Multiple Priority, Per Flow, Dual GCRA Rate Controller for ATM Switches

Avi Hagai Boaz Patt-Shamir
avih@wintegra.co.il boaz@eng.tau.ac.il
Dept. of Electrical Engineering
Tel-Aviv University
Tel Aviv 64583
Israel

Abstract—**We propose a rate controller for ATM switches. The rate controller supports multiple priorities, and dual leaky bucket (GCRA) traffic descriptors (such as VBR). While regulating each stream independently, our rate controller requires relatively modest computation bandwidth so that it can be implemented without any additional special-purpose hardware. The memory space requirement under reasonable circumstances is close to the most space-efficient schemes. It also enjoys the important advantage of being decoupled from the link scheduler. We analyze the outgoing traffic characteristics of our shaper with combination of strict priority and WFQ link scheduler, and find the optimal shaping parameters so as to maintain conformance at downstream switches. We study the best ways to allocate resources to rate controllers along the path of connection, and demonstrate the effectiveness of aggressive and light shaping in a multiple stage network under various network loads.**

I. Introduction

Asynchronous Transfer Mode (ATM) is considered a leading technology for transporting multimedia (including voice, video and data) over wide area networks. One of the most challenging problems to solve prior to the deployment of ATM networks is traffic management and congestion control. Congestion control is difficult in ATM networks because of the high bandwidth, diverse quality of service requirements, and various traffic characteristics.

One effective congestion control technique is to delay early cells until they conform to the connection's traffic descriptors, which typically prescribe burst and bandwidth restrictions. This technique is loosely called *traffic shaping* or *rate control*. Shaping accommodates variation in traffic flow by smoothing the incoming cell stream. Without such action the network must reserve more resources to accommodate bursts or sessions may suffer loss, or settle for a lower rate connection. In addition, a shaper can smooth traffic traveling between networks, which may operate with different switches, possibly at different link speeds. Although traffic shaping can reduce cell loss, the smoothing function introduces additional delay and implementation complexity, since the shaper must buffer nonconforming cells and schedule them for later transmission. A single network-access point often services thousands of connections, with different traffic parameters, requiring efficient shaper and scalable techniques for buffering and scheduling cells. Much work has been done on traffic shaping and rate control: see, e.g., the survey of Zhang [10] and references therein.

The architecture of typical rate controller is depicted in Fig. 1.

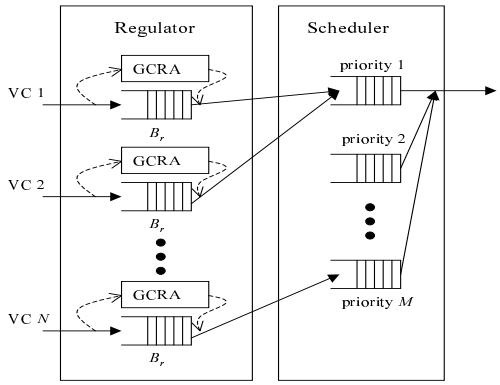

Fig. 1. Architecture of the rate controller. Incoming cells are placed in a per-connection buffer until the (possibly dual) GCRA traffic regulator determines that they are eligible to transmission. At each time step, all cells eligible for transmission are moved to the link scheduler, where they are put in per-priority-class buffers.

The rate controller consists of two modules: *traffic regulator* and *priority scheduler*. Each connection has a dedicated buffer at the regulator, in which incoming cells are stored. The regulator holds each cell until it is *eligible* for transmission according to its traffic model (in our case: GCRA-based model, see below). When a cell becomes eligible, it is put in a scheduler buffer: the scheduler maintains a buffer for each *priority class*. The scheduler's task is to select the next cell to transmit on the outgoing link according to the specific priority policy.

With high speed links and small cell size, a modern ATM switch requires a highly efficient traffic shaping algorithm to process cell arrival and departure every few microsecond, and, in addition, the algorithm should scale to a large number of connections with diverse traffic parameters and quality of service requirements.

