convenience, let $\omega_Q^+(t)$ denote $\omega_Q(t+1)$. Thus

$$\omega_Q^+(t) = \min(K, [\omega_Q(t) + a(t) - d(t)]^+) \ . \tag{6}$$

Let $\omega_F(t)$ denote the number of packets in the overflow lines of the SSS at time t. Let $S(t)$ denote the largest index among the regular lines that have a packet at time t, if any, with $S(t) = 0$ if none of the regular lines have packets at time t. Let $\delta(t)$ denote the number of packets that need to be circulated back into the $N + (K-1)$ delay lines of the SSS at time t. Note that this does not include the packet placed in the departure output of the switch in the SSS, if any. Then we have

$$\delta(t) \leq [S(t) + \omega_F(t) + a(t) - d(t)]^+ \ . \tag{7}$$

This comes from the fact that every packet that needs to be recirculated as well as the packet that departs (if any) is either the arriving packet (if any), a packet at the head of one of the regular lines 1 through $S(t)$, or a packet in one of the occupied overflow lines (of which there are $\omega_F(t)$).

Let $P(t)$ denote the number of shortest regular lines that would be needed to pack $\omega_Q^+(t)$, i.e.

$$P(t) = \lceil \alpha(\omega_Q^+(t)) \rceil \ . \tag{8}$$

The trapping line is given by setting

$$T(t) = \max\{1, S(t), P(t)\} \ . \tag{9}$$

We have $0 \leq S(t) \leq N$, since there are only N regular lines. We have $0 \leq P(t) \leq N$, since we have $\omega_Q^+(t) \leq K$. Hence we have $1 \leq T(t) \leq N$. The algorithm used by the SSS is as described above, with this particular choice for $T(t)$. Thus, of the $\delta(t)$ packets that need to be recirculated into the delay lines at time t, up to exactly $T(t)$ will be put in the shortest $T(t)$ delay lines in increasing order of next-tag, and any others will be placed in the overflow lines, one to a line. We therefore have

$$S(t) \leq T(t-1) \ , \tag{10}$$

$$\omega_F(t) = [\delta(t-1) - T(t-1)]^+ \ , \tag{11}$$

and

$$S(t) < T(t-1) \Longrightarrow \omega_F(t) = 0 \ . \tag{12}$$

For the convenience of the reader, in the following table we give a glossary of the notation we have introduced so far. All quantities are at time t.

$\omega_Q(t)$	Queue size in the priority queue being emulated.
$a(t)$	Indicator of arrival request.
$d(t)$	Indicator of departure request.
$\omega_Q^+(t)$	Projected queue size one step later, i.e. $\omega_Q(t+1)$.
$\omega_F(t)$	Number of packets in the overflow lines.
$S(t)$	Largest index of a regular line with a packet (equals 0 if all regular lines are empty).
$P(t)$	Number of shortest regular lines needed to pack $\omega_Q^+(t)$ (equals 0 if $\omega_Q^+(t) = 0$).
$\delta(t)$	Number of packets needing recirculation into the delay lines.
$T(t)$	Trapping line.

Our aim is to show that with $T(t)$ chosen as in equation (9), we have $\omega_F(t) \leq N-1$ for all t. We do this in Proposition 2 after first proving a sequence of lemmas. The SSS and the priority queue of size K that is being emulated are assumed to start empty at time $t = 0$. Thus all the variables defined in the preceding table may be taken to be zero for all $t \leq 0$, the exceptions being $a(0)$ and $d(0)$, which are given at $t = 0$, $\omega_Q^+(0)$, which equals $(a(0) - d(0))^+$, $P(0)$, which equals $\lceil \alpha(\omega_Q^+(0)) \rceil$,

$\delta(0)$, which equals $(a(0) - d(0))^+$, and $T(0)$, which will equal 1 for all $t \leq 0$.

Lemma 2. *For all t we have*

$$\omega_F(t) \leq \omega_F(t-1) + 1 \ . \tag{13}$$

Proof.
Starting with equation (7) we have

$$\begin{aligned} \delta(t-1) &\leq [S(t-1) + \omega_F(t-1) + a(t-1) - d(t-1)]^+ \\ &\leq S(t-1) + \omega_F(t-1) + 1 \\ &\leq T(t-1) + \omega_F(t-1) + 1 \ , \end{aligned} \tag{14}$$

where the last step uses equation (9). Now, starting with equation (11) we have

$$\begin{aligned} \omega_F(t) &= [\delta(t-1) - T(t-1)]^+ \\ &\leq [\omega_F(t-1) + 1]^+ \\ &= \omega_F(t-1) + 1 \ , \end{aligned}$$

where the inequality comes from equation (9). This completes the proof.

