To appear in *Computer Networks and ISDN Systems*

Scheduling and Admission Control for Integrated-Services Networks: The Priority Token Bank*

Jon M. Peha, Associate Professor, Carnegie Mellon University
peha@ece.cmu.edu, http://www.ece.cmu.edu/afs/ece/usr/peha/peha.html

Abstract

This paper proposes a new mechanism called the *Priority Token Bank* for admission control, scheduling, and policing in integrated-services networks. In such networks, both arrival processes and performance objectives can vary greatly from one packet stream to another. There are two principal components to the Priority Token Bank: accepting or rejecting requests to admit entire packet streams, where acceptance means guaranteeing that the packet stream's performance objectives will be met, and scheduling the transmission of packets such that performance objectives are met, even under heavy loads. To the extent possible, the performance of traffic is also optimized beyond the requirements. The performance achieved with the Priority Token Bank is compared to that of other typical algorithms. It is shown that, when operating under the constraint that the performance objectives of applications such as packet voice, video, and bulk data transfer must be met in an ATM network, the mean delay experienced by other traffic is much better with the Priority Token Bank. Furthermore, the admission control algorithm can guarantee requirements will be met, and admit more traffic than the common alternatives.

Keywords: Admission Control, ATM, Integrated-Services Network, Scheduling, Quality of Service

1 Introduction

As high-speed packet-switched networks become more practical, it becomes feasible to efficiently support more diverse applications on a single *integrated-services* network. Mechanisms are needed in such a network to guarantee that the packets associated with each application will experience adequate performance, where the performance that is considered to be adequate varies greatly from one application to another. Thus, unless network utilization is to be kept so low that all application performance objectives will always be met, network algorithms must be supported that differentiate traffic based on performance objectives and adjust delay distributions accordingly, and prevent network load from becoming so large that performance requirements cannot be met.

There are several ways in which application performance objectives may differ, and network service (and pricing [1, 2]) should reflect these differences. First, the most appropriate measure of performance may differ. For example, performance is often measured in mean delay. This, however,

*This material is based in part upon work supported under Grant NCR-9210626 from the National Science Foundation and under a grant from Bellcore.

is inappropriate for applications such as packet voice and video, where packets are buffered at the destination and played out some fixed delay after they were generated at the source. Performance for such applications is better measured in loss rate, which is the fraction of packets not received within some fixed maximum delay, making playback impossible. Second, even when two applications use the same performance measure, objectives can differ tremendously quantitatively. For example, file transfer can typically tolerate a much greater mean delay than interprocess communication. Finally, applications differ in the consequences of failure to meet performance objectives, and therefore what should be done when objectives cannot be met. For example, if it is not possible to guarantee adequate performance throughout the duration of a telephone conversation, the network should block that telephone call before it can begin. In contrast, if it is not possible to guarantee that a file transfer will complete in 1 second, but a 10 second limit can be guaranteed, it is often appropriate to guarantee the 10-second bound, and *try* to achieve better.

Traffic for which performance has been guaranteed in advance is *guaranteed* traffic. It is not possible to make guarantees unless the network can prevent the arrival rate of guaranteed traffic to the network from growing arbitrarily large. Consequently, an application must first make a call request to the network that includes characteristics of the arrival processes such as average arrival rate and burstiness, as well as the desired performance to be achieved. An *admission control* algorithm either *admits* the corresponding packet stream, thereby guaranteeing objectives will be met, or *blocks* the packet stream. For some admitted streams, performance that exceeds the minimum guarantees is highly desirable; for others, like video, exceeding the guarantee is of no benefit. Since the performance guarantee is dependent on the stated arrival process characteristics, some *policing* mechanism is required to prevent an application from generating more data than was declared in advance, thereby degrading performance for other packet streams.

