# Performance Evaluation of Dynamic and Static Bandwidth Management Methods for ATM Networks

Sponsor: Sprint

Cameron Braun

David W. Petr

Technical Report TISL-10230-5

Telecommunications and Information Sciences Laboratory

Department of Electrical Engineering and Computer Science

University of Kansas

December 1994

# Chapter 1

# Introduction

**1. Introduction**

The future of telecommunications seems almost limitless with fiber optics allowing transmission rates measured in gigabits/sec and memory chips approaching 100's of megabits per chip. But as large as these resources seem, they are not infinite. Eventually there will be enough users with enough network activity to use up any network's resources. For these reasons network resource and traffic management is a very important topic to study.

But what are the resources in a packet-oriented network and why is it important to manage the resources? While there are many network resources, the important resources for this technical report are the link bandwidths (rates) and the buffer (or queue) capacities. The link bandwidths specify the maximum rate at which information may be transmitted while the buffer sizes control how many packets may be queued at a switch before a loss occurs. If the amount of traffic being offered to a link exceeds the link capacity then more information goes into a node than comes out, which eventually leads to buffer overflow and packet loss. There are several ways to control this problem. Call admission control limits the number of users that are allowed into the network at any given time. Queue management methods include weighted round-robin schemes that give slots belonging to inactive queues to one of the active queues [Mezg94]. Queue overflows can be caused by improper queue management and, conversely, proper queue management can reduce the size of buffers needed at nodes in the network. Policing methods can drop packets (or mark them as discardable) that are not conforming to the specified input traffic characteristics. Shapers can be used by customers to control

the rate and burstiness of traffic before it enters the ATM network.

### 1.1 Motivation for Bandwidth Management

One important aspect of network management, and the focus of this technical report, is bandwidth management. The term bandwidth management can mean a bandwidth calculation, a bandwidth allocation, or a bandwidth limitation. A bandwidth calculation does not guarantee bandwidth but simply computes what a user would utilize if there were no other competing users in the network. A bandwidth guarantee actually assigns bandwidth to users as a means of resolving contention between them. The third meaning of bandwidth management, bandwidth limitation, penalizes a user if an upper bandwidth limit is exceeded.

By performing bandwidth calculations it is possible to monitor current or predicted network usage. Users are not constrained, but it is possible to calculate the bandwidth that each user requires. From these calculations the network manager can make decisions on routing and call admission control. These calculations do not guarantee that the users will get the bandwidth as calculated by the network manager; it is just a tool that is used to control the load on the network links. Because the users are not constrained by the bandwidth calculation there may still be competition for bandwidth.

By allocating bandwidth to a user, a network manager is guaranteeing information throughput which is important because a user will pay for the connection and will expect a certain quality of service (QoS) on that connection. Quality of service is measured by several characteristics such as delay, loss, and throughput. The user may get better QoS (higher throughput, lower loss, etc...) through the network under favorable conditions, but will always maintain the guaranteed QoS. If the bandwidth allocation is done by a queueing structure it is possible to segregate traffic so that contention is limited. Traffic contention can cause a degradation in the performance of one user due to the traffic from another user. Clearly users do not want to be controlled by other users.

As mentioned above, a connection in a packet switched network can have a varying bandwidth. If

the bandwidth allocated for the connection was equal to the peak rate of the connection, the QoS would be excellent but there would be wasted bandwidth when the source is not operating at its peak rate. An ideal bandwidth allocation scheme should therefore release bandwidth that is not currently being used and adapt to the characteristics of the source.

Bandwidth allocation (as used in this technical report) is similar to bandwidth reservation in that the bandwidth is guaranteed to be available to the customer; it differs in that if the customer does not use all of the allocated bandwidth, the unused portion is made available to others. It is for this reason that bandwidth allocation is a more important issue for packet switched networks than circuit switched networks. Circuit switched networks use bandwidth reservation: a dedicated circuit with "hard bandwidth limits" is a permanent structure for the duration of the call. Packet switched networks on the other hand, transmit individual packets through the network so the bandwidth used during a connection can vary greatly and could interfere with traffic from other users if not managed properly.

Bandwidth limitations can be enforced by a network policer (a module that allows traffic through if it conforms to predetermined traffic characteristics) that will discard packets or mark them (indicate that this packet may be discarded by later modules if congestion occurs) in order to maintain the bandwidth limit. A customer can also impose an upper bandwidth limit by installing a traffic shaper that will queue ATM cells and then release those cells so that the rate and burstiness of the source conform to specified limits.

### 1.2 Dynamic vs. Static Bandwidth Management

From the name, it is obvious that dynamic bandwidth management changes the bandwidth associated with a particular user over time. This process is done by monitoring the user traffic to determine its behavior or receiving user requests and then allotting available network resources and/or adjusting network controls to accommodate this traffic. The important thing to notice about dynamic bandwidth management is that due to the constant user monitoring, the effects of the bandwidth management are always visible to the user. Since the bandwidth management scheme tracks the traffic

characteristics on a long term basis, the user cannot increase the bandwidth usage very much on a short term basis without hitting the upper bandwidth limit set by the allocation algorithm. If the user increases the amount of traffic being sent on a long term basis or requests more bandwidth, then the management scheme will associate more network resources (if available) with the user, and if the user reduces the amount of traffic being sent on a long term basis or requests less bandwidth, the algorithm will reduce the amount of network resources that had been associated with that user.

Static bandwidth management, as the name implies, does not change over time. Because there is no monitoring of the user traffic and no changes in the associated resources and/or controls, the effects of the static bandwidth management are only evident if the user tries to use more bandwidth than currently agreed upon. If this does happen the user may not get more information through, but may be clamped by either the static bandwidth limit or bandwidth allocation.

### 1.3 Technical Report Outline

The general question that this technical report addresses is, "How can bandwidth be managed efficiently among customers to comply with their desired QoS requests?" It is impossible for this question to be completely answered here so this technical report will focus on the performance of three bandwidth management schemes: a static admission control policy based on bandwidth calculations, a static bandwidth allocating queue server, and a dynamic traffic (bandwidth) shaper.

This chapter has been an introduction to the topics that will be covered in the following chapters. The three bandwidth management schemes will be presented in Chapter 2 along with performance evaluations used for each scheme and conclusions that can be drawn from those evaluations. The last chapter, Chapter 3, will present the general conclusions about traffic management and bandwidth allocation methods and possible future work ideas.

# Chapter 2

# Bandwidth Management Methods

**2. Bandwidth Management Methods**

**2.1 Equivalent Capacity**

The equivalent capacity is a basis for admission control that is presented in [IBM91]. The equivalent capacity is an estimate of the bandwidth that is needed to serve a queue being fed by a specified number of sources with known peak rates, average rates, and mean burst lengths. Equivalent capacity also depends upon the buffer size and the desired cell loss rate. A network provider can then calculate the equivalent capacity of all existing calls plus any new call and base admission control on these calculations: if the equivalent capacity at any network queueing point is greater than the capacity of the link serving the queue, the call cannot be accepted. This algorithm is very simple and [IBM91] shows some promising results. The IBM paper shows that equivalent capacity can predict the amount of the network link capacity a source will need to meet the specified QoS (cell loss) requirements. Simulations are performed using different numbers of sources with different average rates and burst sizes. However [IBM91] shows results for simulations using only homogeneous sources with exponential packet lengths. This section presents results of simulations using combinations of sources with various length distributions.

**2.1.1 Review of Equivalent Capacity**

As stated earlier, equivalent capacity gives an estimate of the capacity that is needed to serve a queue so that the cell loss specifications of the sources are met. The approach that was taken by [IBM91] involves calculating two capacities and using the smaller one. One calculation assumes an

exponential on-off (fluid-flow) source and the other also uses exponential on-off sources but assumes that statistical multiplexing is the dominant factor in the statistical characteristics of the traffic stream.

The equation for calculating the equivalent capacity for a single source using the fluid-flow approximation is shown below:

$$\hat{c} \approx \frac{\alpha b(1-\rho)R_{peak} - x + \sqrt{[\alpha b(1-\rho)R_{peak}]^2 + 4x\alpha b(1-\rho)R_{peak}}}{2\alpha b(1-\rho)}$$

where $\alpha = ln(1/\varepsilon)$, $b$ is the average burst size, $R_{peak}$ is the peak rate of the source, $\rho$ is the source utilization, $x$ is the buffer size, and $\varepsilon$ is the desired buffer overflow probability (the QoS). When this equation is extended to N multiplexed sources the equation becomes:

$$\hat{C}_{(F)} = \sum_{i=1}^{N} \hat{c}_i$$

For the stationary approximation the equation for the capacity is:

$$\hat{C}_{(S)} \approx m + \alpha'\sigma$$

where

$$\alpha' = \sqrt{-2\ln(\varepsilon) - \ln(2\pi)}$$

In the above equations, $m$ is the mean aggregate bit rate and $\sigma$ is the standard deviation of the aggregate bit rate:

$$m = \sum_{i=1}^{N} m_i \qquad \sigma^2 = \sum_{i=1}^{N} \sigma_i^2$$

The final capacity, $\hat{C}$, is the minimum of $\hat{C}_{(S)}$ or $\hat{C}_{(F)}$. The above equations are easily implemented in a computer program, but obtaining the values for all of the parameters is not trivial. For example, if this admission control scheme were to be implemented in a real system with real sources, it would be necessary to obtain those values. For statistical models the values can be obtained through mathematical methods, but for real sources it would be necessary to monitor the traffic and calculate the values from those measurements.

### 2.1.2 Performance Evaluation

#### 2.1.2.1 Approach

Simulations for this paper were performed using the Block Oriented Network Simulator (BONeS) [BONeS] on SUN workstations. The general block diagram for the simulations is shown in Figure 1. The sources (described later) generate packets that are segmented into ATM (Asynchronous Transfer Mode) cells which are multiplexed into a FIFO that is served at a rate equal to the equivalent capacity. By using the specified buffer size and serving the FIFO at the calculated rate, we expect the cell loss due to buffer overflow to be below the specified probability (QoS).

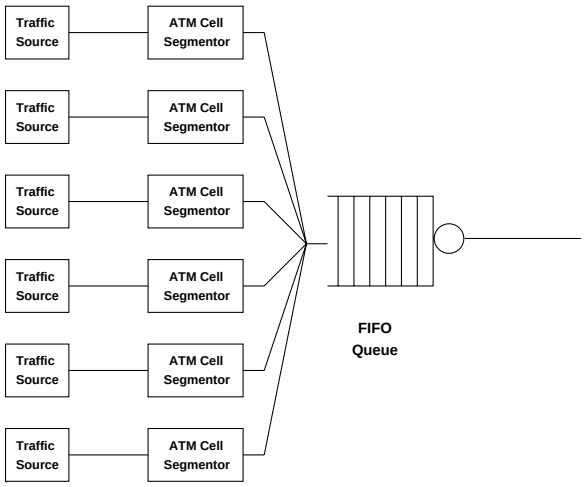


**Figure 1.** System Block Diagram

Since equivalent capacity is intended to guarantee a cell loss ratio (CLR) we monitored the number of cells that were rejected from the FIFO and also monitored the queue fill. If the equivalent capacity was smaller than the required capacity, then the number of cells lost would indicate how much the queue is over-utilized. By the same token, if the equivalent capacity was larger than the required capacity for that source, the queue fill would give an estimate of how under-utilized the queue was.

