EXACT EMULATION OF A PRIORITY QUEUE WITH A SWITCH AND DELAY LINES

by

Anand D. Sarwate and Venkat Anantharam

Memorandum No. UCB/ERL M04/47

1 April 2004

EXACT EMULATION OF A PRIORITY QUEUE WITH A SWITCH AND DELAY LINES

by

Anand D. Sarwate and Venkat Anantharam

Memorandum No. UCB/ERL M04/47

1 April 2004

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

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

Anand D. Sarwate* and Venkat Anantharam†
Department of Electrical Engineering and Computer Science
University of California, Berkeley

{asarwate, ananth}@eecs.berkeley.edu

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 $M = \Omega(\log K)$ 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.

1 Introduction

Packet switched communication networks need to resolve contention between flows that compete for the same link by buffering some of the contending packets. Optical networks with wavelength division multiplexing (WDM) can already support data rates of several terabytes per second [1]. Unfortunately, there do not exist buffering strategies for optical packets that have the flexibility of their electronic counterparts [2]. 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 [3]. This so-called O-E-O bottleneck is one of the main stumbling blocks to the rapid deployment of all-optical networks.

A packet switch accepts packets coming from several inputs and transfers each of them to one of several destinations [1]. Each packet may be thought of as having a header which prescribes its intended destination. A packet switch must resolve the contention that arises when packets from several inputs request to be directed to the same output. In electronic packet switches such contention is resolved by buffering packets and the use of switch scheduling rules. Buffers may be needed at one or more of several locations, depending on the architecture of the switch : at the

*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.

[1]We do not consider multicast flows in this paper.

input ports, at the output ports, or distributed throughout the switching fabric [4]. In all-optical packet switched networks that do not use O-E-O conversion, this approach would require buffering optical packets. In contrast to the situation in electronics, there currently exists no acceptable method for creating optical random access memory (RAM); research laboratories are still working on overcoming this [5].

There are three major strategies for resolving this issue while avoiding O-E-O conversion. One is deflection routing, where packets that lose in conflict resolution are switched to a different output, passing the responsibility for correct routing to the next node in the network [6]. This idea has also been used earlier in electronic networks [7]; the extreme situation where this is used with no buffering has been called hot potato routing [8]. Another idea is wavelength conversion, which can be used together with WDM to try to pack contending packets onto the same fiber in a nonconflicting matter [9, 10]. The third strategy is to attempt to mimic buffering. It is with an eye to this third approach that we propose the ideas in this paper. We now discuss the background work in this direction in more detail.

There have been many schemes previously proposed for mimicking buffering [11]. Almost all involve using a optical switch and fiber delay lines (FDLs), which may be thought of as having a fixed-length that is an integer multiple of the length occupied by a single optical packet [2]. These can be classified along two axes – feedforward vs. feedback and single-stage vs. multistage. The simplest kind of buffering architecture is a single-stage feedforward architecture. The staggering switch [12], KEOPS [13], and OASIS [14] are all single-stage feedforward schemes that attempt to resolve contention by delaying contending packets by different amounts [3]. Multi-stage feedforward schemes allow greater control over traffic, and can be used to try to mimic deeper buffers. Some proposed schemes are CORD [15], COD [16], Wave-Mux [17], SLOB [18], the logarithm delay-line switch [19], and QUADRO [20]. Recently, Chang et al. [21] (as referenced in [22]) proposed a scheme to mimic a work conserving FIFO buffer using a switch and delay lines in a multi-stage feedforward configuration.

The simplest single-stage feedback architecture is the STARLITE switch [23], in which all contending packets except one are delayed by a single slot and fed back to the input of the switch in the next slot. The shared-memory optical packet switch [24] uses several feedback paths of different delay lengths, and multiple packets can enter a long delay line sequentially. At the present time, no multi-stage feedback architectures appear to have been proposed, to the extent that we could determine. An important feature of feedback switches is that they can allow incoming packets to preempt packets in the buffer, so that it is reasonable to attempt to mimic priority queues rather than just first-in first-out (FIFO) queues. It should be noted that there are still physical-layer issues required to mitigate crosstalk [25] in the delay loops.

In general, buffer designs for packet switches are designed with specific traffic models in mind. Optical *burst* switches are designed for traffic models in which it is desirable to maintain contiguity of sequences, or bursts, of packets arriving from different sources [26]. For example, consider two independent bursts of packets arriving into the switch destined for the same output. Packet contention architectures insensitive to the packet content may produce an output stream that interleaves the two bursts. However, it may be desirable to buffer one burst and output it after the other. In order to resolve contention for bursty traffic, buffers of lengths on the order of thousands are needed [27]. Of the above designs, a few [12,18,20,22] are designed specifically for bursty traffic.

In all the schemes discussed above, the extent to which they are successful in emulating buffering

[2] In this paper we will assume that all packets are of fixed length and that all systems are time synchronous.