In this work, we propose an algorithm for rate control in ATM switches. The important properties of the algorithm are the following

- Traffic shaping based on ATM's version of leaky bucket, called GCRA. This includes the prominent service types of Constant Bit Rate (CBR), and Variable Bit Rate (VBR), as well as Guaranteed Frame Rate (GFR).
- Support for multiple priorities.

• Decoupling rate regulation from link scheduling. This allows one, for example, to replace the priority policy or mechanism without changing the regulator part. It also allows for a simple analysis of the maximum cell delay, and the buffer space requirement.

• The computation overhead of the regulator is low enough to be implemented in software. In particular, the computation depends only on the actual number of the cells transmitted, *regardless of the number of connections*. This property makes the algorithm highly scalable.

The decoupling of rate control from link scheduling forces us to re-calculate the burstiness parameters of the outgoing stream. The problem is as follows. After being shaped, the rate controller passes the cells to the link scheduler. The link scheduling task is to resolve conflicts between competing connections. Conflicts occur when multiple cells, from different connections, become eligible for transmission during the same time slot. Depending on how the scheduler arbitrates amongst competing cells, the burstiness of the connection may increase, thus violating the expectations of downstream nodes in the network, possibly increasing cell loss rates and end-to-end delay. In this work, we analyze the influence of two basic link scheduling disciplines (priority and weighted fair queuing) and derive exact expressions for the shaping required by the rate controller so as to guarantee that the stream *leaving the switch* conforms to a given description.

We conclude with simulation results. We evaluate the performance of the GCRA shaping in a simple linear network, where in each switch, several connections are multiplexed into the output link with various input and shaping parameters. We study the best ways to allocate resources to rate controllers along the path of connection, and demonstrate the effectiveness of aggressive and light shaping in a multiple stage network under various network loads in terms of network buffer space, cell jitter and end-to-end delay.

The remainder of this paper is organized as follows. In Section II we describe the basic traffic models. In Section III we describe the rate control algorithm we propose. In Section IV we analyze the combination of our rate controller with a simple priority link scheduler. In Section V we analyze the combination of our rate controller with a WFQ link scheduler. In Section VI we present the experimental results. In Section VII we compare our rate controller with previous algorithms.

Many missing details can be found in [5].

II. The Traffic Model

To facilitate any meaningful rate control, one must have a *traffic model*. A traffic model is required, on the one hand, to impose some limits on the incoming traffic, and on the other hand, to allow for a useful commitment on the outgoing traffic. Several traffic models have been considered in the literature. For example, the $(X_{\min}, X_{\text{ave}}, I, S_{\max})$ model [3], the (σ, ρ) model [2], the (r, T) model [4]. The basic idea in most of these abstractions is to model *burstiness*, and the differences between them are usually subtle. In this paper we consider mainly the (σ, ρ) model and the ATM services based on it, in particular Constant Bit Rate (CBR) and Variable Bit Rate (VBR) [9].

The basic concept underlying the definitions of CBR and VBR is the ATM variant of leaky bucket, called *Generic Rate Control Algorithm*, abbreviated GCRA. A GCRA is defined by a per-flow procedure which determines, for each arriving cell, whether it is *conforming* or not to the GCRA model. The GCRA model has two parameters: an *increment* denoted I, and a *limit* denoted L. Intuitively, conforming cells should arrive with I time units spacing, but they are allowed to arrive at most L time units before their "expected" arrival time. The "expected" arrival time is maintained by the procedure in a variable `TAT`, whose value is I time units after the previous `TAT` value or the previous cell arrival time—whichever is larger. Pseudo-code for the procedure is given in Figure 2.

```
GCRA specification for increment I and limit L:
    When a cell arrives at time t:
    if first cell then TAT ← t                          // initialize TAT
    if t ≥ TAT then                                     // late cell
        TAT ← t
    else                                                // early cell
        if t < TAT − L then                             // too early
            return NON CONFORMING
    TAT ← TAT + I
    return CONFORMING
```

Fig. 2. Procedure defining whether an arriving cell is conforming to a GCRA (I, L) specification. Note that `TAT` keeps its value through invocations of the procedure.

