Priority Scheduling and Buffer Management for ATM Traffic Shaping

Todd Lizambri, Fernando Duran and Shukri Wakid
National Institute of Standards and Technology
Building 222, Room A353, 100 Bureau Drive
Gaithersburg, MD 20899-3207, U.S.A.
(301) 975-8074 (Voice), (301) 975-8254 (FAX)
E-mail: {todd.lizambri, fernando.duran, shukri.wakid}@nist.gov

Abstract

The impact of buffer management and priority scheduling is examined in stressful scenarios when the aggregate incoming traffic is higher than the output link capacity of an Asynchronous Transfer Mode (ATM) traffic shaper. To simultaneously reduce cell loss and extreme delay behavior for two or more classes of service, we show that a dynamic priority scheme is required. We propose a scheduling algorithm where the priority of different service queues is dynamically modified to allow for the provisioning of isochronous services on one of the queues. Buffer management ensures that all service queues are guaranteed a minimum amount of memory, yet available memory can be shared between service queues when necessary. This approach guarantees that no cells are lost under strain conditions until all buffer is exhausted.

Keywords

ATM Networks, priority scheduling algorithm, buffer management.

1. Introduction

The basic function of a traffic shaper is regulating the traffic flow as per the Quality of Service (QoS) negotiated during the session set up to achieve better network efficiency [1].

Traffic can be shaped by placing it into buffers, and delaying its entry into the network, thereby ensuring a more constant flow of traffic in the network. With larger buffers the probability of losing cells decreases but the overall delay increases. Therefore, it is desirable to utilize buffering and scheduling algorithms to regulate QoS attributes for bursty input traffic. The most significant of these attributes are cell loss, cell delay, and the bounds on cell delay variations or jitter.

Complying with the QoS negotiated in a service contract may be a simple matter for one circuit, but it is certainly quite difficult and complex to honor different QoS attributes for multiple circuits having a single output port at the shaper. In general, the performance of a traffic shaper is not only driven by its internal design but by its ability to regulate the incoming flow such that a virtual circuit utilizes the least resources. It is possible to implement traffic shaping with a closed loop interaction with the application, such as a video encoder, to smooth the traffic. These interactions with the applications, and even the protocol stack, are quite complex and often unpredictable. The sensitivity to delay and burstiness of the incoming traffic is very difficult to predict and varies from one customer domain to another. Also, various types of applications are typically integrated, or multiplexed, when reaching the shaper thus making it hard to segregate and regulate each application.

We concentrate on two of the most important internal design factors in traffic shapers: buffer management and scheduling algorithms. We assume that one can always improve performance by increasing microprocessor speed, selecting a more efficient operating system, or even building specialized hardware to replace inefficient software components. However, for a predetermined set of available resources, we examine the impact of buffer management and scheduling algorithms on the two most important QoS attributes, cell loss and delay, under stress conditions. Because of the high cost of wide area output ports, and the unpredictable peak traffic bursts, stress conditions are likely to occur and need to be considered in the design of any traffic shaper. This work also shows that the shaper must dynamically adapt to the unpredictable bursty behavior of the incoming traffic. As shown in this paper, this adaptation can be accomplished by incorporating a scheduling algorithm which assigns priority as a function of the traffic shaper's internal state.

This paper has been organized as follows: Section 2 describes briefly basic existing buffering and scheduling techniques. In Section 3, we describe the traffic scenario to be used in the discussions. Section 4 introduces a dynamic scheduling algorithm. In Section 5, the implementation of the simulations is explained. Section 6 presents the results for the proposed algorithm and compares them against first-in-first-out and round robin schemes and Section 7 provides conclusions.

2. Buffer Partitioning, Discard Policies and Scheduling Algorithms

It can be easily shown that as the input traffic load increases the throughput of the network decreases abruptly after a critical occupancy level is reached. There are many techniques, called *congestion control mechanisms*, that can be put in place to correct, limit, or avoid this traffic condition. A network designer's choice of these mechanisms is driven by the duration of the congestion to be controlled. For example, sustained periods of congestion can be limited through proper capacity planning and network design. Congestion added by a sudden increase in network connections could be limited using connection admission control, dynamic routing, dynamic compression, and end-to-end feedback. Ultimately, instantaneous congestion due to short bursts of traffic on existing connections can be limited using buffering techniques [2]. It is this last mechanism that we will address in the following discussion. We are going to focus on the buffering mechanisms for congestion control that can be implemented in a single network node. The basic elements of this mechanism include *buffer partitioning*, a *discard policy* and a *scheduling algorithm*.