#### 2.1.2.2 Sources and Configurations

The simulations use six types of traffic sources: three data sources (exponential, modeled, and trace) and one each of image, video and voice. The exponential data source uses a simple decaying exponential for the packet length distribution, while the modeled data source uses a bi-modal [EBNTa] distribution, but both sources have exponentially distributed silence times. The trace data source outputs packets according to files containing actual ethernet data obtained from Bellcore [Lel94]. The image source is a very bursty source that uses a reverse truncated exponential length distribution to generate large packets (ave: 400,000 bits; max: 500,000 bits) with exponential silence times. The video source outputs fixed length packets with exponential interarrival times, while the voice source generates constant bit rate (CBR) traffic with fixed length packets.

For the evaluation of equivalent capacity we used 6 simulation configurations:

— 6 Homogeneous Exponential Data Sources
— 6 Homogeneous Modeled Data Sources
— 6 Homogeneous Trace Data Sources
— 6 Homogeneous Image Sources
— 6 Homogeneous Video Sources
— 1 Modeled Data, 1 Image, 1 Video, 1 Voice

The simulation configurations used were chosen to test the equivalent capacity under widely different source conditions. The exponential sources were used to verify the results obtained in [IBM91], the modeled sources were used because they are commonly used models, and the Bellcore trace data sources were used because the characteristics are very different from the modeled and exponential sources [Lu94]. The image, voice, and video sources represent different types of traffic and will show the flexibility of the equivalent capacity algorithm.

#### 2.1.2.3 Equivalent Capacity Calculations

The equivalent capacity was needed before the BONeS simulations could be run. This section presents those calculations along with a brief discussion of those capacities and what they suggest about equivalent capacity. For all simulations the target cell loss ratio, QoS, was $1 \times 10^{-4}$ and the

buffer size was ten times the average source packet size. The target cell loss ratio was set to $1 \times 10^{-4}$ so that the simulations would run quickly.

Table 1 shows the comparison of the average bit rates to the calculated bit rate from equivalent capacity for all 6 source configurations. This table seems to indicate that the calculated bit rate for all the source types should be adequate to guarantee the QoS target. This assumption is based on the fact that the calculated bit rate is greater than the average rate for all source configurations and for some configurations this difference is very large. Of particular notice is the calculated bit rate for the image source which is over three times the value of the average rate.

**TABLE 1.** Capacity Results for Data Sources

| Traffic Source Type | Average Bit Rate (Mbps) | Equivalent Capacity (Mbps) |
|---|---|---|
| Exponential Data | 0.588 | 0.863 |
| Modeled Data | 0.588 | 0.862 |
| Trace Data | 1.340 | 2.830 |
| Image | 0.422 | 1.420 |
| Video | 5.5 | 6.7 |
| Mixed | 6.59 | 11.13 |

Since the video source is a generating packets at a high rate it is almost a CBR source (not very bursty) and we would expect equivalent capacity to give an estimate that is only slightly larger than the average rate. According to Table 1, equivalent capacity does give a slight overestimate of the average rate, so again it would seem that equivalent capacity should guarantee the cell loss for the video source.

The final simulation used four different sources and again the equivalent capacity seems to have overestimated the average aggregate bit rate (see Table 1). The apparent overestimate is primarily due to the image source which had such a large overestimate in the homogeneous image simulations. The value of the calculated capacity is almost twice the average rate so it would be expected that the equivalent capacity would guarantee the cell loss for this simulation also.

The previous table showed only a comparison of bit rates before the simulations were run so the assumptions concerning the performance of the equivalent capacity were not based on simulation results. The following section presents the results of the actual BONeS simulations which will show for certain whether or not the equivalent capacity truly under-estimated or over-estimated the sources.

#### 2.1.2.4 Results

The first simulations were performed with six identical exponential data sources in order to verify the results of our simulations. (These are the same sources used in [IBM91].) From Table 2, it is clear that equivalent capacity can estimate the capacity required for the exponential and bi-modal sources since the cell loss for those two sources are less than $1 \times 10^{-4}$, but performs poorly with the trace data from Bellcore. A closer examination of the data in Table 2 shows that the estimate for the exponential source is accurate since the cell loss for this source was close to, but still below, the target. For the bi-modal source equivalent capacity over-estimates the required capacity since the source's cell loss ratio is well below the target. The estimate for the trace data source, on the other hand, is not large enough to attain a $1 \times 10^{-4}$ cell loss ratio. Instead, the cell loss ratio is two orders of magnitude over the target CLR, indicating that even though the equivalent capacity for the trace data source is more than twice the average bit rate, the equivalent capacity still under-estimates the required capacity for this source. The reason for the under-estimate is due to the fact that the burst length used by equivalent capacity does not give an indication of correlation of large packets that is present in the trace data [Lu94]. In an attempt to improve the performance of equivalent capacity when tested with trace data, the average packet size of the trace data file was recalculated such that two packets with interarrival times less than 2 milliseconds were combined into a single packet. (This is simply for calculation purposes and does not change the trace data file.) The new average packet size changed from 3474 bits to 7649 bits. It was hoped that this new average packet size would reflect the correlation that is seen in the trace data. The increase in the calculated average packet size increased the *Mean Burst Length* parameter and also the queue length (10 * *average packet length*). The equivalent capacity, however, was the same

(17 *x* $10^6$ bps). The end result was that there was zero cell loss and the average queue fill was only 20 out of a buffer size of 3200 cells. While this extra calculation solved the problem for this simulation, it is not a calculation that can be done by a network manager in real-time at call setup.

**TABLE 2.** Loss Results for Data Sources

| Data Source Type | Maximum Queue Fill (cells) | Measured Cell Loss Ratio | Queue Capacity (cells) |
|---|---|---|---|
| Exponential | 375 | 5e-5 | 375 |
| Modeled | 270 | 0 | 375 |
| Trace | 2700 | 1e-2 | 2700 |

Table 3 presents the results for the simulations using six identical image sources. This table shows no loss, and very small maximum and average queue fills. The rate over-estimation and, hence, the absence of cell loss is due to the fact that the burst size and average rate do not characterize the source's on-off behavior. The source bursts at the peak rate for about 40 ms and then does not transmit anything for about 1 second. Equivalent capacity treats the source as if it were transmitting constantly at the average rate.

**TABLE 3.** Loss Results for Image

| Maximum Queue Fill (cells) | Average Queue Fill (cells) | Cell Loss Ratio | Queue Capacity (cells) |
|---|---|---|---|
| 3655 | 427 | 0 | 8522 |

The loss results in Table 4 show that the cell loss ratio for the homogeneous video simulations is more than an order of magnitude below the desired ratio. The average queue fill is very small, but since there was loss, obviously the maximum queue fill was 50 cells, the buffer size.

The last simulation used four different traffic sources to investigate how the equivalent capacity performs for combinations of heterogeneous sources. The cell loss results for this simulation are shown in Table 5. For this case, there was no loss and the queue fill was small compared to the buffer size, 10600 cells. The reason for the over-estimation for the combination simulation is primarily due

**TABLE 4.** Loss Results for Video

| Maximum Queue Fill (cells) | Average Queue Fill (cells) | Cell Loss Ratio | Queue Capacity (cells) |
|---|---|---|---|
| 50 | 5 | 5e-6 | 50 |

to the image source. The image source calculation adds approximately 1 Mbps to the equivalent capacity, but adds 10,400 cells to the buffer size. As can be seen from Table 5, the buffer does not even fill to 15% of the capacity resulting in wasted buffer resources.

**TABLE 5.** Loss Results for Combination

| Maximum Queue Fill (cells) | Average Queue Fill (cells) | Cell Loss Ratio | Queue Capacity (cells) |
|---|---|---|---|
| 1250 | 250 | 0 | 10600 |

#### 2.1.2.5 Conclusion

The results that were presented in this section show that equivalent capacity has some limitations when the source does not conform to exponentially distributed packet lengths and silence times. In particular, the sources that had the worst performance were the trace data and the image sources. The length and silence time distributions for the trace data in no way resemble exponential distributions, and there is correlation between the larger packets resulting in a very bursty source. The image source also did not match the exponential distribution, but had a higher probability of larger packets, also resulting in a bursty source. Since these are both bursty sources it seems reasonable that the performance trends for the two sources would be the same, but the simulations results show opposite trends. For the trace data source the equivalent capacity *under*-estimated the capacity needed by the source resulting in a CLR that was higher than that specified, while for the image, equivalent capacity *over*-estimated the required capacity so that bandwidth was being wasted.

## 2.2 $T_1/T_2$ Queue Server

### 2.2.1 $T_1/T_2$ Basics

The $T_1/T_2$ queue [Sri] is a very simple bandwidth allocation mechanism that uses timers to control the amount of service given to two separate queues. The parameters $T_1$ and $T_2$ are the timer lengths for queues 1 and 2, respectively. Figure 2 shows the block diagram of the $T_1/T_2$ queue as implemented in BONeS (Block Oriented Network Simulator). The structures of interest are the two separate queues (*High Priority Queue* and *Low Priority Queue*) and the two separate service modules (*High Priority Queue Service Module* and *Low Priority Queue Service Module*). The two priority queues are where the packets are stored and the two service modules contain the two timers that control the length of time that each queue gets service.

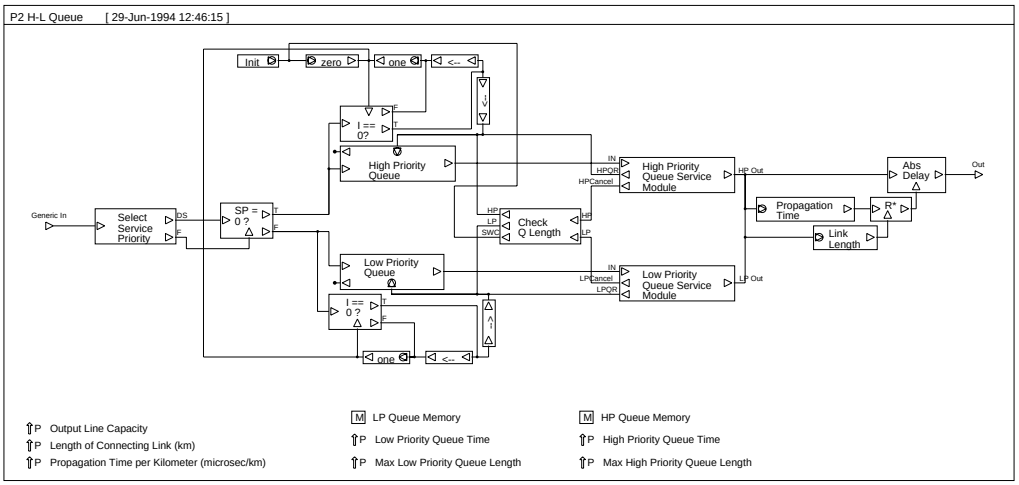


**Figure 2.** BONeS Diagram of $T_1/T_2$ queue

The queue has three modes of operation. The module will remain idle while waiting for a packet to enter the module when both high and low queues are empty. When the module is overloaded the high and low queue service is limited by the timers. The third mode of operation happens when the module is underloaded and high and low queue service is limited by the number in the queue. When there are packets in the module, a queue will get service until either the queue is empty or the timer for that

queue expires. This service method allows the $T_1/T_2$ queue to automatically adjust to input traffic. If queue 1 is receiving heavy traffic while queue 2 is receiving no traffic then queue 1 will be able to use all of the output bandwidth. However, if queue 2 begins receiving traffic, the service rate for queue 1 will be limited to accommodate the change in queue 2 traffic. When both queues are receiving heavy traffic, the service rate for each queue will be limited by the pre-determined bandwidth allocation (described below).