Note that if $L \geq I$, then it is possible that a few cells arriving back-to-back are conforming. More precisely, the maximal number of cells arriving back-to-back is

$$N = \left\lfloor \frac{L}{I - t} \right\rfloor + 1 \,, \tag{1}$$

where t is the transmission time of a single cell. We say that a stream S *conforms* to the GCRA model with parameters (I, L), denoted $S \sim \text{GCRA}(I, L)$, if all cells in the stream are conforming according to the GCRA algorithm of Figure 2.

The GCRA model is closely related to the (σ, ρ) model. Specifically, let k be the number of bits in a cell. Then, using the definitions of [2], for any stream $S \sim \text{GCRA}(I, L)$, we have that $S \sim (\sigma, \rho)$ under the following transformation.

$$\rho = \frac{k}{I} \,, \quad \text{and} \quad \sigma = k \cdot \left(\left\lfloor \frac{L}{I - t} \right\rfloor + 1 \right) \,. \tag{2}$$

III. The Regulator

The architecture of our GCRA rate controller is depicted in Fig. 1. The rate controller consists of two modules: *traffic regulator* and *priority scheduler*. Each connection has a dedicated buffer at the regulator, in which incoming cells are stored. The regulator holds each cell until it is eligible for transmission, where a cell is said to be eligible if it is conforming according to its GCRA-based traffic model. When a cell becomes eligible, it is put in a scheduler buffer: the scheduler maintains a

buffer for each *priority class*. The scheduler's task is to select the next cell to transmit on the outgoing link according to the priority policy. Usually, the scheduler enforces the *strict priority* policy, wherein at any given time, the cells of the highest non-empty priority class buffer are served. Within each priority class, the scheduler may use another policy such as Weighted Fair Queuing (WFQ) [7], which ensures that each connection gets its allocated bandwidth share.

```
next_conf(t, I, L) :                    // current cell is transmitted at time t
    if first cell then                                  // initialize TAT
        TAT ← t
        return TAT
    else
        TAT ← I + max(t, TAT)
        return max(t, TAT − L)
```

Fig. 3. Procedure used to compute the next time a cell will be eligible to transmission according to a single $\mathrm{GCRA}(I, L)$ specification. The value of TAT is retained through different invocations.

```
Constants
SIZE: size of calendar queue

Data Structures
cal:  array of length SIZE of lists of connections
now   current entry in cal
vc:   current connection
next: next time vc should be scheduled

Execute every cell-slot time:
now ← now mod SIZE
while cal[now] ≠ NULL do
    vc = remove_from_head(cal[now])
    if vc is empty then
        mark vc "eligible"                  //whenever next cell arrives...
    else
        dispatch first cell of vc to scheduler
        next ← next_conf(now, vc.I, vc.L)
        append_to_tail(cal[next mod SIZE])
```

Fig. 4. The regulator algorithm.

The algorithm is based on the calendar queue algorithm [1]. Pseudo code is given in Figure 4. The basic idea is as follows. We maintain a *calendar* called cal, which is a cyclic array of linked lists of active connections. Each entry in the array corresponds to a cell-slot time, and the calendar, at any time, corresponds to the sliding time interval which starts now and spans SIZE cell-slot times into the future. There is a pointer called now advancing cyclically over the array, always pointing to the current entry. The list hanging from the current entry contains structures representing all connections which are eligible for transmission now. The action taken by the algorithm in a time step is as follows. After advancing the now pointer, the list of connections pointed to by the current calendar entry is scanned in order. For each connection on the list, a cell is submitted to the scheduler (if available—see below), the next conformance time is computed, and the connection is concatenated to the list hanging from the entry corresponding to that time.

Note that in general, the next conformance time of a connection may be the current step again: in this case, the connection which was removed from the head of the list will be appended to the tail of the same list.