[3] The switch designs cited in this literature survey are listed by their acronyms. For the full form of these acronyms, please refer to the corresponding papers.

for arbitrary traffic patterns is rather unclear. Our purpose in this paper is to pursue an approach to the emulation of buffered queueing disciplines that starts from first principles. We are guided in our problem formulation by a broad vision of how an all-optical packet switched network might look. This is described in the next section.

2 A proposed all-optical packed switched network architecture

In this paper we consider *exact* emulation of priority queues with finite buffer size. Our general aim is to motivate the study of the problem of emulation of finite buffer priority queues by structures that can be implemented in all-optical technology. The specific problem studied in the next few sections is how to exactly emulate a priority queue of finite buffer size by a structure comprised of a switch and fixed length delay lines.

In this 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. By a structure that approximately emulates a priority queue we mean a black box with one input port and one output port, accepting packets on the input port and releasing them from the output port, such that the mapping from the arrivals to the departures approximates that required by the queue to be emulated. The kind of approximation we are proposing is discussed below.

Our idea is to view the dropping pattern of the packets in a flow as a parameter of quality of service. In the context of an approximate emulation using switches and fixed length delay lines, the eventual goal would then be to design switching algorithms that can give a rigid guarantee that the dropping pattern is not more severe than a prescribed threshold. This architecture decouples the network level design problem into a set of switch level problems.

Error control coding is generally used at the physical layer in order to recover from errors introduced by a noisy channel [28]. The idea is to introduce redundancy into the transmitted information so the intended message can be correctly recovered with high probability even in the presence of channel errors. A fundamental observation to make in networking is that the network is basically 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. Since packet loss is unavoidable in packet switched networks with limited buffering, and therefore in particular in optical packet switched networks, it makes sense to use error control coding to overcome this kind of error.

The appropriate view of the channel induced errors posed by packet drops is as *erasures* at the packet level. The Shannon capacity of the basic erasure channel (which assumes independent erasures from bit to bit) is well known to be $1 - p$ bits per channel use, where p is the bit erasure probability [29]. If one is dealing with packets instead of bits, the same formula applies under the assumption that packet drops are independent from packet to packet. A simple rule of thumb, therefore, is that in order to transmit information at a data rate of R Gbps one would have to send coded information at the rate of $\frac{R}{1-p}$ Gbps, if the packet drop probability were p.

To be able to use erasure control coding in practice to recover from packet drops, one must answer the question : are there any good codes (good meaning efficient to encode and decode) which enable one to actually communicate at a higher data rate in the above way, and to recover from packet drops? Fortunately, there has been a great deal of recent work in coding theory that has already satisfactorily answered this question, see Alon et al. [30], and the more recent work of

Luby et al. [31] for instance [4]. The codes constructed in [31] have the property that, for any $\epsilon > 0$, they can be encoded in time proportional to $\log(\frac{1}{\epsilon})$ times the total number of packets sent, and the information can be recovered from the received stream of packets as long as at least a fraction $(1+\epsilon) \cdot (1-p)$ of the packets are received, when the drop probability is p per packet. Further, the recovery algorithm is also guaranteed to run in a time that is proportional to $\log(\frac{1}{\epsilon})$ times the total number of packets sent. Indeed, the decoding algorithms for such codes are very attractive to implement, since they are based on message passing algorithms on graphs; a great deal of recent work has developed techniques to efficiently implement such decoding algorithms in hardware [34].

The preceding discussion highlights the importance of designing scheduling algorithms at switches that are able to offer stringent drop probability guarantees to the flows they handle. Such switches would need buffers at the input or output ports, or buffers distributed throughout the switch fabric, depending on the switch architecture. Since packet drops are inevitable, the focus should be on approximate emulation of target input output behavior at these buffers, which allows for the possibility of controlled packet drops.

Our proposal is that the desired input-output behavior at a buffer be emulated by means of a switching system with fixed length delay lines. Thus the overall architecture of a switch would be comprised of the main switch, and of smaller switching systems with fixed length delay lines, each of which approximately emulates a priority queue of fixed buffer size, from the point of view of the main switch. The pattern of demands on the smaller switching systems is determined by the switching algorithm of the main switch, and arrivals are sent to the smaller switching systems, as if they were going to buffers.

The guarantee we need to be able to offer at one of the smaller switching systems is that the fraction of times that a packet is *not* released in response to a demand request is less than some prescribed threshold. Some thought reveals that the last sentence does not quite make sense, because the notion of packet drop probability is an asymptotic one. We therefore propose the following characterization of approximate emulation : it is defined by s, an integer, and p, a number in $(0,1)$, such that over any stretch of n successive demand requests, the total number of such requests that are not successfully met is at most $s+pn$. This characterization is analogous to that in the familiar theory of burstiness constrained flows [35–38], except that it is being applied to packet drops. We might then try to study the problem of most efficient emulation of a priority buffer by a black box which achieves this goal in its input-output behavior. The correct metric for evaluating the cost of such an emulation would be technology dependent; the goal would be to find the most efficient approximate emulation for any given parameter pair (s, ρ), desired priority queueing discipline, and desired buffer size at the queue.