Buffer partitioning delineates the amount of buffer space available to a given queue and defines how space is shared among the different queues. Buffering methods can be classified as one of the types discussed below [3]. In a *Complete Partitioning* scheme, each queue gets a fixed amount of the buffer space. This is the easiest method to implement, but is not efficient because a queue that is starved for memory cannot make use of free space available to another queue. An antipodal approach to *Complete Partitioning* is the *Complete Sharing*, where all the buffer space is fully shared among all the queues. This is very efficient, but it can lead to problems concerning fairness, as a single queue can consume the entire buffer. As a compromise between the previous two policies, the *Sharing with Minimum Allocation* method can be used, which reserves a minimum buffer space for each queue while the rest of the buffer is completely shared among the queues. We have selected this buffering scheme for our testbed, because it is efficient, fair and simple. In addition to these basic buffering schemes, many other variations have been proposed, as in [4] [5] [6].

The discard policy determines whether an incoming cell is to be dropped or placed into the buffer space. Typically, a discard policy is thought of as a mechanism to police the data to ensure that it conforms to a specified service contract that guarantees a certain quality of service. It may also have enough knowledge of the system to discard a cell that already exists in the buffer space (for example, oldest cell in buffer is dropped).

The scheduling algorithm is the component that determines which queue is given the opportunity to transmit a cell that is stored in the buffer. The ideal algorithm would have properties of *efficiency* and *fairness*. The aspect of *efficiency* can be easily measured. However, *fairness*, is not so easily understood. In a system where each class of service has different requirements for acceptable latency and for cell loss that can be tolerated, determining which metrics to use for *fairness* is a bit more subjective. In our attempt to provide fairness, we consider a class of service to be treated fairly if it continues to be serviced and its requirements for cell delay and cell loss are fulfilled. Some of the simplest scheduling algorithms are first-in-first-out (FIFO), round robin (RR), and a "fixed priorities" scheme where a queue with higher priority is always served before a queue with a lower priority. Many other algorithms have been proposed in this area [7] [8] [9] [10] [11].

We will propose a *Dynamically Weighted Priority* scheduling algorithm in this paper that maintains the fairness attribute of FIFO and, at the same time, provides an efficient priority scheme. Further adaptation of the buffering can be achieved by monitoring the congestion levels of the queues (queue occupancy level or queue occupancy increase rate) and using this information as a feedback mechanism to adjust weights, modify buffer partitioning or adjust the discard policy. This will be the basis of future work.

3. Network and Traffic Scenario

Private networks typically have a wide-area access line that carriers are more likely to provide as an OC-3 ATM circuit[1]. Inside a customer premise island, common networking fabric such as Ethernet, frame relay, and ATM converge on a traffic shaper that links the outgoing traffic to the carrier's OC-3 leased line. The traffic reaching the shaper from the local networks must go through a segmentation process that produces ATM cells that are scheduled in various output queues according to their designated class of service. A class of service is either assigned by Q.2931 signaling[2] or more commonly specified during configuration of a "nailed up" circuit. Each queue represents a class of virtual circuits that competes for the outgoing OC-3 bandwidth.

For the purpose of this paper, we only consider traffic shaping at the ATM (link layer) level. We therefore

[1] Optical Carrier 3 at 155 Mb/s

[2] Signaling protocol for layer 3 (Call set up protocol for ATM circuits)

assume that all packets are segmented into ATM cells ready to exit a customer premise through a single wide area port.

Numerous types of circuits have been discussed in the ATM Forum. To understand the impact of various scheduling and buffering techniques on the quality of service (especially cell loss and latency), we have chosen to consider an example containing two classes of service: Constant Bit Rate (CBR) and Variable Bit Rate (VBR). In this paper, we consider two generic classes of service where each class has one or more users as well as one or more circuits. The intention behind choosing these important types of services is to analyze the feasibility of isochronous services in the context of situations where bursty data services can dominate the traffic flow.