Some special care is needed for connections which become eligible for transmission, but do not have any cell to transmit yet. For this case, we have a special flag *eligible* to each connection (see Figure 4), and an additional test performed by each incoming cell: When a cell arrives, if the buffer of its connection is empty *and* if the *eligible* flag is set, then it clears the *eligible* flag and joins the tail of the list of the next entry to be scheduled.

A. Analysis and Extensions of the Algorithm

• It turns out that our algorithm has extremely modest resource requirements. For example, an OC12 link containing up to 64K connections requires less 37MHz computation steps on a simple RISC processor, and less than 80KByte of memory (assuming that the speed of each connection is at least 32Mbps).
• The basic algorithm can easily be extended to handle traffic contrained by any number of leaky buckets (e.g., VBR is defined using two GCRAs).

Details can be found in [5].

IV. Analysis of the GCRA Regulator with a Priority Scheduler

In this section we analyze, based on techniques of Cruz [2], the buffer space requirement and the delay added by the combination of our regulator and a simple priority scheduler: recall that the output of the rate controller may have many cells submitted simultaneously to the scheduler, i.e., the regulator ignores the limited bandwidth of the outgoing link; the scheduler has to deal with it, based on its policy. Therefore, the burstiness of streams coming out of the switch may be increased by the scheduler. The analysis below allows us to adjust the parameters used by the regulator so as to conform to given parameters outside the switch. Here, we analyze the simple *strict priority* policy.

For the sake of simplicity, we do our calculations for the single GCRA case, presented in Figure 4. Using Eq. (2), we will work in the (σ, ρ) model. We denote the number of connections sharing the link by n, and the output link capacity by C. We assume that the total reserved bandwidth is less than C. We make our calculations under the assumption that there is no cell loss: we will find the size of the buffer which justifies this assumption.

In the *strict priority policy*, only the highest priority class is served at any given time, i.e., a cell will be served only after all cells of higher priority have been served. We denote the number of priority classes by M, where class 1 is the highest priority. Within each priority class i, we order all connections in some arbitrary order, so that i_j denotes the j-th connection in the i-th priority class. Each connection i_j conforms to the $GCRA(R, N)$ model by virtue of the traffic shaping, with burst size N_{i_j} cells and rate R_{i_j} cells per second. We denote by R_i the total rate of priority i connections, i.e., $R_i = \sum_j R_{i_j}$. Similarly,

we denote by N_i the total possible burst size of priority-i connections, i.e., $N_i = \sum_j N_{i_j}$. Let D_{i_j} denote the maximal delay experienced by a cell of connection i_j arriving at the scheduler at time t. Following [2], we obtain:

$$D_{i_j} \leq \frac{\sum_{k=1}^{i} N_k}{C - \sum_{k=1}^{i-1} R_k} \ . \tag{3}$$

Note that D_{i_j} is a bound for *all* cells in the i-th priority class, since effectively, Eq. (3) bounds the worst-case delay for a cell with priority i. We are therefore justified in denoting $D_i = D_{i_j}$ for any j. Using the fact that a delay of D seconds for a stream with rate R can increase the size of bursts by no more than RD cells, we can bound the size of the buffers in the schedule as follows. The buffer space in the scheduler is equal to the sum of all N_i and the maximum delay in the buffer multiplied by the total rate of cells entering the scheduler. Thus, the total buffer space required for priority i satisfies

$$B_i \leq N_i + R_i \frac{\sum_{k=1}^{i-1} N_k}{C - \sum_{k=1}^{i-1} R_k} \tag{4}$$

and the total buffer space required in the scheduler is simply $B = \sum_{i=1}^{M} B_i$.