Lemma 3. *For all t we have*

$$T(t) < T(t-1) \Longrightarrow \omega_F(t) = 0 \ . \tag{15}$$

Proof.
From equation (9) we know that $T(t) < T(t-1) \Longrightarrow S(t) < T(t-1)$. From equation (12) this further implies that $\omega_F(t) = 0$. This completes the proof.

Lemma 4. *For all t, if $T(t-1) < T(t)$ then $T(t) = T(t-1) + 1$ and $\omega_Q^+(t) = \omega_Q^+(t-1) + 1$.*

Proof.
We have

$$\max(1, S(t)) \leq \max(1, T(t-1)) = T(t-1) \ ,$$

where the first step is from equation (10) and the second step is from equation (9). From the definition of $T(t)$ in equation (9) it follows that

$$T(t) = \max(1, S(t), P(t)) \leq \max(T(t-1), P(t)) \ .$$

It then follows that

$$T(t) > T(t-1) \Longrightarrow T(t) = P(t) > T(t-1) \geq P(t-1) \ .$$

Since $\omega_Q^+(\cdot)$ and $P(\cdot)$ can increase by at most 1 at any time, this completes the proof of the lemma.

Lemma 5. *For all t, if $T(t-1) < T(t)$ then $S(t) = T(t-1)$.*

Proof.
From equation (10) we know that $S(t) \leq T(t-1)$. Assume that $S(t) < T(t-1)$. From equation (12) this implies that $\omega_F(t) = 0$. From the algorithm used by the switch in the SSS to recirculate packets, we have that $\omega_F(t) = 0$ implies $\omega_Q(t) \leq \frac{T(t-1)(T(t-1)+1)}{2}$, because all the packets in the system at time t would be in the shortest regular lines up to and including the line of index $T(t-1)$. However, since we have also assumed that $S(t) < T(t-1)$, it follows that $\omega_Q(t) < \frac{T(t-1)(T(t-1)+1)}{2}$. This then implies that $\omega_Q^+(t) \leq \frac{T(t-1)(T(t-1)+1)}{2}$, because $\omega_Q^+(\cdot)$ can increase by at most 1 at any time. Hence $\alpha(\omega_Q^+(t)) \leq T(t-1)$, and so $P(t) = \lceil \alpha(\omega_Q^+(t)) \rceil \leq T(t-1)$. It then follows that $T(t) = \max(1, S(t), P(t)) \leq T(t-1)$. We have thus completed the proof of the lemma.

Lemma 6. *For all t, if $T(t-1) < T(t)$ then $S(t+1) = T(t)$.*

Proof.
From equation (10), we always have $S(t+1) \leq T(t)$. If $S(t+1) < T(t)$, then, from equation (12), we would have $\omega_F(t+1) = 0$. From the algorithm used by the switch in the SSS, this means that all packets in the queue at time $t+1$ are in the regular lines of least index up to and including the line of index $T(t) - 1$, if any[7]. We then have $\omega_Q^+(t+1) \leq \frac{(T(t)-1)((T(t)-1)+1)}{2}$. Since we assumed that $T(t-1) < T(t)$ and since $T(\cdot)$ can increase by at most 1 at any time, what this says is that $\omega_Q^+(t+1) \leq \frac{T(t-1)((T(t-1)+1)}{2}$. From this, exactly as in the proof of Lemma 5, we conclude that $T(t) \leq T(t-1)$, a contradiction. This concludes the proof of the lemma.

Lemma 7. *For all t, if $T(t-1) < T(t)$ then $S(t+1) = S(t+2) = \ldots = S(t+T(t)) = T(t)$ and $T(t) = T(t+1) = \ldots = T(t+T(t)-1)$.*

[7] If $T(t) = 1$ the conclusion would be that there no packets in the queue at time $t+1$.

Proof.