We modified an ATM simulator developed at NIST [12] by adding modules for buffer management and priority scheduling. The two classes of service give us insight as to the complexity of providing numerous QoS attributes. Applications such as video conferencing and voice are examples that qualify for a CBR service class, while multimedia exchanges would be an example of a VBR application. Using only those two classes of service, we stressed the system such that all the buffers are full and the aggregate incoming rate to the traffic shaper exceeds the output flow rate to the OC-3 link. We are assuming that there is no throttling mechanism for the shaper to slow the transmitting senders.

4. Dynamically Weighted Priority Scheduling Algorithm

We consider a time dependent "instantaneous priority index" $P_j(t)$ for a jth class of service (a queue) at a given time t to be

$$P_j(t) = \frac{u_j}{\left[w_j(t)\right]^{\beta}}$$

where u_j is the associated fixed priority number, a lower u_j means a higher priority, and $w_j(t)$ is the amount of time the oldest cell in the jth class has waited in queue j. In our implementation of this algorithm, the priority index for each queue is recalculated for every output time slot (based on the speed of the outbound link). The queue with the lowest value of priority index is awarded the time slot and is permitted to transmit a cell during that time. The fixed priority values, u_j, can be chosen arbitrarily (given that the higher priority queue gets the smaller number). For our test case, we have chosen the values 1 and 2 for CBR and VBR classes, respectively. The value for w_j is specified in units of 1/100th of a microsecond. Note that for β=0 we have a fixed priority scheduling where $P_j(t)=u_j$. For a very large β, $P_j(t)$ is heavily influenced by the wait time of the cell and the scheduling mechanism behaves as a *FIFO*. This algorithm can be considered a *dynamically weighted priority* scheme where the weights depend on the state of the queues at a given time.

5. Implementation of the Simulation

The buffering schemes mentioned have been analyzed using a network simulator. The configuration of the traffic shaper consisted of two queues: Queue 1, reserved for the CBR traffic class, and Queue 2, reserved for the VBR traffic class. The total buffer space of the traffic shaper was limited to 1024 entries. The implementation of the buffer space allocation was such that each queue was allotted a guaranteed number of entries (512), but was allowed to utilize additional entries if the buffer space was available. However, if a queue was using more than its guaranteed allotment, it was required to relinquish an entry if all buffer space is occupied and an under-subscribed queue (a queue utilizing less than its guaranteed buffer space allotment) requested an entry [13]. The traffic sources used in this study consisted of CBR sources and VBR sources with a single OC-3 output link at 155 Mbs for which traffic is to be managed, as shown in

Figure 1 (a). The CBR sources were transmitting data at a constant rate of 155 Mbs. The VBR sources were transmitting "bursty" traffic at a rate of 155 Mbs for 2 milliseconds followed by an OFF period where no data was transmitted for 2 milliseconds. The ON and OFF periods of the "bursty" traffic were actually a Poisson distribution with mean values of 2 ms; the square function drawn in Figure 1 (a) and (b) shows only those average values. Note that OC-3 has only been used as an example bandwidth, and that in the simulation scenario no propagation delay was considered (only transmission delay: link speed / bits sent). Service contracts may use OC-3 as an upper bound for outgoing traffic. Incoming traffic may be at rates very different from the two incoming OC-3 links considered in this example.

It is very clear through simulation and visually displaying the states of the buffer that the buffer management and priority scheduling is highly dependant on the incoming frequency and duration of the traffic bursts.

The role of the traffic shaper is to assign the available bandwidth to the cells that are waiting to be sent. Since bandwidth is neither created nor eliminated and the input traffic profile under study remains above the speed of the outgoing link, the total number of cells dropped remains constant. However, in an actual implementation, one must consider the processing delay introduced by a specific traffic shaping algorithm.