Using Eqs.(3,1), we can finally calculate the GCRA parameters which should be used by the rate controller in order to ensure conformance of the stream *leaving the switch*. Consider a stream j_0 of priority i which should leave the switch conforming to (R, N^*). Suppose that the regulator shapes the stream to conform to (R, N'). Clearly, the maximal number of cells from stream j_0 which may be leaving together the scheduler is $N' + RD_i$, where D_i is given by Eq. (3). Thus, the stream j_0 should be shaped with $N' = N^* - RD_i$. This corresponds to a shaper's GCRA L-value of

$$L = \left(\frac{L^*}{I - t} - \frac{D_i}{I} \right) \cdot (I - t) \ , \tag{5}$$

where L^* is the required L parameter of the stream leaving the switch, t is the cell transmission time, I is the reciprocal of the line average speed, and D_i is given by Eq. (3).

V. Analysis of the GCRA Regulator with a WFQ Scheduler

In this section we analyze the buffer space requirement and the delay added by the combination of our GCRA rate controller with a weighted fair queuing (WFQ) scheduler [7]. A weighted fair queuing scheduler consists of n classes of service where each class is associated with a weight W_i so that $\sum W_i = 1$. Within each service class i, we order all connections in some arbitrary order, so that i_j denotes the j-th connection in the i-th service class. Similarly to the previous analysis, each connection i_j conforms to the (R, N) model by virtue of the traffic shaping, with burst size N_{i_j} cells and rate R_{i_j} cells per second. We denote by R_i the total rate of service class i connections, i.e., $R_i = \sum_j R_{i_j}$. Similarly, we denote by N_i the total possible burst size of service class-i connections, i.e., $N_i = \sum_j N_{i_j}$.

Following [7], we can bound the size of the service class buffer in the scheduler as follows. Using the fact that the delay bound provided by WFQ is within one packet of that provided by GPS, the required buffer space is equal to the sum of all N_i plus one cell. Thus, the total buffer space required for service class i satisfies $B_i \leq N_i + 1$, and the total buffer space required in the scheduler is simply $B = \sum_{i=1}^{M} B_i$. Let D_{i_j} denote the maximal delay experienced by a cell of connection i_j arriving at the scheduler at time t. The delay is related to the time it takes to clear the service class buffer B_i. We have

$$D_{i_j} \leq \frac{B_i}{W_i C} \ , \tag{6}$$

where W_i is the service class weight and C is the line capacity. Thus we get

$$D_{i_j} \leq \frac{N_i + 1}{W_i C} \ , \tag{7}$$

Using Eqs. (7,1), we can finally calculate the GCRA parameters which should be used by the rate controller in order to ensure conformance of the stream leaving the switch. Consider a stream j_0 of service class i which should leave the switch conforming to (R, N^*). Suppose that the regulator shapes the stream to conform to (R, N'). Clearly, the maximal number of cells from stream j_0 which may be leaving together the scheduler is at most $N' + RD_{i_j}$, where D_{i_j} is given by Eq. (6). Thus, the stream j_0 should be shaped with

$$N' = N^* - RD_{i_j} . \tag{8}$$

This corresponds to a shaper's GCRA L-value of

$$L = \left(\frac{L^*}{I - t} - \frac{N_i + 1}{I W_i C} \right) \cdot (I - t) \ , \tag{9}$$

where L^* is the required L parameter of the stream leaving the switch, t is the cell transmission time, I is the reciprocal of the line average speed.

VI. Performance Evaluation

In this section we evaluate the performance of the GCRA shaping in a simple network. The network model we use consists of a line of m switches (see Figure 5 for a schematic representation of a network with $m = 4$). This line is the path of a single connection we examine, called the *focus connection* henceforth. In each switch, other connections are multiplexed into the output link with various input and shaping parameters. Specifically, we have three types of connections denoted by A, B and C (see Table I). Each traffic source conforms to $\mathrm{GCRA}(I_{in}, L_{in})$ with the given parameters, and the regulator shapes them to conform to $\mathrm{GCRA}(I, L)$ as given. The focus connection is of type A (setting the source to be of another type does not affect the results in a significant way). In each switch, we inject fresh traffic streams: 9 of type A, 10 of type B, and 10 of type C (so that in each link, there are 30 streams overall). All streams except the focus connection leave the system after one hop (once their conflicts with the focus connection are resolved, their effect is negligible).