Because this queue module is a static bandwidth allocation mechanism (assuming that $T_1$ and $T_2$ are fixed), the best way to see the effects of the bandwidth allocation is to maintain the $T_1/T_2$ queue in a state of overload. By graphing the number served from each queue versus time the bandwidth split due to the timers is clearly visible (see Figure 3). The bandwidth allocation can be calculated as follows:

$$Queue\ 1\ Guaranteed\ Bandwidth\ Percentage = \frac{T_1}{T_1 + T_2}$$

$$Queue\ 2\ Guaranteed\ Bandwidth\ Percentage = \frac{T_2}{T_1 + T_2}$$

where $T_1$ and $T_2$ are the timer lengths for queues 1 and 2, respectively. As an example, if $T_1$ = 0.75ms and $T_2$ = 0.375ms, then:

$$Queue\ 1\ Guaranteed\ Bandwidth\ Percentage = \frac{0.75ms}{0.75ms + 0.375ms} = 0.667$$

$$Queue\ 2\ Guaranteed\ Bandwidth\ Percentage = \frac{0.375ms}{0.75ms + 0.375ms} = 0.333$$

In other words, when the system is in overload, Queue 1 will have twice the bandwidth of Queue 2.

#### 2.2.2 Performance Results

To better understand the performance of the $T_1/T_2$ queue it is important to understand the topology of the network in which the queue structure was used [EBNTa, EBNTb]. Figure 4 shows the topology of the network including the customer premises muxs (routers) and hubs (switches). The sources (not shown) would be connected to each customer premises mux and would be of different types (voice,

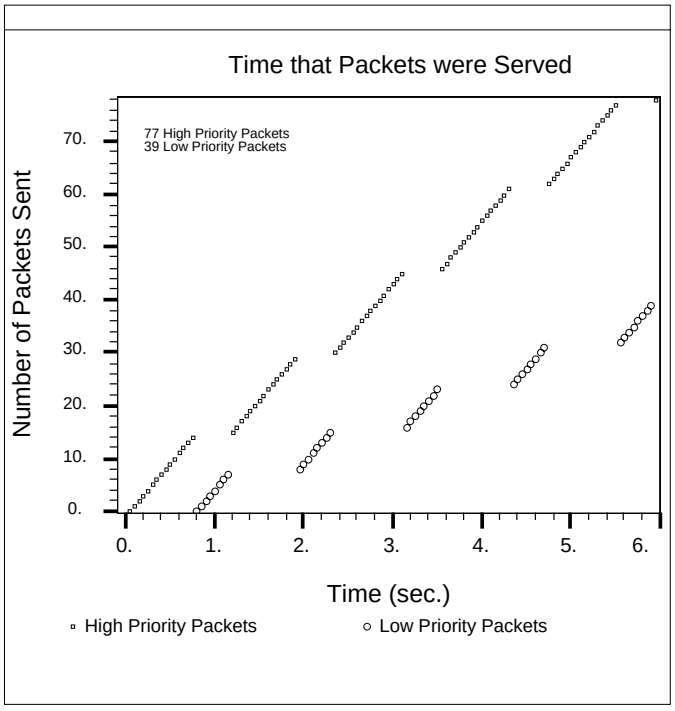


**Figure 3.** Bandwidth Split for $T_1/T_2$ Queue

video, data, etc.). The voice source outputs constant length packets at constant intervals. The video source models compressed video traffic by generating constant length packets with exponential interarrival times. Data sources generate packets from a bi-modal length distribution and use exponential silence times. The image sources generate large packets from a reverse-truncated exponential with exponential silence times. The video, data, and image source are described in more detail in section 3.1.2.2, [EBNTa], and [EBNTb]. These sources are multiplexed on to the access lines and then the access lines are multiplexed onto the trunk lines. At each of the multiplexing points, the $T_1/T_2$ queue structure is used to control the performance of the different types of traffic. Voice and video traffic shared Queue 1, and data and image traffic shared Queue 2. The values of $T_1$ and $T_2$ were set to allocate sufficient average bandwidth for traffic in each queue.

Simulations of the above network have shown that this partial separation of traffic is good for the video and voice traffic because they have similar traffic characteristics (non-bursty, voice - CBR, video

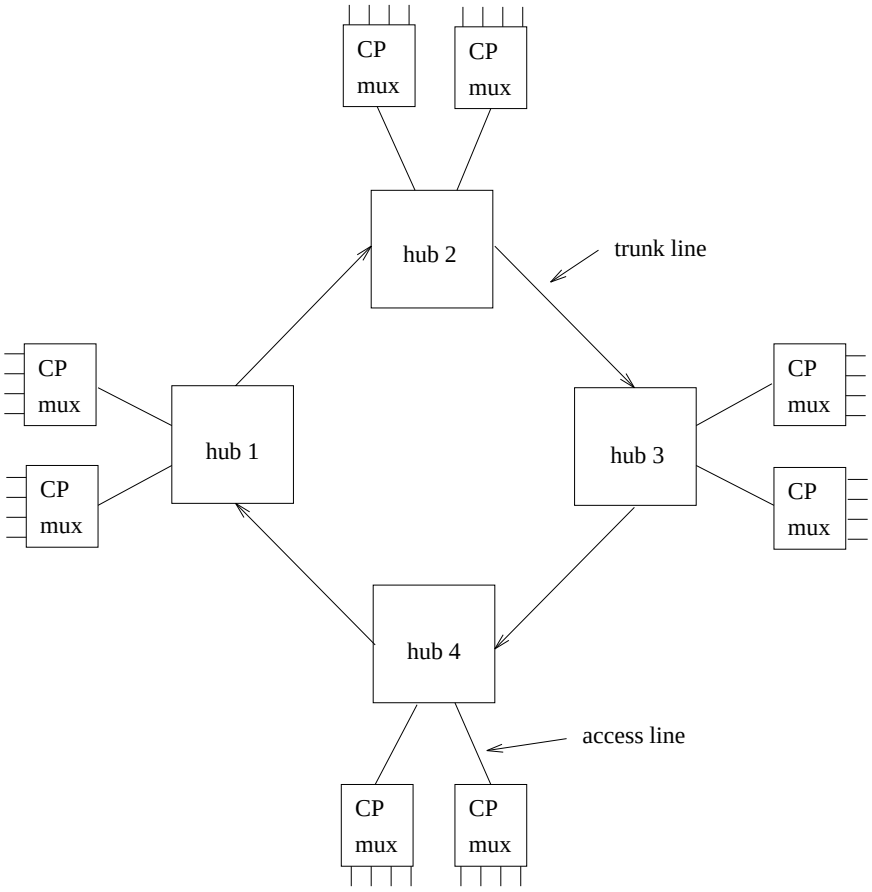


**Figure 4.** Top Level of $T_1/T_2$ Network

- essentially CBR) but gives bad performance for the data traffic due to the queue sharing with image traffic. Figure 5 shows that the cell loss ratio (CLR) for the traffic of Queue 1 (labeled res) decreases with increasing buffer size. The CLR for the traffic of Queue 2 (labeled std) does not decrease very much with increasing buffer size. The difference in CLR for reserved and standard data traffic can be attributed to the fact that the standard data shares Queue 2 with the bursty image traffic. By generalizing the $T_1/T_2$ queue it would be possible to segregate the traffic further such that each traffic type had its own queue so that the queue parameters could match the traffic characteristics.

Figure 3 shows how the $T_1/T_2$ queue can allocate bandwidth, but a problem exists if the original source packets are very large. Large source packets are segmented into many ATM cells before

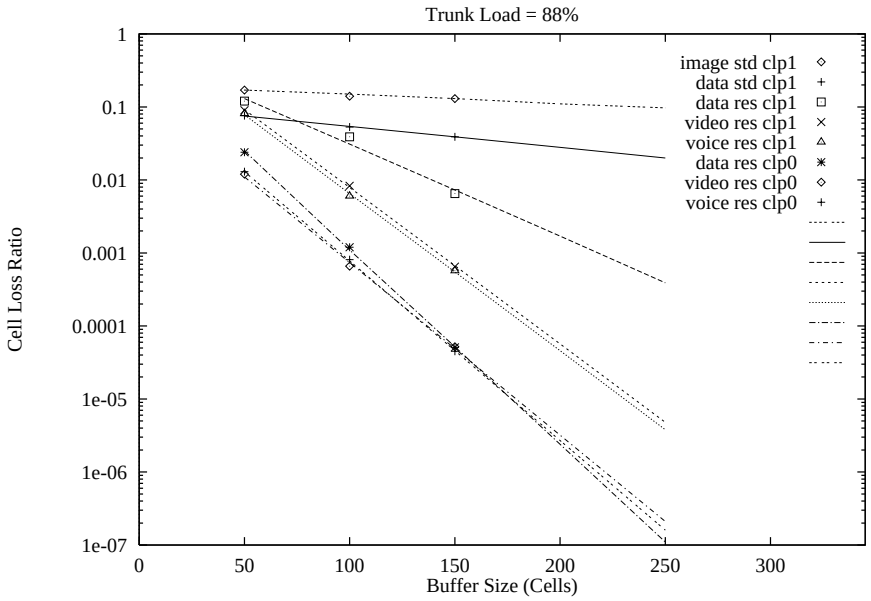


**Figure 5.** CLR vs. Buffer Size showing protection of Queue 1 traffic

entering the $T_1/T_2$ module. If both queues in the $T_1/T_2$ module are receiving traffic and the timer length is small compared to the time required to transmit all the cells in a packet, the delay through the network for that packet will be large. The reason for the large delay is due to the queue server switching between queues 1 and 2. The cells enter, say, queue 2 but do not all get served before the server switches to queue 1. The server again returns to queue 2 and serves some of the cells before switching to queue 1. This process of switching adds a maximum of $T_1$ seconds to the delay of the packet each time the server switches from queue 2 to queue 1. In the network used for simulations (see Figure 4), source packets were segmented into Frame Relay frames or ATM cells before being queued in the $T_1/T_2$ queue. The image sources (one of the four types of sources used) generated large packets (400,000 bits) that would be segmented into many frames or cells and thus overload its queue. The value for $T_2$ was smaller than the time needed to clock out all of the image frames or cells. Figure 6 shows that the delay for the image traffic is 20 times that of the other traffic sources. (End-to-end refers to the delay from source output to destination.) If the timer lengths were increased to counter this delay, the high priority queue would suffer larger delays due to the high granularity of the timer

values. A better solution would be to have a separate queue for the image so that it can be served according to its own traffic characteristics, e.g. increase the service rate of the image for some or all of the image burst.