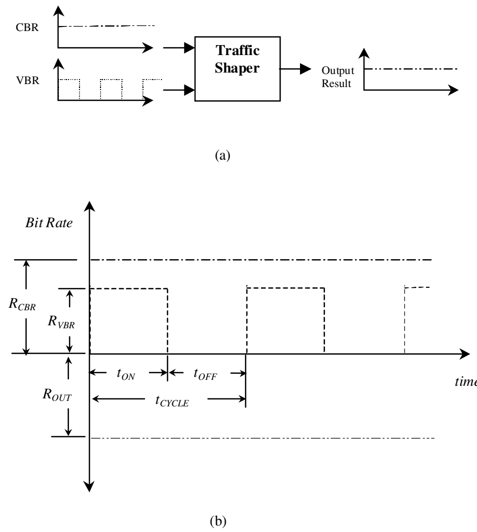

Figure 1. (a) Traffic Shaper Block Diagram. (b) Traffic Analysis Diagram[3]

For the sake of clarification, we will show basic relationship among the parameters of our traffic patterns that will help us establish some bounds. The behavior of the data moving through the traffic shaper can be compared to that of fluid flow through a conduit of finite capacity and maximum throughput. For a cyclical traffic pattern, one can determine if zero cell loss can be achieved by analyzing the average input and output traffic rates over one cycle. If the sum is positive, the traffic shaper is receiving data at a greater rate than it can dispose of the data thus cell loss will eventually occur. However, if the sum is less than or equal to zero, zero cell loss can be achieved. It is important to note that in either case, the amount of buffer space on the traffic shaper will play a major factor in determining if and when cell loss occurs. In
Figure 1 (b), the sum of the averages can be determined as

$$\overline{R} = \overline{R}_{CBR} + \overline{R}_{VBR} + \overline{R}_{OUT}$$

where,

$$\overline{R}_{CBR} = R_{CBR} \ ,$$

$$\overline{R}_{VBR} = \frac{R_{VBR} \cdot t_{ON}}{t_{CYCLE}} \ ,$$

$$t_{CYCLE} = t_{ON} + t_{OFF} \ ,$$

and

$$\overline{R}_{OUT} = R_{OUT}$$

For a given traffic cycle, cell loss can be avoided by sizing the buffer to be greater than or equal to the maximum difference between cells sent to the traffic shaper and cells transmitted from the traffic shaper for any period of time during that cycle. In
Figure 1(b) it can be shown that the difference in cells sent to the shaper and transmitted from the shaper is at a maximum at the time when the VBR source finishes its burst (t = t_{ON}). The amount of data that must be buffered is

$$R_{CBR} \cdot t_{ON} + R_{VBR} \cdot t_{ON} + R_{OUT} \cdot t_{ON}$$

For the traffic profile under study, 37.84 Kilobytes of memory would be required to buffer the excess traffic for the 4-millisecond cycle, only to be deluged with additional excess traffic in the next cycle. Although zero cell loss can be realized for a given period of time, one must determine such possibility given the amount of memory required for buffer space to achieve zero cell loss as well as the delay added due to buffering the data. This is one of many decisions that must be taken into account in determining an optimal design that provides a balance of cell loss, delay, and memory requirements.

6. Results

6.1 FIFO Scheduling

The *FIFO* scheduling mechanism was studied in order to introduce a baseline for comparison. This mechanism represents a scheduling algorithm where cells are processed as they are received. To ensure that a fair comparison is made, this scheduling mechanism was implemented with the same rules for sharing buffer space as the other scheduling schemes analyzed later. However, the policy for determining which data was to be "de-queued" was simply the data that was "First In" (or oldest). Figure 2 shows the cell loss for the *FIFO* scheduling. Because the VBR source is only transmitting 50% of the time (2 ms ON followed by 2 ms OFF), the number of cells dropped from the VBR source (Queue 2) is approximately half of the cells dropped from the CBR source (Queue 1). There is a slight variance due to the sharing of buffer space mentioned above.

---

[3] Note that the traffic analysis diagram in
Figure 1 (b) and following equations depict the input traffic rates as positive and the output traffic rate as a negative value. Also we have shown $R_{VBR} < R_{CBR}$ for intelligibility.