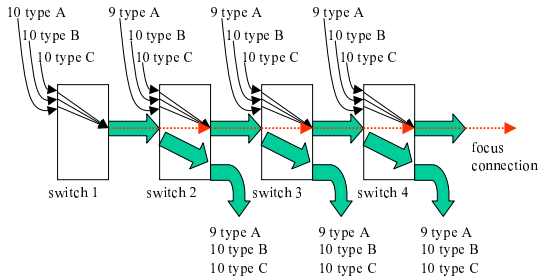

Fig. 5. Network model used in simulations with 4 switches. End-to-end delay and cell spacing were measured only for the focus connection.

TABLE I
CONNECTION PARAMETERS TESTED

Type	(I_{in}, L_{in})	shaped traffic (I, L)		
		$50\%L_{in}$	$25\%L_{in}$	$5\%L_{in}$
A	(20,400)	(20,200)	(20,100)	(20,10)
B	(100,2000)	(100,1000)	(100,500)	(100,100)
C	(25,500)	(25,250)	(25,125)	(25,25)

All traffic sources are generated by a stochastic on/off process with $\mathrm{GCRA}(I_{in}, L_{in})$ parameters. The idea is that the process emulates a source which generates packets: the L_{in} parameter corresponds to the packet size, and the I_{in} parameter corresponds to the average rate. To control the interaction between connections, the arriving time of each packet is uniformly distributed between 0 and the connection period L_{in}/I_{in}.

In this setting, a backlog of non-conforming cells may be generated in the input buffers since $L_{in} > L$, i.e., the regulator shapes the outgoing traffic to be smoother than the incoming traffic. Temporary congestion in the output buffers may be created when packets from several streams arrives at the same time. We measured the performance of the GCRA shaper in terms of buffer occupancy, end-to-end delay, and inter cell arrival time at the destination. The delays and the inter-cell arrival times were measured only for the focus connection. We present results for various values of the shaper's L value, modeling the burstiness of the shaped traffic. We compared L values ranging from $L = L_{in}/2$ (corresponding to rough shaping) to $L = L_{in}/20$ (corresponding to smooth shaping). The (I, L) values we tested are summarized in Table I.

We present results for a simple FCFS scheduler (results for the 2-priority case can be found in [5]). Figure 6 shows the buffer requirement inside the network for the focus connection. We remark that smaller L values at the shaper reduce the buffer space requirement dramatically at *all* switches except the first [5].

An important consequence of space reduction for smoother traffic is that the end-to-end delay was reduced as well when more aggressive shaping has been enforced by the switch regulators, as can be seen in Figure 7. This result is consistent: the more switches we have, the better reduction in end-to-end delay we get—see Figure 8.

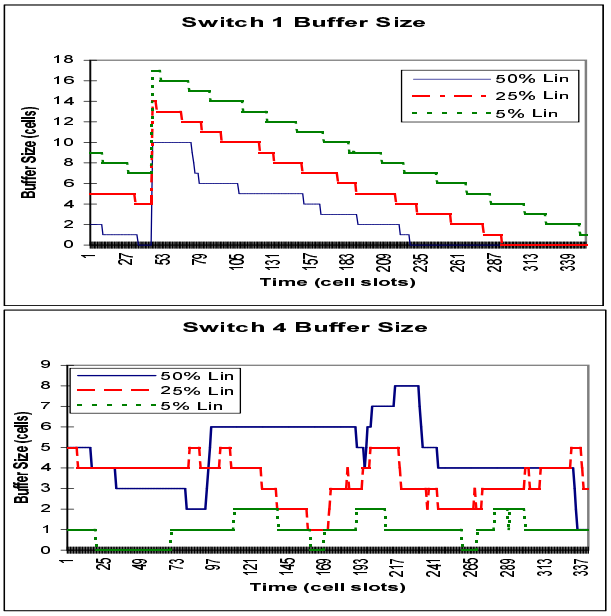

Fig. 6. Buffer occupancy in the first and fourth switches for different values of the smoothing parameter L.