In Lemma 3 we proved that $T(t-1) < T(t)$ implies that $T(t) = T(t-1)+1$ and that $\omega_Q^+(t) = \omega_Q^+(t-1)+1 = \omega_Q(t)+1$. It follows that $\omega_Q^+(t) = \frac{T(t-1)(T(t-1)+1)}{2}+1$. In Lemma 6 we proved that $S(t+1) = T(t)$. Intuitively, what has happened is that the algorithm has placed, at time $t+1$, exactly one packet in the "new" regular line, of index $T(t)$ that it has just "opened" at time $t+1$. Now, since the queue size can increase by at most 1 at each time step, we have $\omega_Q^+(t+i) \leq \frac{T(t-1)(T(t-1)+1)}{2}+i+1$. So, for $0 \leq i \leq T(t)-1$, we have $P(t+i) \leq T(t)$ from equation (8). Since $S(t+1) = T(t)$ and $P(t+1) \leq T(t)$, we have $T(t+1) = T(t)$. The packet that was placed in the line of index $T(t)$ at time $t+1$ must continue to be in the same line at time $t+2$. Hence $S(t+2) = T(t)$. If $T(t) = 2$, we have completed the proof. Otherwise, we may continue in this manner to argue that since $S(t+2) = T(t)$ and $P(t+2) \leq T(t)$, we have $T(t+2) = T(t)$. If $T(t) \geq 3$, the packet that was placed in the line of index $T(t)$ at time $t+1$ must continue to be in the same line at time $t+3$, so we have $S(t+3) = T(t)$. In the general case, the last step is to use the previously proved fact that $S(t+T(t)-1) = T(t)$ with $P(t+T(t)-1) \leq T(t)$ to conclude that $T(t+T(t)-1) = T(t)$. Because the packet that was placed in the line of index $T(t)$ at time $t+1$ must continue to be in the same line at time $t+T(t)$ we conclude that $S(t+T(t)) = T(t)$.

Lemma 8. *For all t, if $T(t-1) < T(t)$ then $\omega_F(t+i) \leq \omega_F(t+i-1)$ for all $i = 1, 2, \ldots T(t)$.*

Proof.

In Lemma 5 we proved that $T(t-1) < T(t)$ implies that $S(t-1) = T(t)$. In Lemma 3 we proved that $T(t-1) < T(t)$ implies that $T(t) = T(t-1)+1$ and that $\omega_Q^+(t) = \omega_Q^+(t-1)+1 = \omega_Q(t)+1$. This implies that $a(t) = 1$ and $d(t) = 0$. We may now start from equation (7) to write

$$\begin{aligned} \delta(t) &\leq [S(t) + \omega_F(t) + a(t) - d(t)]^+ \\ &= S(t) + \omega_F(t) + 1 \\ &= T(t-1) + \omega_F(t) + 1 \\ &= T(t) + \omega_F(t) \ . \end{aligned}$$

From equation (11) we now have $\omega_F(t+1) = [\delta(t) - T(t)]^+ \leq \omega_F(t)$.

From Lemma 7, using the fact that there is no packet that is present at the output of the regular line of index $T(t)$ at times $t+i-1$ for $i = 2, \ldots, T(t)$, we have $\delta(t+i-1) \leq (T(t)-1) + \omega_F(t+i-1) + 1$

for all such i. Further, Lemma 7 tells us that $T(t+i-1) = T(t)$ for all such i, so $\omega_F(t+i) = [\delta(t+i-1) - T(t+i-1)]^+ \leq \omega_F(t+i-1)$ for all such i. This completes the proof of the lemma.

We are now able to prove the claim that the SSS never needs to use more than $N-1$ overflow lines.

Proposition 2. *We have $\omega_F(t) < T(t)$ for all t. In particular, we have $\omega_F(t) \leq N-1$ for all t.*

Proof.

The proof is by induction on $T(t)$. To start with, it is true for all t with $T(t) = 1$. To see this, consider two cases. First, if $T(t-1) = 1$, then $\omega_Q^+(t-1) \leq 1$ so $\omega_F(t) = 0$. Second, if $T(t-1) \geq 2$, then since $S(t) \leq 1$, we have $\omega_F(t) = 0$. In either case, we are done.

Next, we show that the claim is true for all t with $T(t) = 2$. We have to consider three cases. First, if $T(t-1) = 1$, then $\omega_Q^+(t-1) \leq 1$ so $\omega_F(t) = 0$. Second, if $T(t-1) \geq 3$, then since $S(t) \leq 2$, we have $\omega_F(t) = 0$. Third, if $T(t-1) = 2$, then $\omega_Q^+(t-1) \leq 3$, so if $S(t) = 2$, we have $\omega_F(t) \leq 1$, while if $S(t) < 2$, we have $\omega_F(t) = 0$. In all three cases, we are done.

Assume now that the claim has been established for all t with $T(t) \leq r-1$, where $r-1 \geq 2$. Consider t with $T(t) = r$. We again consider three cases. First, if $T(t-1) \leq r-1$, then $T(t-1) = r-1$ and $\omega_F(t-1) < r-1$ so $\delta(t-1) < 2(r-1)+1$, so $\omega_F(t) \leq r-1$. Second, if $T(t-1) \geq r+1$, then since $S(t) \leq r$ we have $\omega_F(t) = 0$.