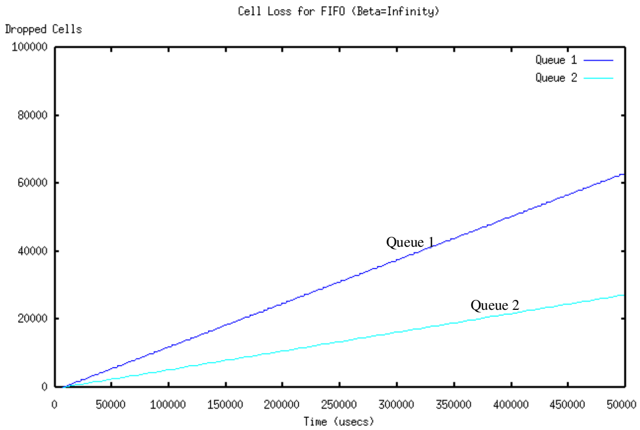

Figure 2. Cell Loss with FIFO Scheduling

Since the traffic for both queues is transmitted at the same rate (155 Mbs), the average cell delay for each queue in the *FIFO* scheduling mechanism will be the same. The delay for these cells when the queues are full is bound by the minimum time to process a 53 byte cell based on the outbound link speed of 155 Mbs (~2.7 ns) and the length of time spent in the queue. The minimum delay of a cell in the *FIFO* scheduling scheme, $d_{\min}$, can be determined from

$$d_{\min} = t_{cell} \times L_{Queue}$$

where, t_{cell} is the transmission delay of one cell, and L_{Queue} is the length of the queue. With a queue length of 1024 saturated, the $d_{\min}$ is approximately 2.8 ms. Since the queue is initially empty and the 1024 entries are shared between the queues (as opposed to a single queue), the average delay for this FIFO is less than 2.8 ms.

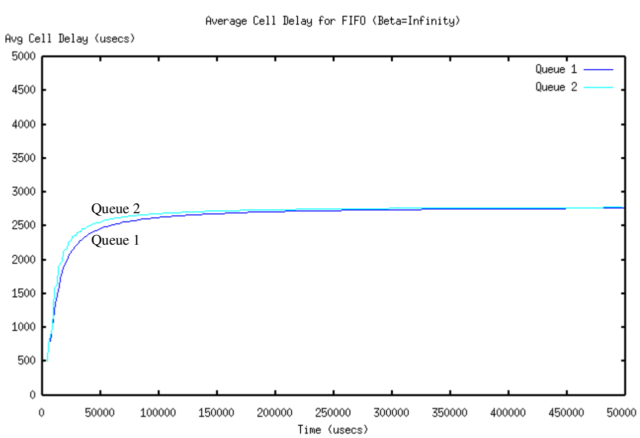

Figure 3. Average Cell Delay with FIFO Scheduling

6.2 Round Robin Scheduling

The *round robin* scheduling algorithm treats each queue with equal priority. Traffic sent to the traffic shaper would be queued to the CBR queue (Queue 1) or the VBR queue (Queue 2) based on the class of service assigned to the data. The "fairness" of the *round robin* scheme occurs when data is to be "de-queued" from the traffic shaper and sent on the outbound OC-3 link. If data exists in more than one queue, then each queue will be serviced in order with an equal share of the outgoing bandwidth. If data exists in only one queue, then that queue will be able to transmit data at the rate of the outbound link (minus any processing delay of the traffic shaper). Figure 4 below shows the cell loss as a function of time for the *round robin* scheduling mechanism with the traffic profile of the queues under stress.