For applications like voice, for which it is preferable that a call be blocked rather than it be accepted and then experience excessive delays, admission control is essential. However, if packet transmissions are sporadic, as might occur with a link between local area networks, it may be inefficient to get an a priori guarantee, and thereby reduce the capacity available to future guaranteed streams. Still, low delay is important for some inter-LAN traffic. In addition, some packets are too urgent to tolerate the delay associated with the admission control process (which is at least one round-trip delay). Thus, many packets will be transmitted *best-effort*, i.e. without any a priori guarantee. Best-effort traffic could be sent over a datagram service, or with a connection-oriented service such as Available Bit Rate (ABR). Performance objectives can vary from one best-effort packet to another, and the network should meet the performance objectives of best-effort traffic to the extent possible. Indeed, some best-effort traffic may be extremely important to the application.

Once the source-destination path has been determined, the only delay that can be influenced by network algorithms is the queueing delay at the network access point and in switches along the path. Queueing delay is controlled through the *scheduling algorithms* operating in each of these queues, where a scheduling algorithm is the algorithm that orders the transmission of queued packets. Its goal is (to the extent possible) to achieve performance as good as, or better than, the performance objectives for each packet stream. This is best achieved if guaranteed packets are given priority when necessary to meet requirements, and at other times, the more important or urgent queued packets should be transmitted, whether they are best effort or guaranteed. Whenever a source decides to transmit a message, that message is, in effect, queued at the network access point awaiting entry into the network. (This queue is sometimes considered to be part of the source(s) and sometimes part of the network, but as long as the propagation delay from where the data resides to the network access point is insignificant, it is still effectively a single queue.) For example, in an enterprise network, this

queue may form at the interface between the public common carrier and the private local network, a likely scenario in the near term. Since the arrival rate of packets into the queue at the network access point at any given time can greatly exceed the rate at which packets are allowed to enter the network, queueing delay at the network access point is likely to be especially great, possibly dominating queueing delay within the network. Buffer sizes and therefore worst-case mean delays at the switches are also more limited. Consequently, the effectiveness of the scheduling algorithm operating there is likely to be especially critical. For this reason, in this paper, we concentrate on the effects of scheduling in a single queue at the access point rather than a network of queues. Moreover, many additional issues must be addressed to demonstrate effective operation in a network of queues, so we must postpone much of that discussion to a separate paper [3]. We also focus discussion in this paper on networks with constant-length packets, such as ATM networks. (The mechanisms can be extended to variable-length packets, but this is also outside the scope of this paper.)

We will present a novel set of algorithms that provide the functions described above: admission control, scheduling, and policing. In the following section, relevant previous research will be discussed. Section 3 describes our approach: the *Priority Token Bank*. We also show how the Priority Token Bank can be applied to achieve some common sets of performance objectives. The resulting performance is explored in Section 4. Finally, the paper is concluded in Section 5.

2 Related Work

The simplest scheduling algorithms that can protect guaranteed traffic from best effort traffic to enable performance guarantees are *static priority* and *earliest deadline first*. With static priority, packets are assigned a priority from a finite range before entering the network. The next queued packet to be transmitted is the one with the greatest priority, and packets with equal priorities are typically transmitted first-come-first-served. The static priority algorithm is generally not well suited for packets with deadlines such as voice and video packets. In contrast, with earliest deadline first, the next queued packet to be transmitted is the one with the earliest deadline. If objectives are all based on loss rate, late packets should be dropped. This algorithm is far more effective than static priority when performance objectives are based on tardiness, or on loss rate [4, 5], provided that all packets have objectives based only on loss rate/tardiness, and the loss/tardiness of one packet is no more or less desirable than the loss/tardiness of any other packet. However, both of these common algorithms have significant liabilities when diverse traffic must be supported. Packets with loss rate objectives must be given some form of priority over their counterparts without deadlines, so that deadlines will be met during periods of congestion, causing the performance of packets without deadlines to be significantly and unnecessarily degraded when the network is not heavily congested, which is most of the time [6, 7]. In addition, with these approaches, best-effort traffic must always be served last (through low priorities or infinite deadlines), even when a best effort packet is the most important and the most urgent of the queued packets. We will now present a series of more sophisticated approaches, most of which will be evaluated in Section 4.