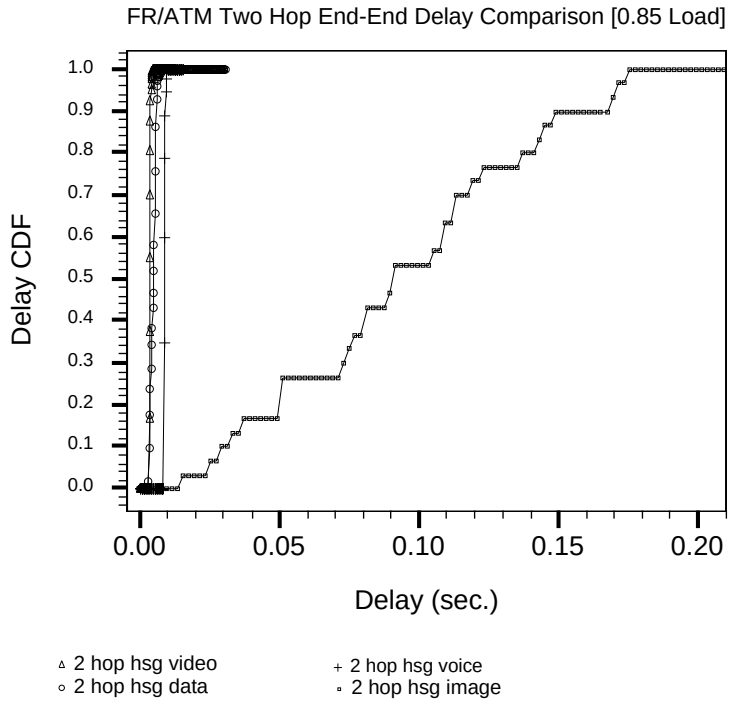


**Figure 6.** End-to-End Delay Comparison

### 2.3 Dynamic Traffic Shaping

The algorithm presented here allows an i-out-of-m shaper to dynamically adjust the values of *i* and *m* in order to match the source traffic characteristics. The inputs to the algorithm are two indirect measurements of the traffic stream: the rate and the average burst size. Before the measurements are described it is important to understand the general operation of the shaper.

In general, a shaper is a module that delays traffic cells so that the burstiness and rate are controlled; in particular, an i-out-of-m shaper is a sliding window that allows the user to transmit, at most, $i$ cells in any $m$ cell slots ($i \leq m$). The i-out-of-m shaper is based on the usage of credits which

are required for a VC to be able to transmit. If a VC runs out of credits, its traffic is queued until a credit is returned. This algorithm can transmit a maximum of *i* ATM cells in any *m* cell slot interval yielding a maximum sustained throughput of *i*/*m* normalized to the line rate. As a VC transmits a cell, the credit bank for that VC is decremented by one and that credit is returned to the credit bank for that VC *m* cell slots later. The goal for adaptive i-out-of-m traffic shaping is to have an algorithm that would be based on two traffic measurements and would be able to dynamically adjust to match the source traffic characteristics. During the early stages of this research, it seemed that it should be possible to make two measurements and then be able to adjust both the *i* and the *m* parameter of the i-out-of-m shaper. Previous work at the University of Kansas, however, had produced an algorithm that dynamically adjusted only *i* while assuming a constant value for the *m* parameter [Bog92]. Starting with that algorithm, a new algorithm was developed, based on different measurements, that would dynamically adjust both *i* and *m*. This algorithm, its development, and performance are presented below.

#### 2.3.1 CPE-Network Interaction for Dynamic Bandwidth Allocation

To understand the motivation for this algorithm it is important to know the possible communication and interaction that can occur between the customer premises equipment (CPE) and the network. The link must cross the UNI (User Network Interface), which is a boundary between a user's equipment and the network provider's equipment. The reason for communicating across the UNI is that the CPE knows what traffic is being sent or will be sent and the network equipment knows what network resources are available. On the CPE side would sit the adaptive i-out-of-m shaper which would monitor customer traffic and on the network side would be a policer such as a Leaky Bucket [UNI] (see Figure 7). This CPE-UNI communication would allow the development of a dynamic allocation scheme in which the shaper would monitor and control the user traffic and the policer would allocate bandwidth in the network. With such a setup there are several modes of communication that could exist between the shaper and the policer.

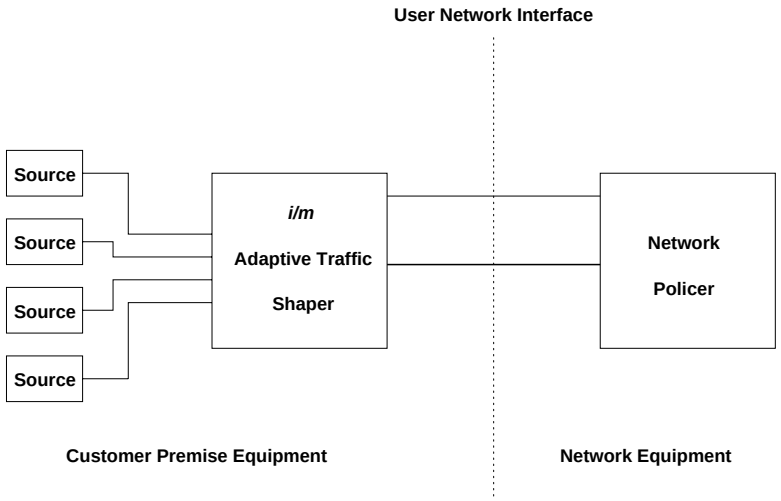


**Figure 7.** CPE Shaper Placement vs. Network Policer Placement

First, the i-out-of-m shaper could simply follow the customer traffic (essentially no shaping) and periodically alert the policer of the traffic parameters (i.e. burst size, average rate) but not wait for a response from the policer. This mode could result in customer traffic being discarded inside the network because it is non-conforming. Since no shaping was done prior to entering the network the traffic is not guaranteed to be conforming. This open-loop operation is more of a data gathering system rather than a traffic shaping system since the shaper tracks the traffic and can output the traffic characteristics.

The second mode of operation is a closed-loop operation in which the shaper tracks the traffic and again periodically notifies the policer of the traffic characteristics, but waits to see if the new characteristics will be accepted by the policer (i.e. the network has enough resources to accept the new characteristics). If the policer accepts the new traffic characteristics the i-out-of-m shaper will shape the traffic to conform to the new characteristics so that the policer will not discard the traffic. On the other hand, if the policer does not accept the new characteristics the shaper will continue to shape the traffic with the old characteristics.

The closed-loop mode of operation allows the possibility of contract renegotiation with the

customer. If it is assumed that the customer must pay for the resources used, the customer would like to know when the "resource package" that was agreed upon when service was set up overestimates the resources that the customer actually needs. The i-out-of-m shaper monitoring the traffic identifies the change (decrease) in the customer traffic, and the network contract with that customer can be adjusted so that the contract more closely agrees with the customer's traffic characteristics. This contract renegotiation saves the customer money and frees up resources for the network provider. Of course, the situation could occur in which the traffic contract underestimates the customer's traffic needs, but again the customer would want to know this so that more resources could be requested.

### 2.3.2 Adaptive i-out-of-m Shaping Algorithm

#### 2.3.2.1 Traffic Measurements

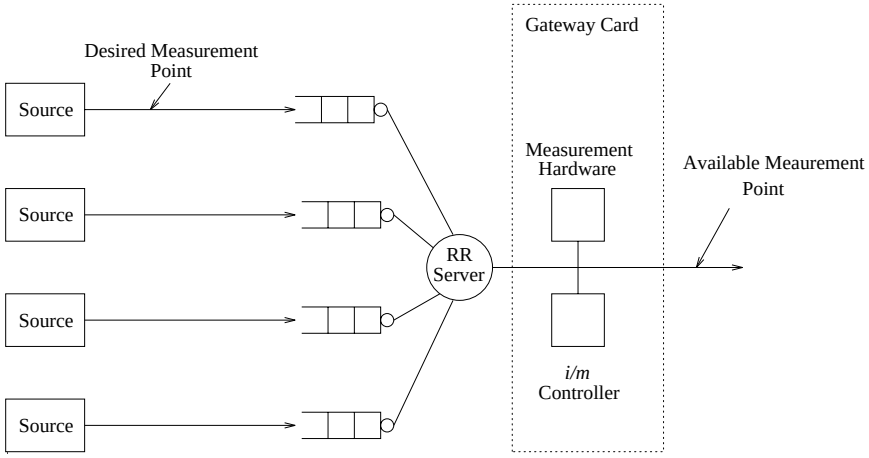


**Figure 8.** Hardware System Layout Showing Measurement Points

Without knowing the layout of the particular network for which this algorithm was designed, it might seem that the solution to adaptive shaping would be to simply monitor the source output and then serve the queue according to the measurements of the source. Indeed that would be a trivial problem, but there are many constraints in this particular system that make it impossible to see that actual source output, so that the measured characteristics are not necessarily the same as the actual source characteristics. In the current hardware configuration (see Figure 8), the sources transmit cells

to per-VC queues which must perform the Digital Equipment Corporation (DEC) AN2 arbitration process before they can be served with a round-robin server, constrained by the i-out-of-m controller. The ATM cells are then transmitted from the Gateway card where the measurement hardware is located. From Figure 8 it can be seen that due to the hardware constraints [TISL] it is impossible to monitor the direct output of the source, but only the output of the rate-controlled round-robin queue. The source output is corrupted further by the arbitration that is performed by the AN2 hardware from DEC that allows a source to send cells at a maximum of every other cell slot. These corruptions to the source traffic stream made it much more challenging to develop two measurements that would give an accurate estimate of the source traffic. Because the algorithm was designed for a system in which neither the source nor the input queues can be seen, the algorithm is general and will work for systems with different configurations.

The first measurement that was tried (and probably the most obvious) was rate. As long as the queue is not constantly building up, the rate of cells out of the queue should be the same as the rate of cells into the queue. Therefore by counting the number of cells that go by during a known time span, it is possible to estimate the rate of the source,

$$\lambda = \frac{Total\ Number\ of\ Cells\ Counted}{Length\ of\ Measurement\ Interval\ (cellslots)}.$$

Of course, this rate is being limited by the ratio $i/m$, so the algorithm (described later) always attempts to keep $i/m$ larger than $\lambda$. The second measurement is designed to give an estimate of the burst size of the source and hence a value for the $i$ parameter. Unfortunately the queue severely corrupts the size of the source bursts so that the average number of cells in a burst entering the queue is not the same as the average number of cells in a burst leaving the queue. The initial idea for a burst measurement was to set a stopped flag when a VC ran out of credits and then accumulate the number of slots during which the VC was stopped. The percent time the VC was stopped could be calculated and then used as a gauge for increasing or decreasing the number of credits, $i$. After simulation analysis it was found that the stopped parameter and stopped time were not well-behaved variables with consistent trends.

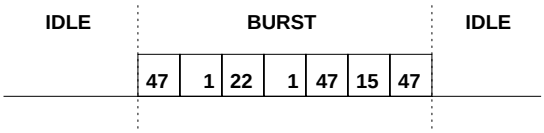


**Figure 9.** Example of a Burst

A new measurement was developed for calculating a burst size for the traffic stream flowing out of the queue. According to the algorithm, a burst occurs when the queue output line changes from being idle to non-idle and then idle again (see Figure 9). If a VC transmits at least one cell during a burst then the number of bursts for that VC is incremented. For example, in Figure 9 the "number of bursts" parameter would be incremented by one for VCs 1, 15, 22, and 47. By also counting the total number of cells transmitted during a measurement interval, it is possible to calculate an average for the burst size,

$$\beta = \frac{Total\ \ Number\ of\ \ Cells\ Counted}{Number\ of\ \ Bursts}.$$

This measurement worked better than the percent stopped measurement, but problems occurred due to the "phasing" of credits. When credits were used, they were used in large clumps, but due to return scheduling they can get returned singly or in smaller groups. If the credits returned in small groups, they could only be used in small groups and so the average burst size as calculated at the output would be artificially low.