The third case, when $T(t-1) = r$, is the most interesting. Suppose $\omega_F(t) \geq r$. Then, as we proved in Lemma 2, we have $\omega_F(t-i) > 0$ for $0 \leq i \leq r-1$. Hence we have

$$T(t-r) \leq T(t-r+1) \leq \ldots \leq T(t) = r \ .$$

Suppose there is $1 \leq \tau \leq r$ with $T(t-\tau) < T(t-\tau+1) = r$. Then by Lemma 8, $\omega_F(t-\tau+1) \geq \omega_F(t-\tau+2) \ldots \geq \omega_F(t) = r$, a contradiction. On the other hand, if there is no such τ, i.e. if we suppose that

$$T(t-r) = T(t-r+1) = \ldots = T(t) = r \ ,$$

then $\omega_Q(t) \geq \frac{r(r+1)}{2} + r - 1 > \frac{r(r+1)}{2}$, so $T(t) \geq r+1$, which is also a contradiction. Thus we are done in all three cases and have completed the proof of the proposition. This implies that there was no need to have as many overflow lines as provided in the SSS : the original switching system would have done as well. This completes our proof that $M = O(\sqrt{K})$.

5. A modified packet rejection scheme

In this section we describe an extension to the previous construction that allows arriving packets to force the switching system to drop the lowest priority packet rather than rejecting the arriving packet. As before, at each time t we will talk about the *tag* for each packet in the queue at time t, and we will talk about the *next-tag* for packets that are in the queue at time t and for the arriving packet at time t (if any). The tag at time t is defined exactly as before : higher priority packets have smaller tags than lower priority packets, and among packets of the same priority, older packets have smaller tags than younger packets. The next-tags at time t will be computed in terms of the tags at time $t+1$ exactly as for the preceding definition of priority queues, except in one case : if the queue is full at time t, there is no departure request at time t, and there is an arrival at time t that has priority higher than precisely j of the packets currently in the queue $0 \leq j \leq K-1$, then the packet that had tag K at time t will get next-tag 0 at time t and the packet that arrived at time t will be given next-tag $j+1$ at time t. The point, of course, is that it is now the arriving packet that gets admitted by displacing a lowest priority packet currently in the (full) queue (which we take to be the youngest of the lowest priority packets).

We will show how to emulate a priority queue of size $K = \frac{N(N+1)}{2}$, in the new sense, with a switching system having a switch of size $N + (N-1) + (N-1)$. The method we propose is like that in the preceding discussion, except that we require $N-1$ additional delay lines of length 1 each and a slight modification to the switch operation.

Let us add to the previous construction a set E of $N-1$ delay lines of length 1 each. We call this the set the *end lines*. To start with, we again use a sloppy switching system (SSS) with N regular lines, $K-1$ overflow lines, and the $N-1$ end lines, prove that the SSS exactly emulates the priority queue of size K and then prove that no more than $N-1$ overflow lines ever need to be used.

The operation of the new SSS is similar to that in the previous section. At each time t the algorithm computes a trapping line $T(t) \geq 1$. Let $A = \{K-N+2, K-N+3, \ldots K\}$. The algorithm places a packet with next-tag 0, if any, in the departure line, places any packets with next-tags in the set A into the lines in E, places $T(t)$ packets in increasing order of next-tag into the regular lines of smallest index, and places the remaining packets in the overflow lines. In this new operation, the end lines E are reserved for the packets with tags in A.

Since a packet is always placed in a line of length no bigger than its next-tag, we can see that packet with highest priority is always available at the input of the switch. The new requirement we have is

that the packet with tag K, if any, should be available at the input of the switch. We will prove this below in Lemma 9. This lemma holds true *irrespective of how the trapping line is computed.*

Lemma 9. *At all times t for which queue is full, the packet with tag K is always available at the input of the switch.*

Proof. Suppose that the queue is full at time t and the packet with tag K is not available at the input of the switch. Then it cannot be in the overflow lines F or the end lines E since they are of length 1. The packet must be in a line of length $j > 1$ in R. Since this line has length less than or equal to N and the packet is not at the input of the switch, it must have been switched into this line at a time s where $t - N + 1 \leq s \leq t - 1$. Thus the packet's next-tag at time s, i.e. its tag at time $s + 1$, must have been at most $K - N + 1$. Since the tag of a packet can increase by at most 1 at each time, its current tag is at most $(K - N + 1) + (t - (s + 1)) \leq K - 1$, a contradiction. Thus the packet with tag K must be available at the input of the switch.

We have now proved that the new SSS correctly emulates a priority queue of size K in the new sense. The proof that $N - 1$ overflow lines are sufficient proceeds along the same lines as in the last section by showing that, with a specific choice of the trapping line given below, the SSS needs to use no more than $N - 1$ of the overflow lines.