Cost-Based Scheduling (CBS) [8]: A packet's priority depends on both its performance objectives and the queueing delay it has experienced so far. A cost function $c_i(\tau)$ is associated with each packet class which defines a cost incurred as a function of the queueing delay τ that a class i packet experiences. These cost functions are selected to best represent application performance objectives, so that the total cost incurred reflects the extent to which all performance objectives are being met.

In a single queue, the priority $p_i(t)$ of a class i packet at time t is

$$p_i(t) = \int_0^\infty \frac{1}{\alpha} \cdot e^{-\tau/\alpha} \cdot c_i(q_t + \tau) d\tau \ - \ c_i(q_t)$$

where q_t is the queuing delay the packet has already experienced, and α is a constant. For example, with objectives based on loss rate, priority increases exponentially until the deadline, when priority falls to 0. For objectives based on mean delay, priority is static. CBS has been shown to be highly effective in supporting traffic with heterogeneous performance objectives [8], but it has two disadvantages. First, implementation is somewhat complicated, and therefore may be expensive [9]. Second, because scheduling is entirely independent from the admission control algorithm, the information describing packet arrival processes that is stated when packet streams are admitted cannot easily be exploited. This paper presents an approach in which admission control and scheduling are integrated, so that this information can be exploited, and comparable performance can be achieved with a simpler algorithm.

Occupancy-based scheduling: Packets are divided into classes. The priority of transmitting a class i packet is a function of the number of class j packets queued for all j [10, 11, 12]. For example, there are voice packets with deadlines and data packets for which mean delay should be minimized. Voice packets could have a higher priority when the number of queued voice packets exceeds a threshold. (The opposite approach was proposed in the network context [11] and was one of those considered in [12], but the variation above is the more effective version [8].)

Polling-based algorithms: Bandwidth allocation and scheduling are more closely integrated. These scheduling algorithms have an explicit or implicit *polling* element, i.e. the packets from each class are guaranteed to receive some service periodically [13]-[22]. The simplest example is time-division multiplexing (TDM) in which some fixed interval within a repeated frame is reserved for transmission of class i packets. This framing mechanism is unnecessarily rigid, so one might improve performance by guaranteeing that a fixed number of class i packets can be transmitted in each frame without specifying the particular interval within the frame [13, 14, 15]. Of course an interval may be left idle if there are no class i packets to transmit. This idle interval can be exploited by allocating it to a given class (possibly best effort) [16, 17, 18], or by shortening the frame, effectively splitting the unused bandwidth among all classes with queued packets [19, 20, 21, 22]. Since some of the classes that share this unused bandwidth may not need it to meet performance requirements, the former approach is preferable for systems with best effort traffic. However, in a polling system, as long as there are class i packets queued, the number transmitted in a given frame does not depend on how many there are, or how long they have already waited. No state is maintained from one frame to the next. These schemes allow bandwidth to be reserved for a packet stream, thereby enabling performance guarantees, but they suffer from many of the same performance liabilities as static priority scheduling. For example, if bandwidth is reserved for a stream of voice packets, those packets will be transmitted in any frame during which there are voice packets queued; it is not possible to delay these voice packets in order to improve the performance of packets with mean-delay objectives, even when the voice packets are in no danger of missing their deadlines.

Virtual Clock [23]: Some state is maintained. A virtual clock v_i is associated with each packet stream i, and a_i is the average interarrival time stated for the packet stream. v_i roughly represents the time the next stream i packet would be transmitted in a TDM system, and the oldest packet from the stream with the earliest value for v_i is selected for transmission. Thus, if the channel would sit idle with TDM, the virtual clock will send some queued packet "early." When the first packet from stream i arrives, v_i is set to the current time. Whenever a packet from stream i is transmitted, v_i is incremented by a_i. Also, v_i is incremented when necessary to prevent its value from becoming

less than the actual time. Like the polling-based algorithms, bandwidth is effectively guaranteed. However, there is no way to select parameters such that a packet has a high priority if and only if it is in danger of not meeting performance objectives.