More generally, we may also allow for the possibility that some demand requests are not satisfied at the time they are made, but are satisfied with some delay. Since we would not want to deal with resequencing problems at the receiving end in a practical system, we might insist that demand requests, even if served with a delay, are served in order; this would mean that once a packet is released as serving a demand request, all earlier packets which have not yet been released from the black box are effectively considered lost. We might impose a requirement that the maximum delay suffered by any packet that serves a demand request, in a delayed way, is at most some predefined integer Δ. We might then pose the problem of characterizing the most efficient structure for a

[4] Some of this work was motivated by another set of ideas to use error control coding in networking, see Byers et al. [32]. This is the idea that for multicast applications, it may make sense to introduce packets into the network in a push mode rather than an acknowledgment based mode, in order to avoid the problem of ack overload. This idea was based on the earlier idea of priority encoded transmission of Albanese et al. [33], and is currently still being commercially pursued by the company Digital Fountain.

smaller switching system that meets the quality of service criteria (s, ρ, Δ) in emulating a desired priority queuing discipline for a given desired buffer size. This formulation is much more complicated to deal with from the point of view of designing scheduling algorithms at the main switch. It may therefore not be of interest in terms of the overall architecture we propose, but appears to be of interest in its own right.

Apart from the potential interest in the context of an architecture for all-optical networking, such as that suggested above, the problems of emulating queuing disciplines using switches and delay lines may also be of interest for various integrated optics applications, e.g. to realize on-chip buffers in optics. It could also be useful in developing all-optical local area networks.

In closing this section, we emphasize once again that in this paper we only consider *exact* emulation of a priority queue of a fixed finite size. The ideas sketched in this section should be taken as meant to illustrate our thinking regarding promising directions for future work.

3 Problem formulation

We now define what we mean by a *priority queue* of size K. At any given time there are $0 \leq k \leq K$ packets in the queue. These packets, if any, are assigned *ranks* 1 through k, with 1 being the highest priority. The relative priorities of packets in the queue never changes from one time to another. At each time there may or may not be an arrival request and there may or may not be a departure request. The arrival request comes with a priority relative to the existing packets in the queue. The queue operates as follows :

- *(no arr. req./no dep. req.)* If there is no arrival request and no departure request, nothing changes : the queue continues to have as many packets as before, and these have the same ranks as before.

- *(no arr. req./dep. req.)* If there is no arrival request and there is a departure request, the highest priority (rank 1) packet, if any, is released from the queue to serve the departure request, and the ranks of each of the remaining packets, if any, is decremented by 1. Thus, if there were k packets in the queue, there are now $(k-1)^+$ packets in the queue, and these have the same relative priority as before.

- *(arr. req./no dep. req./queue not full)* 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 as the packet with rank $j+1$, the packets that originally had ranks 1 through j continue to have the same ranks, and the packets that originally had ranks $j+1$ through k now have ranks $j+2$ through $k+1$ respectively.

- *(arr. req./no dep. req./queue full)* 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 what its priority is relative to the existing packets in the queue[5].

- *(arr. req./dep. req./highest priority arr.)* 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 originally in the queue continue to remain in the queue with the same ranks as before.

[5]It is possible to modify our results to deal with a dynamic such that the arriving packet displaces the lowest priority packet. See Section 6 for this extension.

- *(arr. req./dep. req./lower priority arr.)* 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 that had rank 1, the packets that originally had ranks 2 through j now have ranks 1 through $j-1$, the arriving packet is admitted into the queue with rank j, and the packets that had ranks $j+1$ and bigger continue to have the same ranks as before.

A First-In-First-Out (FIFO) queue of size K is an example of a priority queue of size K, as defined above : here an arrival request always has lower priority than all the packets currently in the queue. A Last-In-First-Out (LIFO) queue of size K is also an example of a priority queue of size K, as defined above : here an arrival request always has higher priority than all the packets currently in the queue. In the LIFO case, as defined above, it is important to note that a packet that is once admitted into the queue cannot leave except to serve a departure request : for an arrival request into a full queue when there is no departure request, it is the arriving packet that is rejected [6]. In a general priority queue of size K, as defined above, one can think of the priority of an arrival request as being assigned by the sender. Packets that are already admitted to the queue maintain their relative order, although new packets may be inserted between them. However, a new packet, whatever its apparent priority relative to the existing packets in the queue, can be accepted only if there is room for it, which happens either if it can directly serve the departure request, if any, or if room is created for it by one of the existing packets serving the departure request, if any, or if the queue is not full when this packet arrives.