Let $\omega_Q(t)$, $\omega_Q^+(t)$, $a(t)$, and $d(t)$ be as before. Let $\omega_E(t)$ denote the number of packets in E. Let $S(t)$ denote the longest length among the regular lines that have a packet at time t, if any, with $S(t) = 0$ if none of the regular lines has a packet at time t. Let $P(t)$ be defined as in equation (8). Let $T(t)$ be defined as in equation (9). Let $\hat{\delta}(t)$ denote the number of packets that need to be circulated back into the $N + (K - 1)$ delay lines in $R \cup F$ at time t. Note that this does not include packets that need to be recirculated into the end lines, if any. Then it is clear that the following holds:

$$\hat{\delta}(t) \leq [S(t) + \omega_F(t) + 1 - d(t)]^+ \ . \tag{16}$$

This bound is different from the analogous bound (7) of the preceding section, because we may have the case where a departure forces a packet formerly in E to be switched into $R \cup F$. It is also clear that we have

$$S(t) \leq T(t-1) \ , \tag{17}$$

$$\omega_F(t) = \left[\hat{\delta}(t-1) - T(t-1)\right]^+ \ , \tag{18}$$

$$S(t) < T(t-1) \implies \omega_F(t) = 0 \ . \tag{19}$$

The following three lemmas are the analogs of Lemmas 2, 3, and 4 respectively. The proof only depends on equations (16)–(19), in exactly the same way in which the proofs of those lemmas depend on equations (7) and (10)–(12) respectively, so they follow immediately.

Lemma 10. *For all t we have*

$$\omega_F(t) \leq \omega_F(t-1) + 1. \tag{20}$$

Lemma 11. *For all t we have*

$$T(t) < T(t-1) \Longrightarrow \omega_F(t) = 0. \tag{21}$$

Lemma 12. *For all t, if $T(t-1) < T(t)$ then $T(t) = T(t-1)+1$ and $\omega_Q^+(t) = \omega_Q^+(t-1)+1$.*

The next three lemmas correspond to Lemmas 5, 6, and 7 respectively. Their proofs are given in the appendix.

Lemma 13. *For all t, if $T(t-1) < T(t)$ then $S(t) = T(t-1)$.*

Lemma 14. *For all t, if $T(t-1) < T(t)$ then $S(t+1) = T(t)$.*

Lemma 15. *For all t, if $T(t-1) < T(t)$ then $S(t+1) = S(t+2) = \ldots = S(t+T(t)) = T(t)$ and $T(t) = T(t+1) = \ldots = T(t+T(t)-1)$.*

The three lemmas above can be used to show Lemma 16, which is the analog of Lemma 8. The proof is given in the appendix.

Lemma 16. *For all t, if $T(t-1) < T(t)$ then $\omega_F(t+i) \leq \omega_F(t+i-1)$ for all $i = 1, 2, \ldots T(t)$.*

With these lemmas one can prove the following analog to Proposition 2. The proof is given in the appendix.

Proposition 3. *We have $\omega_F(t) < T(t)$ for all t. In particular, we have $\omega_F(t) \leq N-1$ for all t.*

This proposition establishes that is possible to emulate a priority queue of size K in the new sense, with $\alpha(K) + 2(\alpha(K)-1) = O(\sqrt{K})$ delay lines. The total length of the lines used is $\frac{\alpha(K)(\alpha(K)+1)}{2} + 2(\alpha(K)-1) = K + O(\sqrt{K})$.

6. Concluding remarks

We have considered *exact* emulation of finite buffer priority queues using a switching system comprised of a switch with delay lines. While this problem is interesting and appears challenging, we have the more general aim of motivating the study from first principles of the problem of *approximate* emulation of queues by structures that can be implemented in all-optical technology, in particular switching systems. In this concluding section we briefly sketch an idea for building an all-optical packet switched network by approximate emulation of finite buffer priority queues, with end to end erasure control coding at the packet level.

Our idea is to view the pattern of drops of packets in a flow as a parameter of quality of service. An approximate emulation of a queueing discipline is thus allowed to occasionally fail to deliver a packet in response to a departure request, but we aim to give a guarantee on the dropping pattern. Since an end to end guarantee on the dropping pattern could be built up from guarantees at the switches along the path of the flow, this architecture decouples the network level design problem into a set of switch level problems.