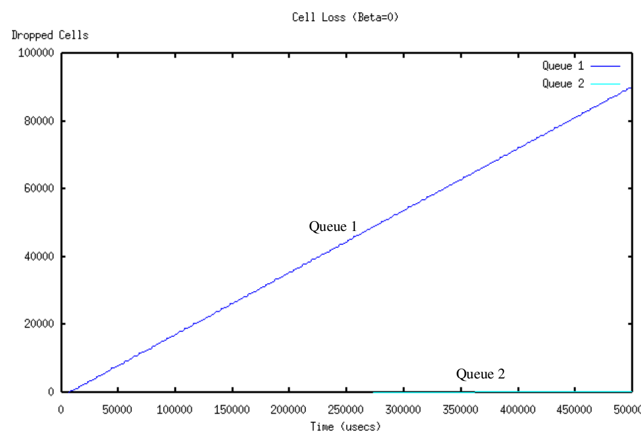

Figure 4. Cell Loss with Round Robin Scheduling

The cell loss of the CBR data is prevalent in this scenario due to the constant arrival of data in Queue 1. Since the arrival of cells into Queue 1 are at a rate that is equal to the outgoing link, CBR cells arriving when the VBR source is idle can be transmitted with no bandwidth to spare. However, when the VBR source is transmitting, the CBR queue (Queue 1) is only serviced for 50% of the outgoing link's time slots (the VBR data receives the other 50%). The VBR data, on the other hand is "bursty" and contains a period of time when data is not being transmitted. This provides a period of relief to the VBR queue (Queue 2) when it can transmit cells and free buffer space for subsequent incoming cells. Queue 1 never receives a relief period and therefore drops cells at a constant rate once its queue is full.Another factor to be considered with the *round robin* scheduling mechanism is the average cell delay. Figure 5 shows the average cell delay as a function of time for the traffic profile described above. As previously discussed, many of the cells from the CBR source have been dropped due to a lack of bandwidth when the VBR source is transmitting. Although the CBR data is given 50% of the bandwidth, the buffer space allocated for the CBR queue is not enough to sustain the duration of the VBR traffic burst. The delay of CBR traffic (Queue 1) is caused due to the queuing of cells when the VBR source (Queue 2) is transmitting data. The VBR data also sustains a delay, although less than the CBR data due to the nature of the VBR traffic.

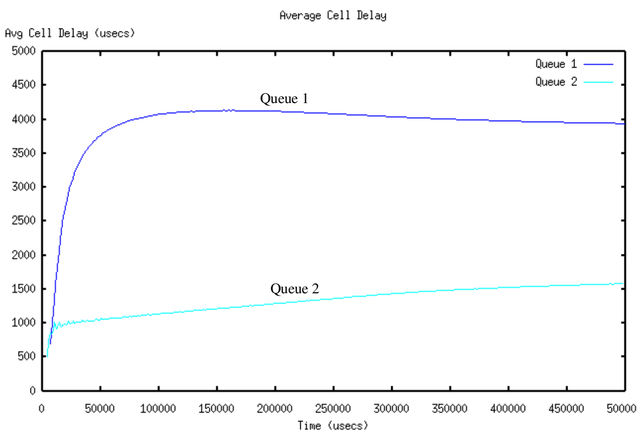

Figure 5. Average Cell Delay with Round Robin Scheduling

6.3 Dynamically Weighted Priority Scheduling

The *dynamically weighted priority* scheduling algorithm provides a means to design a traffic shaper that is priority based but contains a delay sensitive portion to prevent starvation of data. The system under stress was used again but with the *dynamically weighted priority* scheduling policy in place to attempt to adjust the cell loss to be optimal for both CBR and VBR data. To select an optimal value for this β, two graphs are plotted for *cell loss* and *average cell delay* as a function of β for both the CBR (Queue 1) and VBR (Queue 2) queues while the system is under stress conditions (see Figure 6 and Figure 7). This allowed us to estimate the optimal value for β that would minimize cell loss yet retain an acceptable average cell delay (β = 0.9). It is important to note that since the voice traffic is multiplexed with video traffic in the CBR queue (Queue 1) and the bursty data traffic is assigned to the VBR queue (Queue 2), it is only practical to simultaneously minimize delay for Queue 1 and cell loss for Queue 2. This is why we selected a balanced algorithm such as the above formula and decided to use any available memory slots from the neighbor queue when such slots are available and needed.