In this paper we study the problem of *exact* emulation of a priority queue of size K by a bufferless optical packet switch with scheduled fiber delay lines. A priority queue realized in this way can be used as an input or output buffer within an optical packet switch so that the switch has exactly the same functionality as that of a corresponding electronic switch, without the need for O-E-O conversion. As indicated in Section 2, the problem of *approximate* emulation of a finite buffers of given input-ouput characteristics is also of interest in the context of our proposed architecture for all-optical packet switched networks. This problem is not addressed here; it would be an interesting topic for future research. The mathematical question we want to address is how to implement a priority queue of size K, with delay lines and a switch of size $(M+1) \times (M+1)$. For a target queue depth K, we will measure the complexity of the implementation by $M+1$.

The setup is shown in Figure 1. Packets arrive along a distinguished input to the switch, which we call the arrival input, and which is indexed by 0. Packets exit along a distinguished output of the switch, which we call the departure output, and which is indexed by 0. 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, 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. If so, we say there is an arrival request and this packet is called the arriving packet. 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 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. A packet that is switched into the departure output leaves the switching system.

[6] See footnote 5.

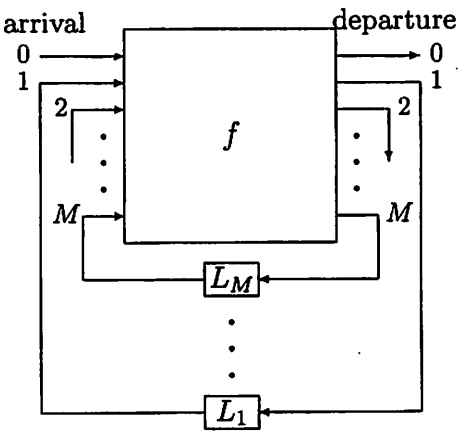

Figure 1: A queue implemented as a switch with delay lines.

The *state* of the switching system at time t is the position of all packets in the delay lines, with their relative priorities, knowledge of whether there is or is not a departure request at time t, and knowledge of whether there is or is not an arrival request at time t, together with the number of existing packets at time t that have lower priority than that arriving packet (this information is carried by the arriving packet). This is a complete summary of the information that a switching algorithm has on which to base its choice of switching pattern at time t.

An *algorithm* is defined by a family of permutations from $\{0,\ldots,M\}$ to $\{0,\ldots,M\}$, indexed by states : such a permutation determines which inputs to connect to which outputs. When the algorithm is applied at time t, packets in the delay lines that are not at the inputs to the switch move one slot forward within the same delay line, and those that are at the inputs to the switch, if any, together with the arriving packet, if any, are switched into the departure output and the beginnings of the delay lines. The switching system is said to emulate a priority queue of size K if the following hold true at every time t :

- *(no arr. req./no dep. req.)* If there is no arrival request and no departure request at time t the algorithm does not place any packet on the departure output, is able to recirculate all the packets that came out from the outputs of the M delay lines back into the inputs of the M delay lines, and the relative priorities of the packets in the switching system do not change.

- *(no arr. req./dep. req.)* If there is no arrival request and there is a departure request, the algorithm places the highest priority packet that was in the switching system at time t on the departure output (this implies, in particular, that this packet must have been available at one of the outputs of the M delay lines), is able to recirculate all the other packets that came out from the outputs of the M delay lines back into the inputs of the M delay lines, and the relative priorities of the packets remaining in the switching system do not change.

- *(arr. req./no dep. req./sw. sys. has less than K packets)* If there is an arrival request which has lower priority than j of the packets currently in the switching system and there is no departure request, and if the total number of packets currently in the switching system is $k < K$, the algorithm places no packet on the departure output, is able to admit the arriving packet as well as the packets coming out at the outputs of the M delay lines into the inputs of the M delay lines, and gives the admitted packet a priority greater than exactly j of the

existing packets, while the existing packets continue to have the same relative priorities as before.

- *(arr. req./no dep. req./K packets in the sw. sys.)* If there is an arrival request and there is no departure request and the total number of packets in the switching system is K, the algorithm places the arriving packet on the departure ouput and is able to place the packets coming from the outputs of the M delay lines back to the inputs of these delay lines, with the relative priorities of these packets being unchanged. The placement of the arriving packet on the departure output in this case 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.

- *(arr. req./dep. req./highest priority arr.)* If there is an arrival request which has higher priority than all the packets currently in the switching system and there is a departure request, the algorithm places the arriving packet on the departure output, is able to place the packets coming out of the M delay lines back into these delay lines, and the relative priorities of these packets is unchanged.