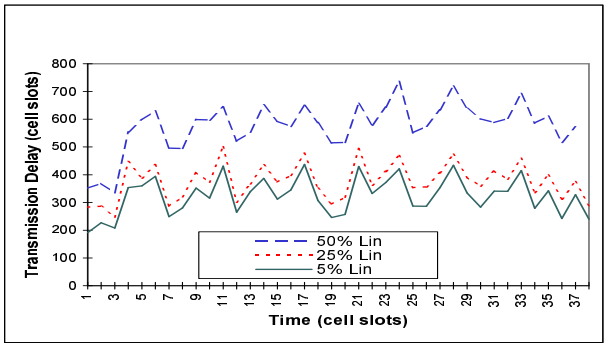

Fig. 7. End-to-end delay for different L values after 4 switches.

This fact is also conspicuous in Figure 9, where the buffer occupancy of the *scheduler* is plotted. The scheduler's buffer represents the link congestion, which is much lower for smoothed traffic.

Our last set of results for the no priority case concerns the inter-cell spacing values, i.e., the jitter (Figure 10). Consistently with our previous observations, we see that the cell spacing in a typical time interval at the 4th link is much better when the streams are highly regulated.

VII. RELATED WORK

Much research effort was devoted to rate control over high-speed networks. Zhang [10] provides an excellent survey. We briefly discuss later papers which are most relevant to our work.

In [6], a comparison is made between two non-work-conserving traffic shaping algorithms, Stop-and-Go [4] and Rate Control Static Priority (RCSP) [11]. They propose to use the

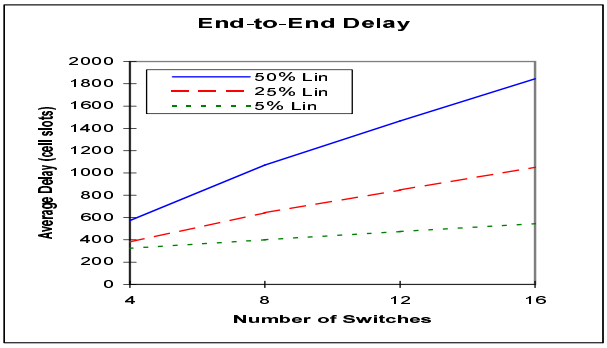

Fig. 8. Average end-to-end delays for different L values as a function of the path length.

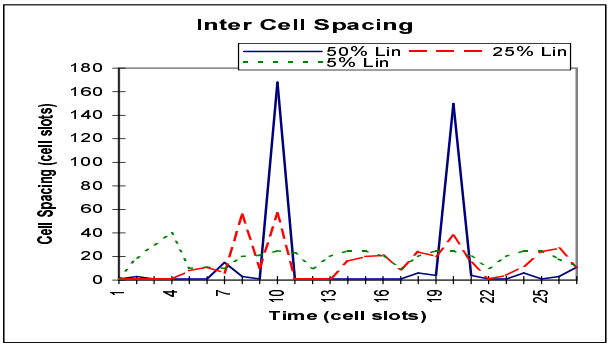

Fig. 10. Cell inter-arrival times for different L values. The standard deviation of the inter-arrival times are 9.81 for $L = 5\%L_{in}$, 15.84 for $L = 25\%L_{in}$, and 32.76 for $L = 50\%L_{in}$.

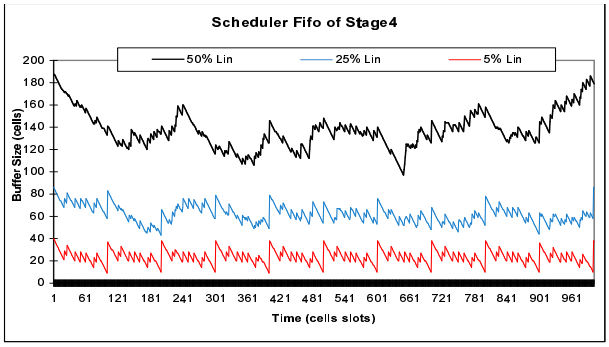

Fig. 9. Buffer occupancy of the link scheduler in the 4th switch.