Since the problem with average burst measurement was that it was counting too many bursts, the solution would be to somehow concatenate bursts. The solution that was arrived at was to use the stopped flag in combination with the above average burst. If the stopped flag was set for a particular VC when a burst ended, the number of bursts for that VC was not incremented when a new burst began (refer to Appendix A for code). The reasoning is as follows. If the VC had credits when the burst ended it would have been able to continue sending the cells in its queue and the second burst would not have occurred. By not incrementing the number of bursts for that VC the same result has occurred. One potential "problem" with this measurement is that it is possible to concatenate every burst together

in a measurement interval. While this result is not an accurate measure of the average burst size for the source it does not adversely affect the algorithm because the adaptation for *i* is not based on the actual value of the average burst parameter, but is based on the value of the average burst parameter relative to the previous value for *i*.

Another exception to the burst parameter rule is the case of a single active source. Due to the hardware operation, a single source cannot transmit back-to-back cells. It can transmit, at maximum, during every other cell slot. According to the burst count mechanism described above, each single cell would be a burst. Clearly, this is an incorrect burst size if the source transmits more than one cell in a burst. To correct this problem, the VC number for the cell currently being served is compared to that of the last cell transmitted. If they are the same, the burst size for that VC is not incremented. If they are different, the operation is the same as that presented above.

#### 2.3.2.2 Adaptive Shaping Algorithm

Now that the measurement variables have been described, the actual algorithm can be presented (see Figure 10). The values for *i*, *m*, and the measurement interval must be initially set by the user. Once a measurement interval is over, the dynamic control process receives the raw measurements that are taken by the hardware and calculates the rate and average burst size as described above. Rate utilization is calculated as:

$$r = \frac{\lambda}{i/m}.$$

This parameter measures how closely the shaper is estimating the actual rate of the source. At this point, the algorithm begins the adaptation on the burst parameter by comparing the value of *i* to the measured average burst size, $\beta$. If *i* is less than $1.1 * \beta$ then *i* must be increased (the number of credits is too small). Similarly, if *i* is greater than $1.3 * \beta$ then *i* must be decreased (the number of credits is too large). If *i* is between 1.1 and 1.3 times the value of $\beta$ no adaptation is performed on *i*. Once the direction for the adaptation has been determined, the *i* adaptation is:

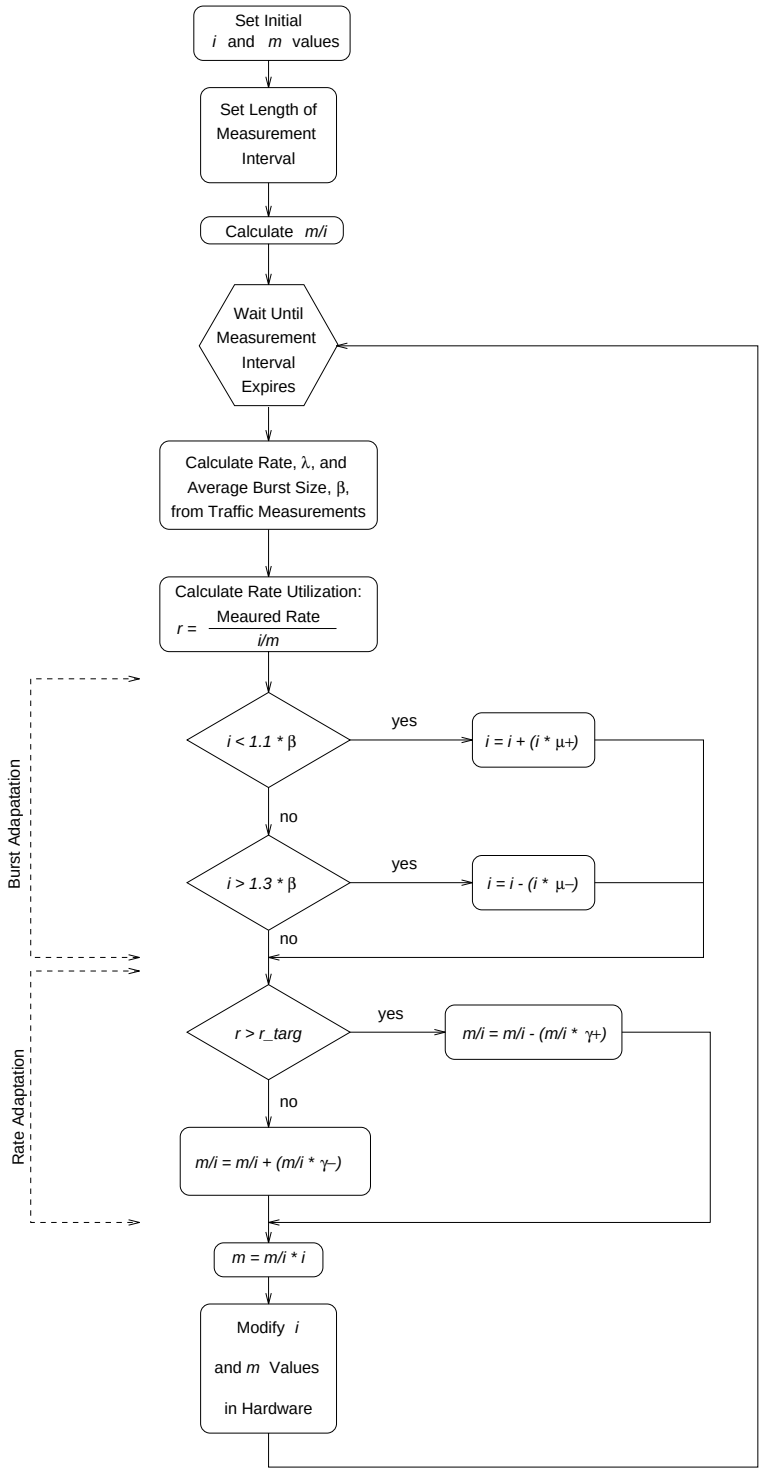


**Figure 10.** Adaptive i-out-of-m shaping algorithm

$$i = i + (i * \mu+)$$

for an increase in the number of credits, or

$$i = i - (i * \mu-)$$

for a decrease in the number of credits. The parameters $\mu+$ and $\mu-$ are adaptation factors. The values of 1.1 and 1.3 for the hysteresis in the $i$ comparison were chosen after performing test simulations.

The decision for the direction of the rate adaptation is simpler because there is no hysteresis. If the rate utilization, $r$, is less than a threshold, $r\_targ$, the rate ($i/m$) is increased and otherwise the rate is decreased. For the simulations presented here, $r\_targ = 0.85$, meaning that the target shaper rate will be 18% greater than the computed average rate, $\lambda$. It should be pointed out that the calculation of $m$ divided by $i$ leads to the inverse of the rate (slots/cell). The reason for this calculation is that the dynamic control process can perform a simple multiplication to arrive at $m$

$$m = m/i * i.$$

Hence, the rate adaptation is performed according to the assignment equations

$$m/i = m/i - (m/i * \gamma+)$$

for an increase in the shaper rate, or

$$m/i = m/i + (m/i * \gamma-)$$

for a decrease in the shaper rate.

#### 2.3.2.3 Simulation Results

The first source used for simulations of the adaptive traffic shaper is a periodic source. The source will output the same size burst, $B$, at integer multiples of the period, $(P + B)$. Since this source is deterministic, the steady-state values of the i-out-of-m shaper are known before the simulations are run. As a specific example, if $B = 10$ and $P = 100$, then it would be expected that the steady-state value for $i$ would be between 11 and 13 (1.1 * 10 and 1.3 * 10) and the steady-state value for $i/m$ would be:

$$\frac{B}{(P+B)} * \frac{1}{r\_targ} = \frac{10}{(100+10)} * 1.18 = 0.1070.$$

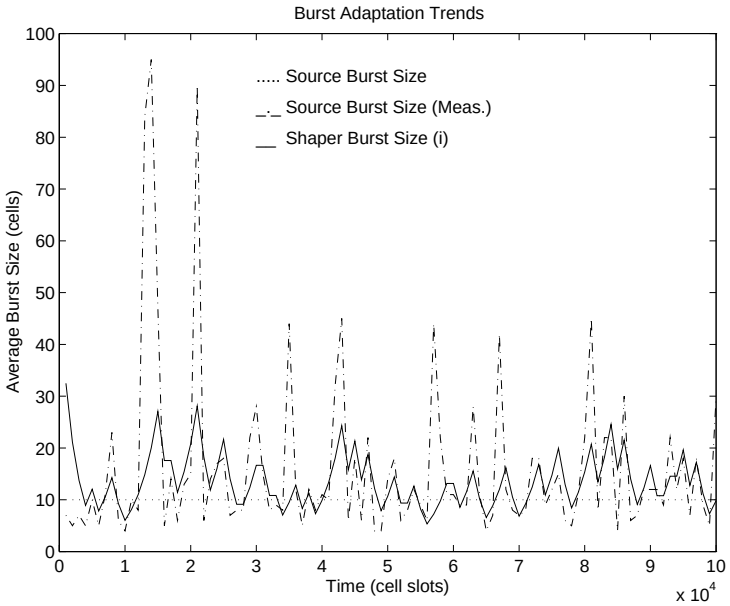


**Figure 11.** Adaptive shaping algorithm tracking burst size of periodic source

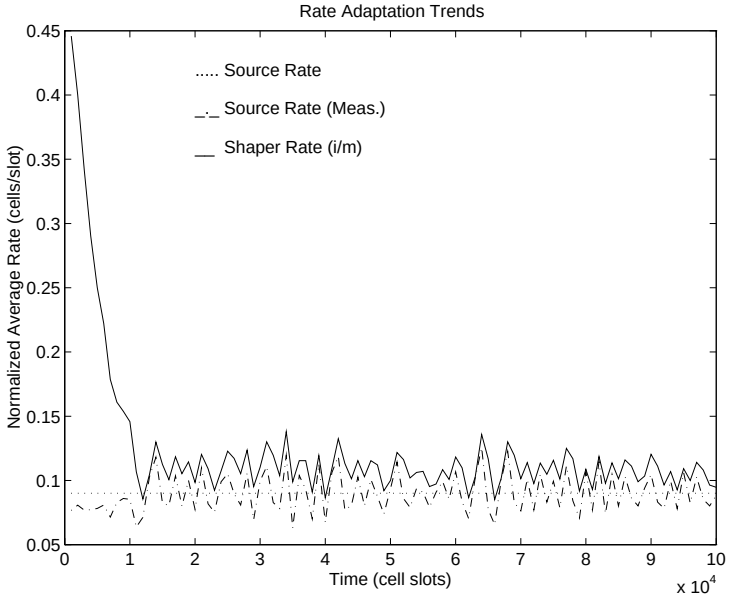


**Figure 12.** Adaptive shaping algorithm tracking rate of periodic source

The simulations used one periodic source and three exponential sources (described below). Figures 11 and 12 show the performance of the shaper for tracking the burst and rate, respectively, for the periodic

source. The initial settings for *i* and *m* were 50 and 100, respectively. As can be seen from the figures, the value of *i* falls from the initial value and tracks the average burst size of the source, *B*. The rate also drops and follows the average source rate. The average value of *i* for the simulation was 12.57 cells, and the average shaper rate, *i*/*m*, was 0.1069. These numbers agree with the expected steady-state values above.