We propose to characterize a dropping pattern by s, an integer, and p, a number in $(0, 1)$, such that, for each $n \geq 1$, over any stretch of n successive demand requests, the total number of departure requests that are not satisfied is at most $s + pn$. This characterization is analogous to that in the familiar theory of burstiness constrained flows [3,8,9,11], except that it is being applied to packet drops. One could study the problem of most efficient emulation of a priority queueing discipline by a switching system that can give an (s, p) guarantee on the dropping pattern. More generally, we may also allow for the demand requests to be satisfied with some delay. To avoid dealing with resequencing problems at the receiving end in a practical system we might insist that demand requests are served in order and insist that the maximum delay suffered by any packet is at most some integer Δ. We could then pose the problem of approximately emulating a priority queue by a system that can give a quality of service guarantee (s, ρ, Δ) on drops and delays. By this we mean that each departure request is either not satisfied or is satisfied within a delay of Δ and further that for each $n \geq 1$, over any stretch of n successive demand requests, the total number of departure requests that are not satisfied is at most $s + pn$. Other related kinds of quality of service definitions can be easily imagined and might be of interest.

A communication network is basically just a channel shared by the flows that use it to achieve the end to end communication that they

desire. In packet switched networks, packet loss can therefore be viewed as a kind of channel-introduced error, more specifically the erasure of a packet. Erasure control coding [2,4,12] can overcome this kind of error, so it makes considerable sense to study such approximate emulation problems. For other ideas that bring erasure control coding to bear in networking, see [1, 5].

Clearly there is a large gap between the upper and lower bounds we proved in this paper for the exact emulation problem. The proof of our lower bound is quite naive, and it may be possible to improve it by more careful analysis. Further, there may exist better constructions for FIFO, LIFO, or other specific priority queues. Simulations with real traffic patterns to measure the number of recirculations would help in determining the practicality of this scheme for implementation. Ultimately, construction and testing of these optical packet switches is needed to prove if this feedback mechanism is feasible in practice.

Appendix

A. Proofs

We prove the lemmas and proposition from Section 5.

Lemma 13. *For all t, if $T(t-1) < T(t)$ then $S(t) = T(t-1)$.*

Proof. Equation (17) gives $S(t) \leq T(t-1)$. Assume that $S(t) < T(t-1)$. By equation (19) we have $\omega_F(t) = 0$. Since $S(t) < T(t-1)$ and $\omega_F(t) = 0$, all of the packets in the system at time t are in the lines $1, 2, \ldots T(t-1) - 1$ in R or in E. This implies that $\omega_Q(t) \leq \frac{(T(t-1)-1)T(t-1)}{2} + \omega_E(t)$, so $\omega_Q(t) < \frac{T(t-1)(T(t-1)+1)}{2} + \omega_E(t)$. Because $T(t-1) < T(t) \leq N$, we have $T(t-1) \leq N-1$, so $\omega_Q(t) < K - N + \omega_E(t)$. Because $\omega_F(t) = 0$, this means there are at most $K - N$ packets in the switching system at time t that are not in E. This in turn implies $\omega_E(t) = 0$ because E contains packets with rank at least $K-N+2$. Thus $\omega_Q(t) < \frac{T(t-1)(T(t-1)+1)}{2}$. Thus $\omega_Q^+(t) \leq \frac{T(t-1)(T(t-1)+1)}{2}$, so $P(t) = \lceil \alpha(\omega_Q^+(t)) \rceil \leq T(t-1)$. Therefore $T(t) = \max(1, S(t), P(t)) \leq T(t-1)$, a contradiction. This completes the proof.

Lemma 14. *For all t, if If $T(t-1) < T(t)$ then $S(t+1) = T(t)$.*

Proof. From equation (17), we always have $S(t+1) \leq T(t)$. Suppose that $S(t+1) < T(t)$. Then, from equation (19), we would have $\omega_F(t+1) = 0$. From the algorithm used by the switch in the SSS, this means

that all packets in the queue at time $t+1$ are in the shortest regular lines up to and including the line of length $T(t)-1$, if any, and possibly in the end lines. We then have $\omega_Q(t+1) < \frac{(T(t)-1)(((T(t)-1)+1)}{2} + \omega_E(t+1)$. From Lemma 12 we have $T(t) = T(t-1)+1$, so $\omega_Q(t+1) < \frac{T(t-1)(T(t-1)+1)}{2} + \omega_E(t+1)$. Because $T(t-1) < T(t) \leq N$, we have $T(t-1) \leq N-1$, so $\omega_Q(t+1) < K - N + \omega_E(t+1)$. From this, exactly as in the proof of Lemma 13, we conclude that $\omega_E(t+1) = 0$, so $\omega_Q(t+1) < \frac{T(t-1)(T(t-1)+1)}{2}$, which is to say $\omega_Q^+(t) < \frac{T(t-1)(T(t-1)+1)}{2}$, so $T(t) \leq T(t-1)$, a contradiction. This concludes the proof of the lemma.