- *(arr. req./dep. req./lower priority arr.)* If there is an arrival request which has lower priority than j of the packets currently in the switching system, $j > 0$, and there is a departure request, the algorithm places the highest priority packet in the switching system on the departure output (again, this implies that this packet must have been available at one of the outputs of the M delay lines), is able to recirculate all the other packets that came out from the outputs of the M delay lines, as well as the arriving packet back into the inputs of the M delay lines, the relative priorities of the packets remaining in the switching system do not change, and the arriving packet is given a priority greater than all but exactly $j-1$ of the existing packets.

Starting 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, and for a given pattern of departure requests, and of arrival requests with associated priorities, one can see by following through the various cases that a switching algorithm that meets the requirements listed above does emulate a priority queue of size K, as described earlier, in that the number of packets in the switching system at any time will exactly equal those in the priority queue, the relative priorities of the packets in the switching system will be exactly that same as in the priority queue, an arrival request into a switching system that currently has fewer than K packets will always be honored, as in the priority queue, an arrival request into a switching system that currently has K customers will be honored if there is a simultaneous departure request, as in the priority queue, and a departure request to the switching system will always be served by the highest priority packet, if any, as in the priority queue.

Since 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 the size of the priority queue being emulated. In this paper, we will show that $M = \Omega(\log K)$ and provide a construction with $M = O(\sqrt{K})$.

We close this section by defining the notion of *post-rank*, which will be used in the proofs. At any time t, each packet that is in the priority queue at time t, as well as the arriving packet, if any, is assigned a post-rank. The post-rank at time t of a packet that is present in the priority queue at time $t+1$ equals its rank at time $t+1$. A packet that is released from the priority queue at time t to serve a departure request is assigned post-rank 0 at time t. An arriving packet that is rejected from a full queue is also assigned a post-rank of 0. Note that the range of values permitted for the post-rank is $\{0, 1 \ldots, K\}$. Also note that at every time t the packets in the queue at that time and

the arriving packet, if any, all have distinct post-ranks, which form a contiguous set of nonnegative integers starting with the lowest possible.

4 A lower bound of $\Omega(\log K)$

In order to establish a lower bound on the switch size, we must find necessary conditions on the lengths $\{L_i\}$ of the delay lines. In this section we will assume that the lengths of the delay lines are ordered so that $L_1 \leq L_2 \leq \cdots \leq L_M$ [7]. Further, since we assume that the switching system is exactly emulating a priority queue of size K, we can talk about the rank and the post-rank of packets at any given time; these are respectively the rank and the post-rank in the priority queue that is being emulated.

There must be at least enough room in the system for K packets, so we must have

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

A lower bound can be derived from this inequality and the observation that a packet with post-rank j at time t cannot be switched at time t into a delay line with length more than j. Suppose that at time t, the packet with post-rank j is switched into a delay line of length more that j. Suppose that at times $t+1, t+2, \ldots t+j$, there are departure requests and no arrivals. Then at time $t+j$, this packet will have post-rank 0, but will not be available to be switched to the output line of the switching system, so the switching system cannot correctly emulate the priority queue. We can use this observation to prove 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. That $L_1 = 1$ follows from the observation that, 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, which must have post-rank 1, must, by our observation, 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 there exists a switching system which exactly emulates a priority queue of size K and such 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 the observation that a packet cannot be assigned to a line with length more than its post-rank at the time it is switched into that line, at time $t+j$ the switching system has j packets which are all in

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

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 post-rank $j+1$. Since the lines $1 \ldots m$ 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 switched into a line numbered $m+1$ or higher. The switching system is therefore forced to assign a packet with post-rank $j+1$ or less to a line with length $j+1$ or higher, which, by our observation, implies that the switching system cannot exactly emulate a priority queue of size K. □

Using Lemma 1 we can prove the following lower bound on the size of any switch that can emulate a priority queue in a system comprised of the switch and fixed length delay lines :

Proposition 1. $M = \Omega(\log K)$.

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 $M = \Omega(\log K)$. □

5 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 $M = O(\sqrt{K})$. For simplicity we will consider only the case where K is of the form $\frac{1}{2}N(N+1)$ for some $N \geq 1$. For such K we will show that a priority queue of size K can be emulated by a switching system with an $(M+1) \times (M+1)$ switch with $M = 2N-1$. Since a switching system emulating a priority queue of a given size can also emulate one of any smaller size, this will prove that a priority queue of any size K can be emulated by a switching system with a switch of size $O(\sqrt{K})$. 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 [8], a quite interesting size in practice. 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

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

That is, $\alpha(k)$ is a root of the equation $k = \frac{1}{2}n(n+1)$. Thus, 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* R and the set of *overflow lines* F.

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

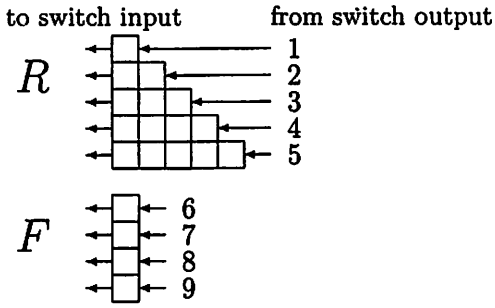

Figure 2: A queue of size K = 15.