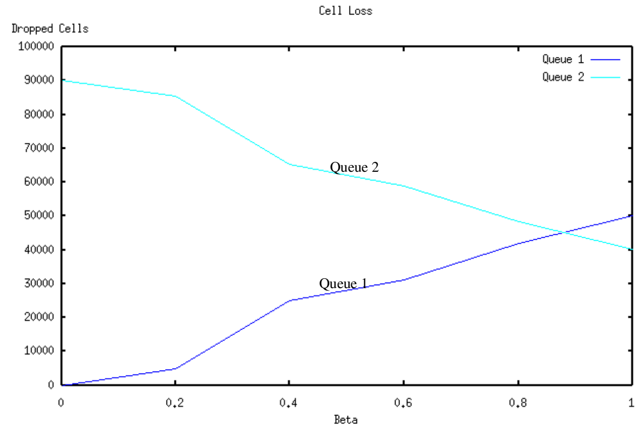

Figure 6. Cell Loss as a Function of Beta Values

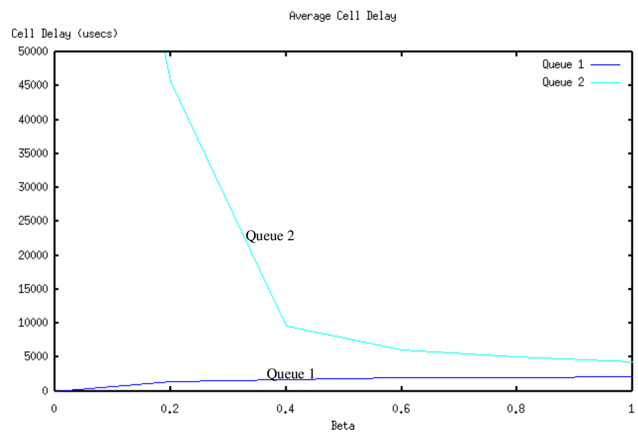

Figure 7. Average Cell Delay as a Function of Beta

Using the value for β obtained from Figure 6 (*β=0.9),* we are able to arrive at a cell loss that is more evenly distributed between the CBR and VBR data (see Figure 8)

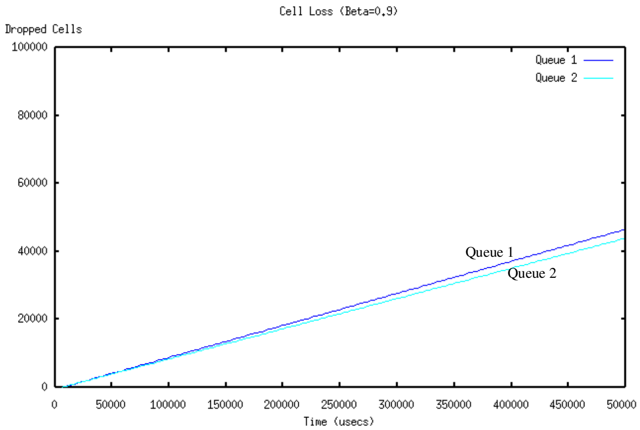

Figure 8. Cell Loss with Optimized Beta Value of 0.9

With the *dynamically weighted priority* algorithm, we were able to optimize the cell loss, but it is also important to consider the effect on the cell delay. Depending upon the application, data delayed beyond a certain threshold may be considered unacceptable and rendered useless by the application.

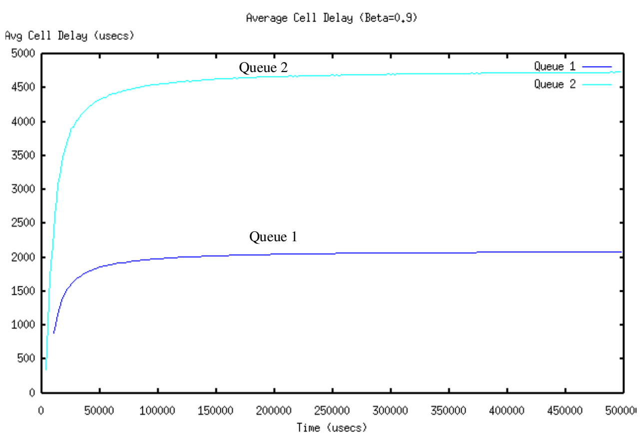

Figure 9. Average Cell Delay with Optimized Beta Value of 0.9

In the *weighted priority* scheduling mechanism, the average cell delay of the VBR data (Queue 2) exceeds that of the CBR data (Queue 1). This can be attributed to the associated fixed priority number, u_j, of the equation used to determine which queue is serviced by the outgoing link.

6.4 Real Input Traffic

In addition to the input traffic profile we chose, we found that it is also desirable to use other sources and compare the results. We have reproduced the same scenario but using real data obtained from the National Laboratory for Applied Network Research (NLANR) collection of traffic samples from their Network Analysis Infrastructure (NAI). These data are publicly available at their web site [14].

The following information is included in the traffic traces: packet length, protocol, source IP address (incomplete for privacy), destination IP address (incomplete for privacy), source port number, destination port number and a local timestamp. The probes are collected during one minute at a fixed times daily.