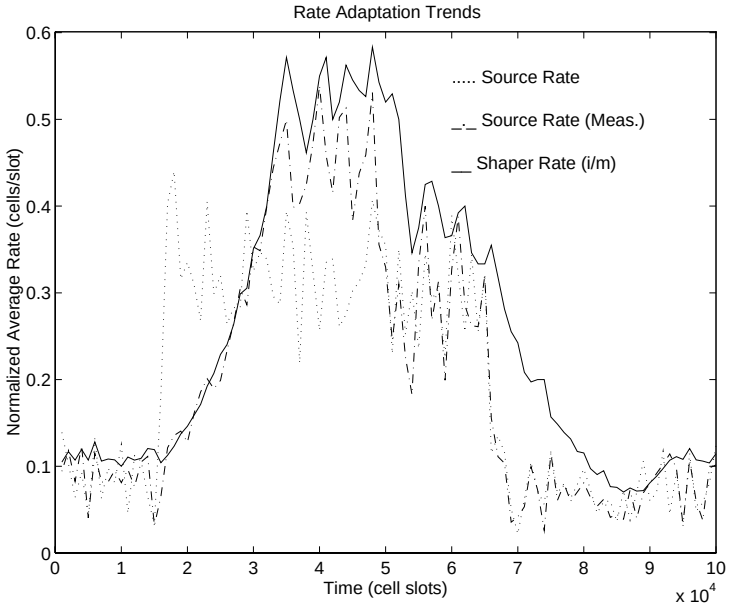


**Figure 13.** Adaptive algorithm tracking a change in the source rate

While the periodic source tests that the shaper works properly, it does not "challenge" the algorithm's performance. The most obvious method to test an adaptive algorithm of any kind is to use a step function or a pulse as the input to the system. In the case of the dynamic i-out-of-m algorithm there are two adaptive parameters that must be tested, *i* and *m*. Since it would be most informative to test the two adaptations separately, the traffic source parameters were adjusted so that only one of the parameters was tested during a single simulation. These simulations used four independent exponential sources. The on/off exponential sources generate exponentially distributed "on" times (packet sizes) and exponentially distributed "off" times (interarrival times). Figure 13 shows the tracking ability of the algorithm for a pulse in the source rate (with $\gamma + = \gamma - = 0.08$). The rate pulse

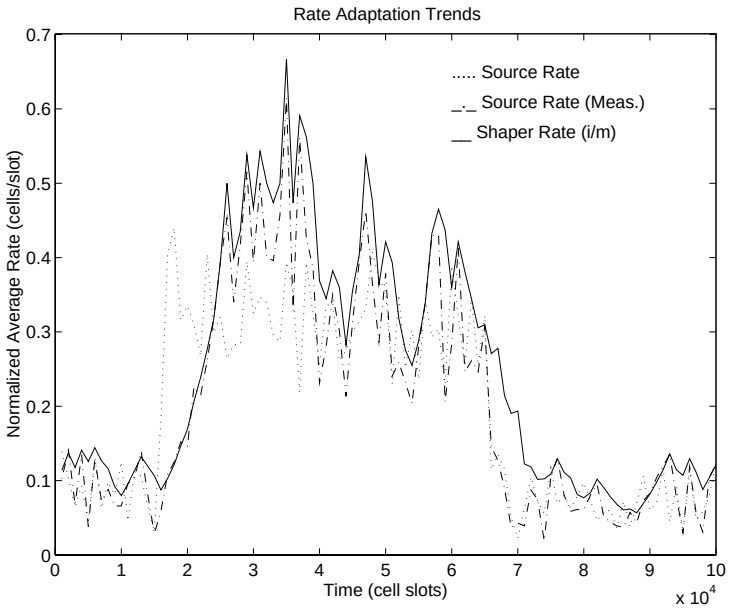


**Figure 14.** Adaptive algorithm tracking a change in the source rate

occurs between 15,000 and 65,000 slot times during which time the normalized average source rate is increased from 0.1 to 0.5. The source rate returns to 0.1 at 65,000 slot times. It is obvious that the algorithm reacts to the source rate pulse, but it is also evident that the reaction time is slow resulting in queue build up and hence delay and possible cell loss. This delay problem can be addressed by selecting adaptation constants, $\gamma +$ and $\gamma -$, for the rate that will result in a faster attack. However, due to the abrupt changes in the source there will still be a delay spike because the algorithm cannot adapt to the changes instantly. Figure 14 shows the rate adaptation for $\gamma + = \gamma - = 0.15$. The rate attack is quicker than that of Figure 13, and the average delay has decreased from 2966 slots/cell to 781 slots/cell. Again only one of the parameters was tested at a time and Figure 15 shows that the source burst size was kept relatively constant during the rate increase. There is some noise in the response for the burst parameter, $i$, for the first 3000 slots, during which time the algorithm is attempting to adapt both the rate and burst parameters. As the input stimulus (the rate bump) persists, the algorithm eventually adapts only the rate parameter.

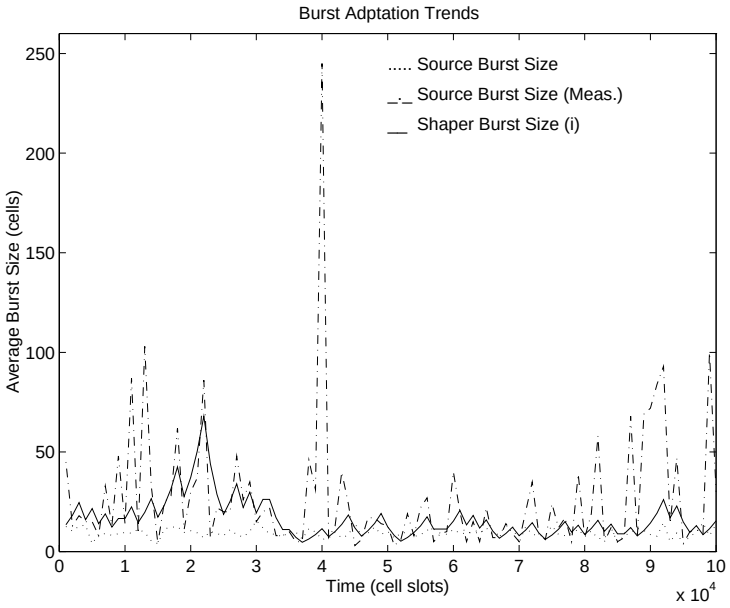


**Figure 15.** Adaptive algorithm smoothing noise in the source burst size

Figure 16 shows a plot of the average delay versus the rate adaptation factor. (For this plot $\gamma + = \gamma -$.) As the rate is allowed to adapt quicker (larger $\gamma$) the average delay decreases until $\gamma$ = .30. At this point the average delay increases and then decreases as $\gamma$ is increased further. The increase in average delay is due to the fact that the adaptation is too quick and there is a lot of overshoot and undershoot. It should be noted that if $\gamma + \neq \gamma -$ i.e., $\gamma + > \gamma -$, further improvements in the rate adaptation may be possible.

Figure 17 shows the ability of the algorithm to track a change in the burst size of the source, with the source rate held constant. The source burst size is increased from an average of 10 cells to 50 cells for 50,000 slot times (15,000 - 65,000). While the increase in the source rate was clear in Figure 13, the adaptation of the average burst measurement $\lambda$ and the average burst parameter *i* are not that clear from Figure 17. Table 6 shows the average source burst size and the average *i* for the periods before, during, and after the burst bump. These numbers show that the source rate increases by a factor of five (as intended) and that the average burst parameter *i* does follow the source burst size, but not as closely as the rate adaptation shown in Figure 13. This behavior for *i* is not unexpected since rate is simply the

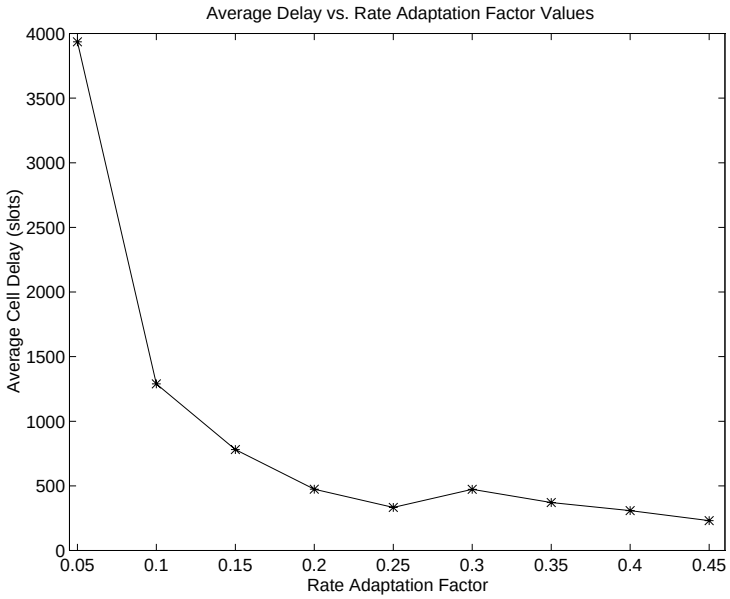


**Figure 16.** Average Cell Delay as a Function of Adaptation Factor

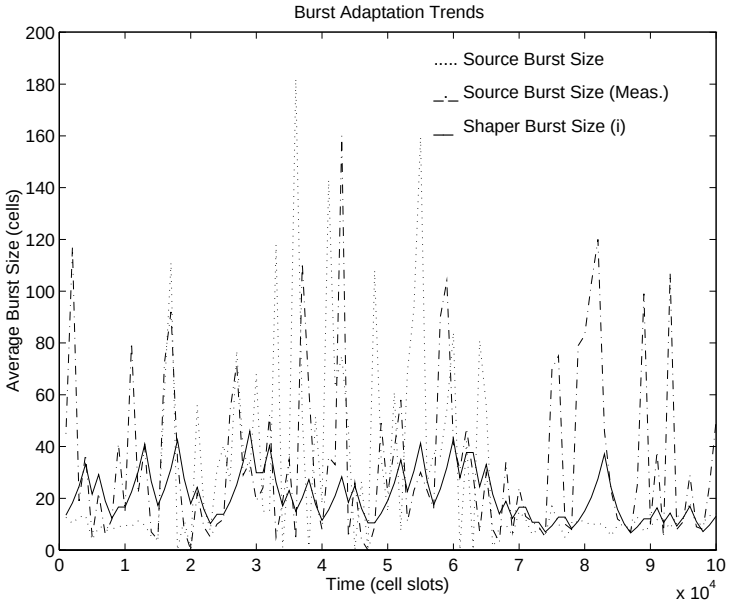


**Figure 17.** Adaptive algorithm tracking a change in the source burst size

number of cells seen in a time interval, but it is harder to develop an accurate measure for the average burst size when observing only the shaped traffic stream. The rate of attack for *i* can be changed by

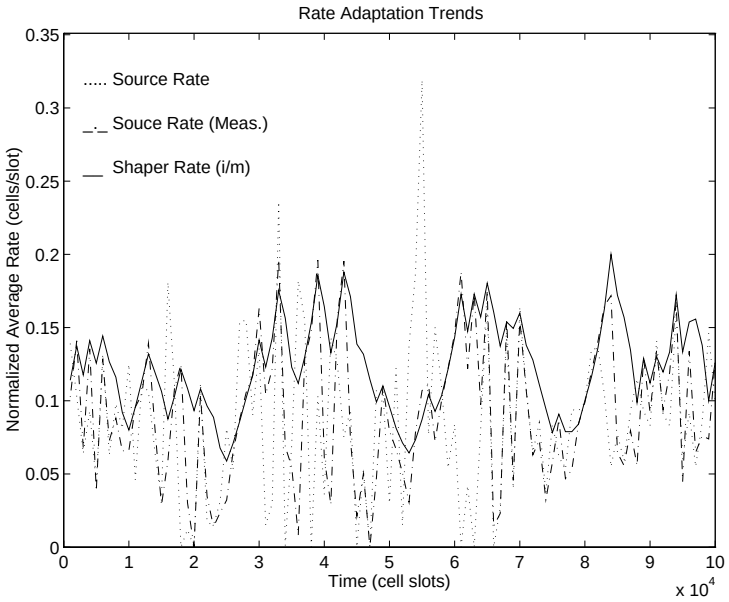


**Figure 18.** Adaptive algorithm smoothing noise in source rate

changing the adaptation factors, $\mu+$ and $\mu-$. The adaptation shown here is for $\mu+ = \mu- = 0.35$ so a value of 0.5 or 0.6 would shorten the time in which *i* reacts. It is also possible to set the adaptation factors so that *i* rises faster than it falls, i.e. $\mu+ > \mu-$. Since there was no change in the source rate, there should be no change in the shaper rate, $i/m$, (see Figure 18). There is noise in the source rate, but no appreciable change, and both the shaper rate and the measured rate track the source rate.