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 straw-man 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 the original switching system also exactly emulates a priority queue of size K, which is what we set out to prove to in this section.

Consider a time t. The SSS is in some state at this time. Recall that the state is comprised of the locations of the packets in the delay lines at time t, together with their relative priorities, knowledge of whether there is a departure request or not at time t, knowledge of whether there is an arrival request or not at time t, and, if there is an arrival request, the number of existing packets that are of lower priority than the arrival request. Assume that the SSS has been working correctly to emulate the priority queue of size K up to the current time t. Thus the packet with highest priority is available at the input of the switch to service a departure request, if any. We will now describe a class of switching patterns, as a function of the state at time t, each of which will ensure that the the emulation of the priority queue continues to be correct up to time $t+1$, and that the packet with the highest priority at time $t+1$ is in the shortest regular line at time $t+1$. This then describes a class of algorithms for the SSS each of which ensures that it exactly emulates a priority queue of size K.

We are interested in algorithms 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*. Since the algorithms is assume to be correctly emulating the priority queue up to time t, we can talk about the rank and post-rank of each packet at time t and about the post-rank of the arriving packet, if

any, at time t. At time t, the algorithm places the packet with post-rank 0, if any, in the departure line, and places the remaining packets that need to be switched, if any, in order of increasing post-rank, in the shortest regular lines, 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 the highest priority at time $t+1$ will be in the shortest regular line at time $t+1$. This follows from the observation that the packet with post rank 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 post-rank.

We will now describe a particular way of computing $T(t)$. For this way of computing $T(t)$, we will show that we need to have at most $N-1$ overflow lines. As indicated earlier, this will tell us that we do not really need to work with the straw-man switching system, but can work with the original switching system. This will complete the proof of our upper bound.

We first introduce some notation that is used for our proof. Recall that we are still working with the SSS. 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 system 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.

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

Let $\omega_Q^+(t)$ denote $\omega_Q(t+1)$. While the state of the SSS at time $t+1$ can be computed only after the application of the switching algorithm, $\omega_Q^+(t)$ can be computed based on the state at time t, so it is appropriate to think of it as indexed by t. One then has the formula

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

which is just a restatement of (5)

Let $\omega_F(t)$ denote the number of packets in the overflow lines at time t. Let $S(t)$ denote the largest length 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 [9]. Then we have

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

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

[9] Note that this does not include the packet placed in the departure output of the switch in the SSS, if any.

order of post-rank, 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

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

where step (a) 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 length $T(t-1)$. 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 shortest regular lines up to and including the line of length $T(t) - 1$, if any [10]. 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.

□

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

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

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 length $T(t)$ 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 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 $T(t+2) = T(t)$. 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 $t+3$, so 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 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 length $T(t)$ 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 straw-man switching system 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 straw-man switching system : the original switching system would have done as well. This completes our proof that $M = O(\sqrt{K})$.

□

6 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. According to the definition of a priority queue that we have used so far, an arriving packet is rejected if the queue is full and there is no departure request, irrespective of what its priority is relative to the packets currently in the queue. However, if the queue is full and there is no departure request and the arriving packet does not have lower priority than all the other packets in the queue, we might like to admit the arriving packet and instead remove from the queue the existing packet with lowest priority. In this section we work with this alternate definition of priority queue.

At each time t we will talk about the *rank* of a packet in the queue, and we will talk about the *post-rank* for packets that were in the queue and for the arriving packet, if any. As before, the rank of a packet at time $t+1$ will equal its post-rank at time t. The post-rank of a packet that is in the at time t, as well as the post-rank at time t of the packet that arrives at time t, if any, will be computed exactly as in the preceding section in all cases, except in case 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 $0 \leq j \leq K-1$. In this case, the packet that had rank K will be given post-rank 0, the arriving packet will be given post-rank $j+1$, the packets currently in the queue with ranks at most j will have post-rank equal to their rank, and each packet currently in the queue with a rank in the range $j+1$ through $K-1$ inclusive will have post-rank equal to one plus its rank.

We will show how to emulate a priority queue of size $K = \frac{N(N+1)}{2}$, in the new sense, with 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 straw-man 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 post-rank 0, if any, in the departure line, places any packets with post-ranks in the set A into the lines in E, places $T(t)$ packets in increasing order of post-rank into the shortest regular lines, and places the remaining packets in the overflow lines. In this new operation, the end lines E are reserved for the packets with ranks in A.

Since a packet is always placed in a line of length no bigger than its post-rank, 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 rank 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 rank K is always available at the input of the switch.*

Proof. Suppose that the queue is full at time t and the packet with rank 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 post-rank at time s, i.e. its rank at time $s+1$, must have been at most $K-N+1$. Since the rank of a packet can increase by at most 1 at each time, its current rank is at most $(K-N+1)+(t-(s+1)) \leq K-1$, a contradiction. Thus the packet with rank 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 trapping line, which we give 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 $\frac{\alpha(K)(\alpha(K)+1)}{2} + 2(\alpha(K) - 1) = K + O(\sqrt{K})$.