Using a separate analysis tool, the trace data was analyzed and a network model was generated to be used as input to the ATM simulator. This model consisted of traffic source and sink for each traffic flow found in the trace data. Since the data is strictly IP data, the traffic shaper was configured with two queues: a queue to service UDP data and a queue for TCP data. The UDP queue was designed to service isochronous data arriving at a near constant rate (comparable to the CBR data in our prior simulations) and the TCP queue servicing data of a more bursty nature (comparable to the VBR data). Although the data sources are IP packets, the simulator performs the conversion to ATM cells using ATM Adaption Layer 5 (AAL5) prior to the data reaching the shaper which performs at the ATM layer.

We found that we were able to apply the techniques described in this paper to find a value of β that produced an acceptable average cell delay for the isochronous UDP data and maintained optimal cell loss for both queues.

7. Conclusions

The *dynamically weighted priority* scheduling algorithm provides a mechanism for simultaneously improving the balance of cell loss and delay. To choose an optimal value for β, knowledge of the input traffic signature is required. Our simulation shows, in results not provided here, that β is sensitive to the input burst rate of the VBR traffic, especially the maximum value of such rate. If a user has a good knowledge of the signature of the peak traffic, it would be a simple matter to determine the β by the methods shown in this paper. Otherwise, β has to adapt to the incoming traffic signature. This is of course possible if probes that link the states of the buffers are used to periodically modify β. We plan to produce reference data set signatures of various types of multimedia traffic as well as continue development on an actual prototype to calibrate the simulation results presented here. The modified ATM simulator of NIST, with its graphic interface, will be made available over the Web to help designers understand the impact of incoming traffic profiles on the performance of the shaper.

References

[1] ATM Forum, "ATM Traffic Management Specification Version 4.0" April 1996, ftp://ftp.atmforum.com/pub/aproved-specs/af-tm-0056.000.ps

[2] Raj Jain, "Congestion Control and Traffic Management in ATM Networks: Recent Advances and A Survey", Computer Networks and ISDN Systems, Vol. 28, No. 13, 1996.

[3] Kamoun and Kleinrock, "Analysis of Shared Finite Storage in a Computer Network Node Environment Under General Traffic Conditions", IEEE Transactions on Communications, vol. COM-28, no. 7, 1980.

[4] Tassiulas, Hung and Panwar, "Optimal Buffer Control During Congestion in an ATM Network Node", IEEE/ACM Transactions on Networking, vol. 2, no 4, 1994.

[5] Cidon, Georgiaddis, Guerin and Khamisy, "Optimal Buffer Sharing", IEEE Journal on Selected Areas in Communications, vol. 13, no. 17, 1995.

[6]Ramesh, Rosenberg and Kumar, "Revenue Maximization in ATM Networks Using the CLP Capability and Buffer Priority Management", IEEE/ACM Transactions on Networking, vol. 4, no. 6, 1996.

[7] Demers, Keshav and Shenker, "Analysis and Simulation of Fair Queuing algorithm", ACM SIGCOMM Computer Communication Review, vol.19, no. 4, 1989.

[8] Golestani, "A Self-Clock Fair Queuing Scheme for Broadband Applications", Proceedings of IEEE Infocom 1994.

[9] Stiliadis and Varma, "Design and Analysis of Frame-Based Fair Queuing: A New Traffic Scheduling Algorithm for Packet Switched Networks", Proceeddings of IEEE Sigmetrics 1996.

[10] Suri, Varghese and Chandranmenon, "Leap Forward Virtual Clock: A New Fair Queuing Scheme with Guaranteed Delays and Throughput Fairness", Proceedings of IEEE Infocom 1997.

[11] Bennet and Zhang, "Hierarchical Packet Fair Queuing Algorithms", IEEE/ACM Transactions on networking, vol. 5, no. 5, 1997.

[12] NIST ATM simulator, http://www.hsnt.nist.gov/misc/hsnt/prd_atm.sim.html

[13] G.Wu, J.Mark, “A Buffer Allocation Scheme for ATM Networks: Complete Sharing Based on Virtual Partition”, IEEE/ACM Transactions on Networking Vol. 3, Num. 6, Dec 95.

[14] National Laboratory for Applied Network Research, http://moat.nlanr.net

Acknowledgements

The authors received valuable comments and advice from David Su and assistance with the ATM simulator from Nada Golmie.