**TABLE 6.** Comparison of Average Source Burst Size and Average Burst Parameter

| | Time in Cell Slots | | |
|---|---|---|---|
| | 0 - 15000 | 15000 - 65000 | 65000 - 100000 |
| Average Source Burst | 9.13 | 47.49 | 11.22 |
| Average *i* | 22.82 | 24.68 | 14.63 |

The previous results have shown that the algorithm can adapt to changes in the source traffic characteristics, but there needs to be a benefit for the increase in hardware. Table 7 shows why the algorithm is beneficial and why it makes sense to increase the size and complexity of the transmitting system. The table compares the long-term average cell delay for the adaptive i-out-of-m algorithm

**TABLE 7.** Delay Comparison for Adaptive and Non-Adaptive i-out-of-m Shapers

| | Fixed Traffic | | | | |
|---|---|---|---|---|---|
| | Dynamic i/m | Fixed i/m | | | |
| | Average $i$ = 20.76 | 10 | 20 | 30 | 40 |
| Delay (slots) | 579 | 904 | 874 | 844 | 816 |

with static i-out-of-m shapers that have various values of $i$, but the same $i/m$ ratio. The source behavior for all these simulations is the same as for the burst bump described above. The adaptive i-out-of-m shaper can achieve lower delays than the static i-out-of-m shaper even for static $i$ values that are larger than the average $i$ for the adaptive case. By increasing the value of $i$, in the static case, it would be possible to achieve delays that are lower than those for the adaptive shaper (these larger static $i$ results are not shown). However, there is an assumed cost for the burst size and in the case of the "burst bump" (changing source burst size while keeping the rate constant) presented here, the larger burst size will not be fully utilized when the source burst size is smaller than the static burst size value. The user would be paying for the burst size and not using it fully, whereas the adaptive algorithm would use a larger burst size only where the customer needs it and therefore would save the customer money and would save resources for the network provider.

The above simulations were all performed using source models. A much better test of the adaptive algorithm would involve testing the adaptive abilities with actual traffic. The actual traffic used in the following simulations is the same Bellcore ethernet trace data that was used in section 2.1.2.2. As was already stated the trace data is very different from any models that are currently being used. Only one of the four exponential sources was replaced with a trace data source so that the performance of the algorithm for the trace data could be isolated. Simulations were run with different values for the adaptation factors, $\mu+$,$\mu-$, $\gamma+$, and $\gamma-$, in order to find those that give the lowest delays. The following results were run with these "optimal" parameter settings ($\mu+ = \mu- = 0.35$, $\gamma+ = \gamma- = 0.15$). Figures 19 and 20 show that the algorithm can follow the trends of the trace data source. These graphs also show that the trace data is very different from the exponential models that were shown earlier. The

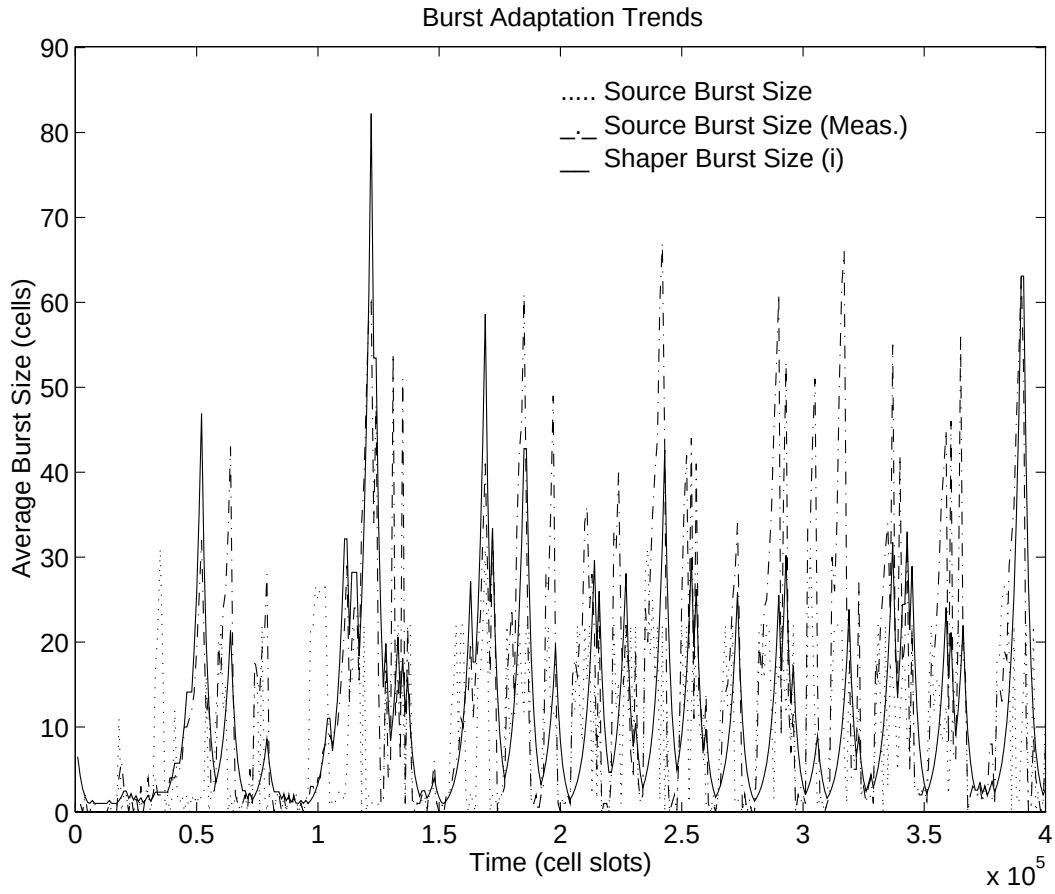


**Figure 19.** Adaptive algorithm tracking average burst size of Ethernet trace data

exponential sources were bursty during the "rate bump", but did not show an order of magnitude variability in the rate when the rate bump was off. The trace data, on the other hand, was not altered with a rate bump and did show order of magnitude variability in the rate and burst. These graphs, however, are not as informative as Tables 8 and 9 which compare the delay for the adaptive shaper and static shapers with different values for $i$ and the ratio $i/m$. Because the source was not constrained as were the exponential sources it was necessary to run static shapers with different rates and different burst sizes. Table 9, therefore, is a matrix with the center numbers being the average longterm delays. Tables 8 and 9 suggest that a static shaper is better than an adaptive shaper for the trace data source. At first it was thought that a longer simulation would capture more of the bursty characteristic in the

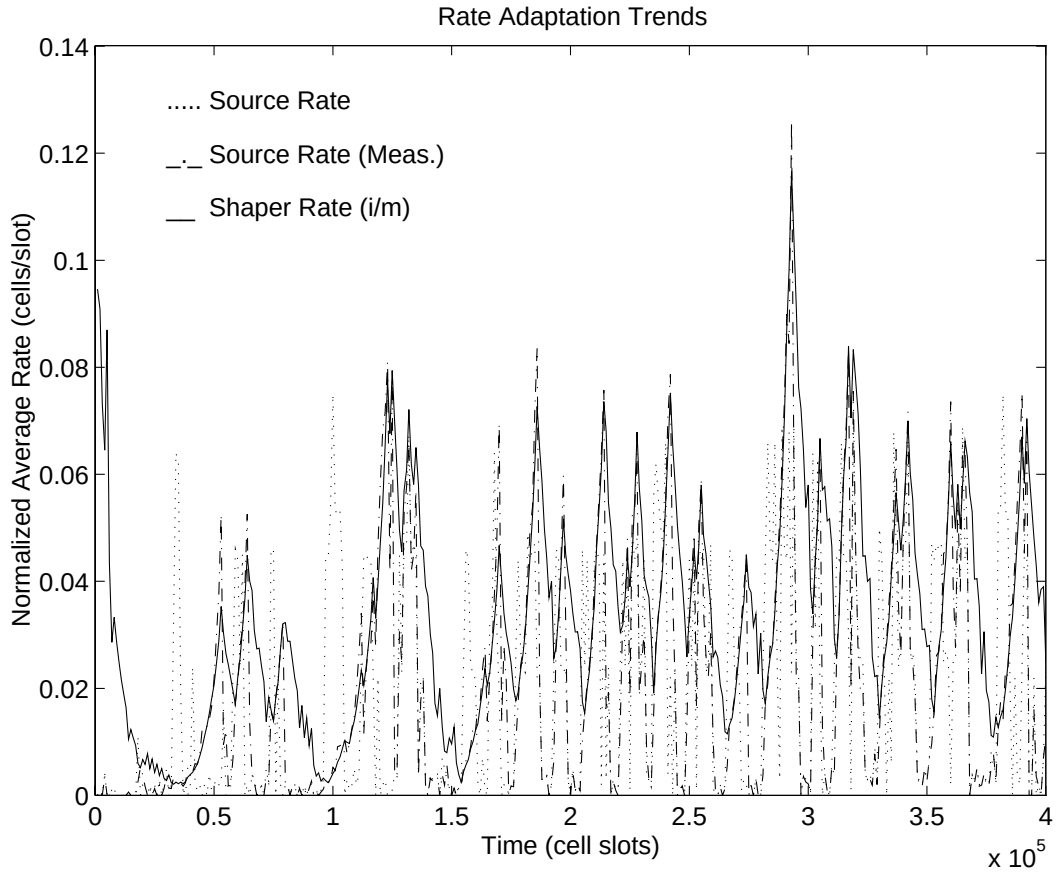


**Figure 20.** Adaptive algorithm tracking average rate of Ethernet trace data

**TABLE 8.** Delay for Adaptive i-out-of-m Shapers with Trace Data

| Fixed Traffic ($\mu$ = 10.95, $\rho$ = 0.0205) | |
|---|---|
| Average $i/m$ Ratio | Average $i$ Value |
| | 11.57 |
| 0.034 | 1,870 |

trace data. Again, however, the results show that an adaptive shaper produces much larger delays than a static shaper. Figure 21 shows the average rate of the trace data file averaged over 6.5 second windows. Even the long simulations (25 million cell slots) read only 68 seconds of the trace data file (shaded area). The only changes that are captured in this small slice of the trace file are very short