7 Conclusion

Before all-optical switching networks can become feasible, we need ways in which to mitigate packet contention and congestion. One way in which these issues are managed in electronic packet networks is via buffering. Therefore it is desirable to find a method of emulating buffering given the constraint that optical RAM does not currently exist. Here we have proposed an architecture for emulating priority queues of finite size in which packets are switched through fixed-length delay lines. We proved that to exactly emulate a priority queue of size K we need a switch of size at least $\Omega(\log K)$ and demonstrated a specific architecture that can emulate the queue with a switch of size $O(\sqrt{K})$. The total length of the delay lines used in our implementation is $K + O(\sqrt{K})$. We gave implementations for two slightly different notions of priority queue, both of which might be of interest.

One advantage of this architecture over feedforward schemes is that packets can wait in the queue for a departure request. This provides more flexibility for congestion control schemes in

which the switch can control the rate at which packets leave on its outgoing lines. In addition, feedback architectures allow the implementation of priority scheduling, which is important for more advanced traffic control [39]. It would be interesting to find architectures which can emulate even more complicated queuing policies.

Clearly there is a large gap between the upper and lower bounds in this paper. The proof of our lower bound is quite naive, and it may be possible to improve this lower bound by more careful analysis. Furthermore, 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.

Acknowledgment

A Proofs

We prove the lemmas from Section 6.

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 5 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] E. J. Tyler, P. Kourtessis, M. Webster, E. Rochart, T. Quinlan, S. E. M. Dudley, S. D. Walker, R. V. Petty, and I. H. White, "Towards terabit-per-second capacities over multimode fiber links using SCM/WDM techniques," *Journal of Lightwave Technology*, vol. 21, pp. 3237–3243, 2003.

[2] R. Ramaswami and K. Sivarajan, *Optical Networks: a practical perspective.* San Francisco: Morgan Kaufmann, 2002.

[3] R. Medina, "Photons vs. electrons [all optical network]," *Potentials*, vol. 21, pp. 9–11, 2002.

[4] G. Nong and M. Hamdi, "On the provision of quality-of-service guarantees for input-queued switches," *IEEE Communications Magazine*, vol. 38, no. 12, pp. 62–69, 2000.

[5] R. Takahashi, T. Nakahara, K. Takahata, H. Takenouchi, T. Yasui, N. Kondo, and H. Suzuki, "Photonic random access memory for 40 Gb/s 16-b burst optical packets," *IEEE Photonics Technology Letters*, vol. 16, no. 4, pp. 1185–1187, 2004.

[6] T. Chich, J. Cohen, and P. Fraigniaud, "Unslotted deflection routing : A practical and efficient protocol for multihop optical networks," *IEEE/ACM Transactions on Networking*, vol. 9, no. 1, pp. 47–59, 2001.

[7] C.-L. Wu and T. Y. Feng, *Tutorial : Interconnection Networks for Parallel and Distributed Processing.* Washington DC: IEEE Computer Society Press, 1984.

[8] P. Baran, "On distributed communication networks," *IEEE Transactions on Communication Systems*, vol. 12, pp. 1–9. 1964.

[9] A. Bononi, G. A. Castañón, and O. K. Tonguz, "Analysis of hot-potato optical networks with wavelength conversion," *Journal of Lightwave Technology*, vol. 17, no. 4, pp. 525–534, 1999.

[10] S. Yao, B. Mukherjee, and S. Dixit, "Advances in photonic packet switching: An overview," *IEEE Communications Magazine*, pp. 84–94, February 2000.

[11] D. Hunter, M. Chia, and I. Androvic, "Buffering in optical packet switches," *IEEE Journal of Lightwave Technology*, vol. 16, no. 12, pp. 2081–2094, 1998.

[12] Z. Haas, "The 'staggering switch': An electronically controlled optical packet switch," *IEEE Journal of Lightwave Technology*, vol. 11, no. 5/6, pp. 925–936, 1993.

[13] C. Guillemot, M. Renaud, P. Gambini, C. Janz, I. Andonovic, R. Bauknecht, B. Bostica, M. Burzio, F. Callegati, M. Casoni, D. Chiaroni, F. Clérot, S. Danielsen, F. Dorgeuille, A. Dupas, A. Franzen, P. Hansen, D. Hunter, A. Kloch, R. Krähenbühl, B. Lavigne, A. LeCorre, C. Raffaelli, M. Schilling, J.-C. Simon, and L. Zucchelli, "Optical packet switching: The European ACTS KEOPS project approach," *IEEE Journal of Lightwave Technology*, vol. 16, no. 12, pp. 2117–2134, 1998.