Lemma 15. *For all t, if $T(t-1) < T(t)$ then $S(t+1) = S(t+2) = \ldots = S(t+T(t)) = T(t)$ and $T(t) = T(t+1) = \ldots = T(t+T(t)-1)$.*

Proof. In Lemma 12 we proved that $T(t-1) < T(t)$ implies that $T(t) = T(t-1)+1$ and that $\omega_Q^+(t) = \omega_Q^+(t-1)+1 = \omega_Q(t)+1$. It follows that $\omega_Q^+(t) = \frac{T(t-1)(T(t-1)+1)}{2} + 1$. In Lemma 14 we proved that $S(t+1) = T(t)$. Again, the same intuition from Section 4 holds – what has happened is that the algorithm has placed, at time $t+1$, exactly one packet in the "new" regular line, of length $T(t)$ that it has just "opened" at time $t+1$. Since the queue size can increase by at most 1 at each time step, we have $\omega_Q^+(t+i) \leq \frac{T(t-1)(T(t-1)+1)}{2} + i + 1$. So, for $0 \leq i \leq T(t)-1$, we have $P(t+i) \leq T(t)$ from equation (8). Since $S(t+1) = T(t)$ and $P(t+1) \leq T(t)$, we have $T(t+1) = T(t)$. The packet that was placed in the line of length $T(t)$ at time $t+1$ must continue to be in the same line at time $t+2$. Hence $S(t+2) = T(t)$. If $T(t) = 2$, we have completed the proof. Otherwise, we may continue in this manner to argue that since $S(t+2) = T(t)$ and $P(t+2) \leq T(t)$, we have $T(t+2) = T(t)$. Since, if $T(t) \geq 3$, the packet that was placed in the line of length $T(t)$ at time $t+1$ must continue to be in the same line at time $t+3$, we have $S(t+3) = T(t)$. In the general case, the last step would be to use the previously proved fact that $S(t+T(t)-1) = T(t)$ with $P(t+T(t)-1) \leq T(t)$ to conclude that $T(t+T(t)-1) = T(t)$ and then to use that fact that the packet that was placed in the line of length $T(t)$ at time $t+1$ must continue to be in the same line at time $t+T(t)$ to conclude that $S(t+T(t)) = T(t)$.

Lemma 16. *For all t, if $T(t-1) < T(t)$ then $\omega_F(t+i) \leq \omega_F(t+i-1)$ for all $i = 1, 2, \ldots T(t)$.*

Proof. From the previous Lemma, we have $S(t-1) = T(t)$. From Lemma 12 we also have $T(t) = T(t-1)+1$ and $\omega_Q^+(t) = \omega_Q(t)+1$.

Thus $a(t) = 1$ and $d(t) = 0$. From equation (16) we can write

$$\begin{aligned}\hat{\delta}(t) &\leq [S(t) + \omega_F(t) + 1 - d(t)]^+ \\ &\leq S(t) + \omega_F(t) + 1 \\ &= T(t-1) + \omega_F(t) + 1 \\ &= T(t) + \omega_F(t) \ .\end{aligned}$$

Therefore $\omega_F(t+1) \leq \omega_F(t)$, as before.

From Lemma 15 we know that there is no packet at the output of the regular line of length $T(t)$ at time $t+i$ for $i = 2, \ldots T(t)$. Therefore we have $\hat{\delta}(t+i-1) \leq (T(t)-1) + \omega_F(t+i-1) + 1$ for $i = 2, \ldots T(t)$. Lemma 15 also tells us that $T(t+i-1) = T(t)$ for all such i, so we have $\omega_F(t+i) = [\hat{\delta}(t+i-1) - T(t)]^+ \leq \omega_F(t+i-1)$. This completes the proof of the lemma.

Proposition 3. *We have $\omega_F(t) < T(t)$ for all t. In particular, we have $\omega_F(t) \leq N-1$ for all t.*

Proof. The proof, as in the previous construction, is by induction on $T(t)$. Assume $T(t) = 1$. We have two cases. If $T(t-1) = 1$ then $\omega_Q^+(t-1) \leq 1$ so $\omega_F(t) = 0$. If $T(t-1) \geq 2$, then since $S(t) \leq 1$ we have $\omega_F(t) = 0$.

Now we show the claim for times t with $T(t) = 2$. There are now three cases. First, if $T(t-1) = 1$, then $\omega_Q^+(t-1) \leq 1$ so $\omega_F(t) = 0$. Second, if $T(t-1) \geq 3$, then since $S(t) \leq 2$ we have $\omega_F(t) = 0$. Third, if $T(t-1) = 2$, then $\omega_Q^+(t-1) \leq 3$, so if $S(t) = 2$ we have $\omega_F(t) \leq 1$, while if $S(t) < 2$, we have $\omega_F(t) = 0$. In all three cases, we are done.