**TABLE 9.** Delay for Non-Adaptive i-out-of-m Shapers with Trace Data

| Fixed Traffic ($\mu$ = 10.95, $\rho$ = 0.0205) | | | | |
|---|---|---|---|---|
| $i/m$ Ratio | $i$ Value | | | |
| | 5 | 10 | 15 | 20 |
| 0.020 | 34,650 | 34,513 | 34,413 | 24,498 |
| 0.025 | 6,950 | 6,873 | 6,815 | 6,770 |
| 0.030 | 2,948 | 2,934 | 2,852 | 2,848 |
| 0.035 | 1,243 | 1,225 | 1,193 | 1,160 |

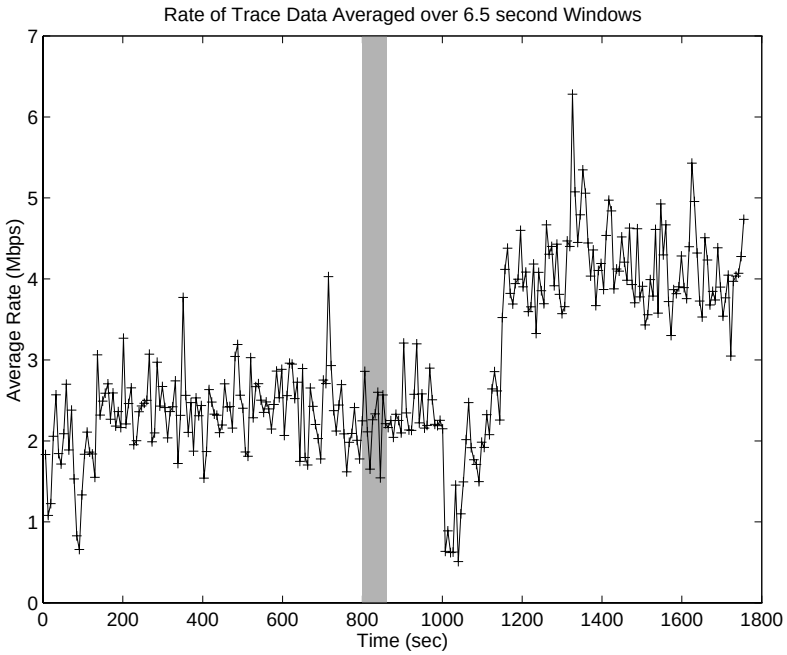


**Figure 21.** Rate of Trace Data Showing Amount Simulated

term. The long term change that occurs at approximately 1000 seconds (see Figure 21) is on the order of 100 seconds. Since the long simulation took 24 hours to run, a simulation that would capture the long term changes would run for several days. It was not possible to run simulations of this length. Even though the trace data simulations did not reflect the expected results, they did show that the shaper parameter settings (adaptation factors) will be different for different traffic types. For the trace data, the adaptation would need to be slow enough to smooth out the high variance of the source, but quick enough to catch the sustained changes that could occur. Because there were no sustained

changes in the section of the trace data that was used for the above simulations, the static shaper gave better performance because it smoothed out the variance of the source whereas the adaptive shaper tried to follow the short term variance causing the adaptive shaper to overshoot (and undershoot) the actual source rate.

# Chapter 3

# Conclusion

### 3. Conclusion

#### 3.1 Summary

This technical report has presented three methods for bandwidth management. Section 2.1 presented a performance evaluation of Equivalent Capacity, a static call admission control scheme developed in [IBM91]. The admission control scheme was developed with exponential sources and proved to be inconsistent when tested with bursty sources. Real ethernet trace data from Bellcore resulted in under-estimations while image traffic resulted in over-estimations. The next section showed the results of performance tests involving the static dual queue server [Srir90]. The $T_1/T_2$ queue server was used in ATM network simulations to test how well the queue server can guarantee bandwidth and shield traffic. Due to the fact that the server is used with a dual queue, problems resulted when different traffic classes were combined in a single queue. The two classes require different service parameters, but can only be served by one. The result is that one (or both) of the classes can suffer larger delays or losses than if the classes were served with individual servers tailored to the service requirements of that class. The last section of Chapter 2 presented the development of a dynamic bandwidth allocation scheme. The dynamic bandwidth allocation scheme centers around the development of an adaptive traffic shaping scheme based on an i-out-of-m traffic shaper. From two traffic measurements it was shown that this adaptive shaper can estimate the source traffic characteristics. Different sources were used with the adaptive shaper to test the adaptation algorithm. The shaper was able to track periodic sources, exponential sources, exponential sources with artificially increased rates and burst sizes, and real trace ethernet data. These simulations also showed that an adaptive shaper can achieve lower delays than fixed shapers with the same parameters when

there are sustained changes in the source rate or burst size.

### 3.2 Conclusions

By testing the equivalent capacity call admission control scheme with different sources it became apparent that the model on which an algorithm is based is very important. The equivalent capacity scheme was based on an exponential on-off source and did not produce acceptable results when tested with sources that differed greatly from the exponential source. These simulations also showed that the current models may not be very good approximations of real traffic. The classic model for data traffic is an exponential source. This source, however, does not produce traffic characteristics that are seen in actual ethernet data.

As a result of the evaluation of the static $T_1/T_2$ queue server, the effects of the traffic grouping became apparent. We conclude that when deciding how to group traffic it is important to group only traffic with similar traffic characteristics. The service given to a queue can only have one set of characteristics. If different types of traffic are held in a single queue, then the service might only match the characteristics of one of the traffic types. The other traffic type might suffer loss or delay because it is not receiving the "right" service.

The adaptive shaping algorithm can follow user traffic, and therefore can be more cost effective than a static shaper because the adaptive shaper will have a smaller average rate and burst size than a static shaper that can produce average delays as low as the adaptive shaper. The adaptation factors for the adaptive shaper, however, must be tuned for different traffic types.

### 3.3 Future Work

The problem that became clear during the evaluation of equivalent capacity was that the three parameters, burst length, average rate, and peak rate do not uniquely describe a source. There needs to be a method to calculate the above parameters (particularly burst length) in reference to an exponential source (since equivalent capacity is based on exponential sources). The method could map the

parameters ($R$ = Peak rate, $m$ = average rate, and $b_1$ = burst length) of a source with a non-exponential distribution to an exponential source with parameters ($R$, $m$, and $b_2$) such that they have the same effect on network modules [Gun93]. The improvement made to equivalent capacity must identify sources such as the trace data and the image so that the resources can be allocated properly. A network provider would not want to continually lose customer's data nor allocate bandwidth that is not being used.

The investigation of the $T_1/T_2$ queue server showed that the server can guarantee bandwidth between the queues, but some problems were also identified. The dual queue can divide the traffic into two groups (Queue 1 and Queue 2), but if there are more than two groups of traffic (classes), then some of these classes must share a queue. This queue sharing can result in one class being detrimentally affected by the other class with which it shares a queue. A generalization of the $T_1/T_2$ queue server would help this problem by creating per class queueing.

If the per-class queueing is used, then it would be possible to further increase the queue's performance by changing the server from a static bandwidth manager to a dynamic bandwidth manager. The queue could monitor the incoming traffic and determine how to guarantee the bandwidth such that the delay through the queue is minimized for each class.

Section 2.3 presented the development and performance for an adaptive i-out-of-m shaper. Also presented was a framework for a dynamic bandwidth allocation scheme involving the adaptive shaper on the customer side and a policer on the network side of the UNI. This framework should be developed further to determine how the policer should communicate and how often the parameters for the policer and shaper should be changed. Since the policer would be adaptive, there needs to be an algorithm that would control the adaptation of the policer.

A general enhancement to the adaptive shaper would be to add probes that monitor the queue and send a signal when the queue reaches a threshold. The algorithm could then increase the $i/m$ ratio or increase only $i$ (keeping $i/m$ the same) in order to drain the queue quicker. These queue probes would

help for sources that have a small longterm average rate, but can burst large amounts of cells into the round-robin queue.

For the adaptive i-out-of-m shaper it would also be beneficial to study the effects of different lengths of measurement intervals. The simulations presented in Chapter 2 had measurement intervals of 1000 cell slots. If the interval were increased to 10,000 cell slots the shaper would not react as quickly , but this may not be bad for certain load, types of traffic, delay, etc.... The allowable range of the measurement interval lengths would need to be determined by the processor that controls the measurement and updating hardware.

**4. References**

[Mezg94] K. Mezger, D. Petr, T. Kelley, "Weighted Fair Queuing vs. Weighted Round Robin: A Comparative Analysis" *The IEEE Wichita Conference on Communications, Networking, and Signal Processing,* April 1994.

[Srir90] K. Sriram, " Dynamic Bandwidth Allocation and Congestion Control Schemes for Voice and Data Multiplexing in Wideband Packet Technology", *IEEE International Conference on Communications (ICC),* April 1990, pp. 1003-1009.

[IBM91a] R. Guérin, H. Ahmadi, and M. Naghshineh, "Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks", *IEEE Journal on Selected Areas in Communications,* vol. 9, no. 7, pp 968-981, 1991.

[Lel94]W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)", to appear in IEEE Transactions on Networking 1994.

[Lu94] Y. Lu, D. Petr, T. Kelley, "Long Range Dependence in Telecommunication Traffic", *IEEE Wichita Conference on Communications, Networks, and Signal* Processing 1994.

[Gun93] L. Gün, "An Approximation Method for Capturing Complex Behavior in High Speed Networks", *Performance Evaluation: Special Issue on Bandwidth Management and Congestion Control in High Speed Networks,* 1993.

[BONeS] K. S. Shanmugan, V. S. Frost, W. W. LaRue, "A Block Oriented network Simulator (BONeS)", *Simulation,* Vol. 58, No. 2, pages 83-94, 1992.

[UNI] ATM Forum, *ATM User-Network Interface Specification, Version 3.0,* ATM Forum, June 1993.

[TISL] S. Seetharam, H. Uriona, "Description of the i-out-of-m controller - Design 1 implemented in Xilinx 3195 FPGA", *TISL Technical Report TISL-9770-26,* Telecommunication and Information Sciences Laboratory, University of Kansas, July 1994.

[Bog92] M. Bog, "Credit Statistics Measurement for Dynamic Bandwidth Management", *TISL Technical Memo TISL-9770-08,* Telecommunication and Information Sciences Laboratory, University of Kansas, December 1992.

[EBNTa] D. Petr, V. Frost, L. Neir, A. Demirtjis, C. Braun, "Evaluation of Broadband Networking Technologies: Phase I Report", *TISL Technical Report TISL-9750-1,* Telecommunication and Information Sciences Laboratory, University of Kansas, January 1993.

[EBNTb] D. Petr, V. Frost, A. Demirtjis, C. Braun, "Evaluation of Broadband Networking Technologies: Phase II Report", *TISL Technical Report TISL-9750-4,* Telecommunication and Information Sciences Laboratory, University of Kansas, July 1993.

CONTENTS

LIST OF FIGURES

LIST OF TABLES