[14] J. M. Gabriagues, F. Maretti, and J. B. Jacob, "Optical atm switching architectures," *IEEE Colloquium on Optical and ATM*, p. 5/1, 1995.

[15] I. Chlamtac, A. Fumagalli, L. Kazovsky, P. Melman, W. Nelson, P. Poggiolini, M. Cerisola, A. Choudhury, T. Fong, R. Hofmeister, C. Lu, A. Mekkittikul, D. Sabido, C. Suh, and E. Wong, "CORD: contention resolution by delay lines," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1014–1029, 1996.

[16] R. Cruz and J.-T. Tsai, "COD: Alternative architectures for high-speed packet switching," *IEEE/ACM Transactions on Networking*, vol. 4, February 1996.

[17] Y. Nakahira, H. Inoue, and Y. Shiraishi, "Evaluation of photonic ATM switch architecture - proposal of a new switch architecture," *Proceedings of the 15th International Switching Symposium (ISS '95)*, vol. 2, pp. 128–132, 1995.

[18] D. K. Hunter, W. D. Cornwell, T. H. Gilfedder, A. Franzen, and I. Andonovic, "SLOB: A switch with large optical buffers for packet switching," *IEEE Journal of Lightwave Technology*, vol. 16, no. 10, pp. 1725–1736, 1998.

[19] D. Hunter, D. Cotter, R. Ahmad, D. Cornwell, T. Gilfedder, P. Legg, and I. Andonovic, "2x2 buffered switch fabrics for traffic routing, merging and shaping in photonic cell networks," *IEEE Journal of Lightwave Technology*, vol. 15, pp. 86–101, 1997.

[20] I. Chlamtac and A. Fumagalli, "QUADRO-star: High performance optical WDM star networks," *IEEE Transactions on Communications*, vol. 42, pp. 2582–2590, August 1994.

[21] C. Chang, D. Lee, and C. Tu, "Recursive construction of FIFO optical multiplexers with switched delay lines." submitted to IEEE Transactions of Information Theory, 2002.

[22] C.-S. Chang, D.-S. Lee, and C.-K. Tu, "Using switched delay lines for exact emulation of FIFO multiplexers with variable length bursts," in *Proceedings of IEEE INFOCOM*, 2003.

[23] A. Huang, "STARLITE: A wideband digital switch," in *Proceedings of GLOBECOM 1984*, pp. 121–125, 1984. paper 5.3.

[24] M. J. Karol, "Shared-memory optical packet (ATM) switch," in *SPIE Vol. 2024: Multigigabit Fiber Communication Systems*, 1993.

[25] W. Pieper, M. Eiselt, G. Grosskopf, R. Langenhorst, A. Ehrhardt, and H. G. Weber, "Investigation of crosstalk interference in a fibre loop optical buffer," *Electronics Letters*, vol. 30, pp. 435–436, 1994.

[26] G. Papadimitriou, C. Papazoglou, and A. Pomportsis, "Optical switching: Switch fabrics, techniques, and architectures," *IEEE Journal of Lightwave Technology*, vol. 21, no. 2, pp. 384–405, 2003.

[27] J. Dupraz, "ATM: Current status and perspectives of evolution," in *Proceedings of ECOC 1994*, (Firenze, Italy), pp. 555–562, September 1994.

[28] R. E. Blahut, *Algebraic Codes for Data Transmission.* Cambridge, UK: Cambridge University Press, 2003.

[29] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* New York: John Wiley & Sons, 1991.

[30] N. Alon and M. Luby, "A linear time erasure-resilient code with nearly optimal recovery," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1732–1736, 1996.

[31] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correction codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.

[32] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1528–1540, 2002.

[33] A. Albanese, J. Blorner, J. Edmonds, M. Luby, and M. Sudan, "Priority encoded transmission," in *Proceedings of the 35th Annual ACM Symposium on the Foundations of Computer Science*, 1994.

[34] E. Yeo, B. Nikolic, and V. Anantharam, "Iterative decoder architectures," *IEEE Communications Magazine*, vol. 41, no. 8, pp. 132–140, 2003.

[35] R. L. Cruz, "A calculus for network delay I : network elements in isolation," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 114–131, 1991.

[36] R. L. Cruz, "A calculus for network delay II : network analysis," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 132–141, 1991.

[37] T. Konstantopoulos and V. Anantharam, "Optimal flow control schemes that regulate the burstiness of traffic," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 423–432, 1995.

[38] V. Anantharam and T. Konstantopoulos, "A methodology for the design of optimal traffic shapers in ATM networks," *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 583–586, 1999.

[39] M. Ribeiro and M. O'Mahony, "Traffic management in photonic packet switching nodes by priority assignment and selective discarding," *Computer Communications*, vol. 24, pp. 1689–1701, 2001.