Assume now that the claim has been established for all t with $T(t) \leq r-1$ for $r-1 \geq 2$. Consider times t with $T(t) = r$. There are again three cases. First, if $T(t-1) \leq r-1$ then by Lemma 12, $T(t-1) = r-1$ and by the induction hypothesis $\omega_F(t-1) < r-1$, so $\hat{\delta}(t-1) < 2(r-1)+1$ and $\omega_F(t) \leq r-1$. Second, if $T(t-1) \geq r+1$, since $S(t) \leq r$ we have $\omega_F(t) = 0$.

For the third case, assume $T(t-1) = r$ and $\omega_F(t) \geq r$. From Lemma 10, we know $\omega_F(t)$ can only increase by one each time, so $\omega_F(t-i) > 0$ for $i = 1, 2, \ldots r-1$. Thus by Lemma 11 we have

$$T(t-r) \leq T(t-r+1) \leq \ldots \leq T(t) = r \ .$$

Suppose there is a $1 \leq \tau \leq r$ such that $T(t-\tau) < T(t-\tau+1) = r$. Then by Lemma 16, $\omega_F(t-\tau+1) \geq \omega_F(t-\tau+2) \geq \ldots \geq \omega_F(t) = r$, which is a contradiction. Suppose now that there is no such τ, so that

$$T(t-r) = T(t-r+1) = \ldots = T(t) = r \ .$$

Then we have $\omega_Q(t) \geq \frac{r(r+1)}{2} + r - 1 > \frac{r(r+1)}{2}$, which implies $T(t) \geq r+1$, also a contradiction. Therefore $N-1$ overflow lines are sufficent to implement a priorty queue under the new dynamic. This concludes the proof of the proposition.

References

1. Albanese, A., J. Blorner, J. Edmonds, M. Luby, and M. Sudan: 1994, 'Priority encoded transmission'. In: *Proceedings of the 35th Annual ACM Symposium on the Foundations of Computer Science*.
2. Alon, N. and M. Luby: 1996, 'A linear time erasure-resilient code with nearly optimal recovery'. *IEEE Transactions on Information Theory* **42**(6), 1732–1736.
3. Anantharam, V. and T. Konstantopoulos: 1999, 'A methodology for the design of optimal traffic shapers in ATM networks'. *IEEE Transactions on Automatic Control* **44**(3), 583–586.
4. Blahut, R. E.: 2003, *Algebraic Codes for Data Transmission*. Cambridge, UK: Cambridge University Press.
5. Byers, J. W., M. Luby, and M. Mitzenmacher: 2002, 'A digital fountain approach to asynchronous reliable multicast'. *IEEE Journal on Selected Areas in Communications* **20**(8), 1528–1540.
6. Chang, C., D. Lee, and C. Tu: 2002, 'Recursive construction of FIFO optical multiplexers with switched delay lines'. submitted to IEEE Transactions of Information Theory.
7. Chang, C.-S., D.-S. Lee, and C.-K. Tu: 2003, 'Using switched delay lines for exact emulation of FIFO multiplexers with variable length bursts'. In: *Proceedings of IEEE INFOCOM*.
8. Cruz, R. L.: 1991a, 'A calculus for network delay I : network elements in isolation'. *IEEE Transactions on Information Theory* **37**(1), 114–131.
9. Cruz, R. L.: 1991b, 'A calculus for network delay II : network analysis'. *IEEE Transactions on Information Theory* **37**(1), 132–141.
10. Hunter, D., M. Chia, and I. Androvic: 1998, 'Buffering in Optical Packet Switches'. *IEEE Journal of Lightwave Technology* **16**(12), 2081–2094.
11. Konstantopoulos, T. and V. Anantharam: 1995, 'Optimal flow control schemes that regulate the burstiness of traffic'. *IEEE/ACM Transactions on Networking* **3**(4), 423–432.
12. Luby, M., M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman: 2001, 'Efficient erasure correction codes'. *IEEE Transactions on Information Theory* **47**(2), 569–584.
13. Medina, R.: 2002, 'Photons vs. electrons [all optical network]'. *Potentials* **21**, 9–11.
14. Ramaswami, R. and K. Sivarajan: 2002, *Optical Networks: a practical perspective*. San Francisco: Morgan Kaufmann.
15. Tyler, E. J., P. Kourtessis, M. Webster, E. Rochart, T. Quinlan, S. E. M. Dudley, S. D. Walker, R. V. Petty, and I. H. White: 2003, 'Towards Terabit-per-second Capacities over Multimode fiber links using SCM/WDM Techniques'. *Journal of Lightwave Technology* **21**, 3237–3243.