(σ, ρ) leaky bucket as a traffic shaper, but they do not discuss the implementation details.

In [10], Zhang proposes an implementation of a regulator based on a modified version of calendar queue [1]. In Zhang's algorithm, each entry in the calendar contains linked lists of cells, one list for each priority class. Every "tick," all lists of the current entry are submitted to the scheduler. In this algorithm, since the lists consist of *cells*, it is completely possible (in fact, it is usually the case), that more than one cell associated with one connection will be pointed to by the calendar. The result is that when a long burst of cells for a specific connection arrives, the cells belonging to that burst may wrap around the calendar, thus breaking the FIFO order. The simple solution is to have a sufficiently large calendar. In our solution, however, the basic scheduling entity is a *connection*. This has two advantages: First, since each connection may appear at most once in the calendar, one can easily smooth large bursts. In our algorithm, the size of the calendar is not dominated by the time required to smooth the *largest burst*, but rather it is determined by the inter-cell arrival time of the *slowest connection*. Under realistic circumstances, the latter is usually smaller than the former. In addition, in our algorithm, the space overhead required for implementing the list structure is proportional to the number of connections, which is much better.

Another work closely related to ours is [8], where a fair leaky-bucket rate controller is proposed. One of the main architectural features of [8] is that shaping and scheduling are integrated. Fairness is achieved by interleaving conforming cells in the scheduler. Our implementation, by contrast, decouples traffic shaping from link scheduling, which gives our design several advantages. First, we can support arbitrary priority policies: it is not clear how can one do that in the algorithm proposed in [8]. Secondly, our algorithm allows for simple analyses of the required buffer space and queuing delays, using standard techniques. Moreover, we give explicit construction for dual leaky buckets required for VBR-type connections and simulations of multiple switch network.

REFERENCES

[1] R. Brown. Calendar queues: A fast $O(1)$ priority queue implementation for the simulation event set problem. *Comm. of the ACM*, 31(10):1220–1227, Oct. 1988.

[2] R. L. Cruz. A calculus for network delay part I: Network elements in isolation. *IEEE Trans. Inform. Theory*, 37(1):114–131, Jan. 1991.

[3] D. Ferrari and D. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE J. on Selected Areas in Communications*, 8(3):368–379, August 1990.

[4] S. J. Golestani. A Stop-and-Go queueing framework for congestion management. In *Proceedings of the ACM SIGCOMM*, pages 8–18, 1990.

[5] A. Hagai. Multiple priority, per flow, dual GCRA rate controller for ATM switches. Master's thesis, Dept. of Electrical Engineering, Tel Aviv University, 2000. Available from `http://www.eng.tau.ac.il/~boaz/Hagai.ps.gz`.

[6] H. Z. E. W. Knightly. Comparison of rate-controlled static priority and stop-and-go. *ACM/Springer Multimedia Systems*, 4(6):346–356, Dec. 1996.

[7] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services network—the single node case. *IEEE/ACM Trans. on Networking*, June 1993.

[8] J. Rexford, F. Bonomi, A. Greenberg, and A. Wong. Scalable architectures for integrated traffic shaping and link scheduling in high-speed ATM switches. *IEEE Journal on Selected Areas in Communications*, 15(5):938–950, June 1997.

[9] The ATM Forum Technical Committee. Traffic management specification version 4.0, Apr. 1996. Available from `www.atmforum.com`.

[10] H. Zhang. Service disciplines for guaranteed performance service in packet-switched networks. *Proceedings of the IEEE*, 83(10), Oct. 1995.

[11] H. Zhang and D. Ferrari. Rate-controlled static-priority queueing. In *Proc. IEEE INFOCOM 93*, Apr. 1993.