MAgnet II Real-time Scheduling (MARS) [24]: Packets are divided into three real-time classes. Classes 1 and 2 have maximum delays. Class 1 should experience no loss (where packets whose delay exceeds the maximum are considered lost). Class 2 loss should be minimized to the extent possible without interfering with class 1. All class 1 packets must have identical maximum delays, and the same for class 2. Class 3 packets should be transmitted as early as possible, thereby minimizing mean delay, to the extent possible without greatly affecting class 1 or 2. This is achieved by using a polling-based system, but the fraction of a frame allocated to each class is constantly changing to reflect the number of arrivals during each frame. As we will show, this is a relatively effective approach when these three classes represent all network traffic, but it is not clear how to expand this approach to support the more diverse performance objectives described in Section 1.

Leaky Bucket: The Leaky Bucket is a queue in which the departure rate of packets is controlled through the use of a token pool as described below. A packet can only be transmitted when the number T of tokens present is > 0. Whenever a packet is transmitted, T is decremented. T is incremented every period τ, unless $T = K$ where K is the maximum permissible number of tokens, so T ranges from 0 to K. The Leaky Bucket provides policing and decreases burstiness, but it is not designed as a scheduling algorithm to meet diverse performance objectives. To do this, one must use an appropriate scheduling algorithm in conjunction with the Leaky Bucket, e.g. the Leaky Bucket determines when a packet is transmitted, but CBS is used to determine the order.

3 The Priority Token Bank

Section 3.1 describes the basic scheduling mechanism of the Priority Token Bank. In Section 3.2, performance bounds are calculated which can constitute the basis of admission control, since they make performance guarantees possible. We apply these bounds to sample applications in Section 3.3.

3.1 Basic Scheduling Mechanism

Packets are divided into N classes such that all of the packets in a given class have the same performance objectives, such as a maximum queueing delay of 30 ms. Some classes contain only guaranteed packet streams, and the others contain only best-effort traffic. For example, class 1 might contain a few hundred guaranteed voice streams, class 2 might contain a guaranteed high-definition television (HDTV) stream, class 3 might contain best-effort telnet traffic, and class 4 might contain low-priority best-effort email. (Implementation can be simplified by associating an application identifier with a packet or packet stream, e.g. ID # 7 identifies HDTV, since performance objectives and arrival process characteristics are typically implicit in this identifier.)

For each class i, there is a first-in-first-out queue (FIFO), a token counter T_i, which can be positive or negative, and a constant τ_i. Additionally, each class has a priority function $P_i(T_i)$ which gives the priority of the oldest packet in the class i FIFO as a function of T_i. The next packet to be transmitted is the packet at the head of the FIFO whose priority function $P_i(T_i)$ currently has the largest value. The actual mechanism is similar to the Leaky Bucket, but unlike the Leaky Bucket, the algorithm is designed so that the current value of T_i reflects the queuing delay experienced to date by the oldest class i packet. Consequently, performance should be comparable to that of Cost

Based Scheduling [8]. T_i is incremented periodically with period τ_i and is decremented every time a class i packet is transmitted, regardless of the value of $T_i : -\infty < T_i < \infty$. One exception: T_i is held at zero when there are no packets queued, because that would be the current age of a packet arriving to the empty queue. The only mechanism that must be implemented in a Priority Token Bank that is not needed in the virtual clock or the Leaky Buckets is a priority comparison, which can be achieved at the necessary speeds using either a serial $O(N)$ or parallel $O(1)$ mechanism [25].

Note that some policing is inherent. With each class i packet transmission, the token counter T_i decreases, and $P_i(T_i)$ and the time between token arrivals τ_i can be selected so that priority decreases too. Thus, if arrival rate for a given class exceeds the declared rate, it can be given low priority, so the performance of other classes will not be degraded. (As with any system that divides traffic into classes, additional policing may be needed for those cases where streams in the same class must be protected from each other.)

We next demonstrate how it is guaranteed that class i packets will be selected for transmission regularly, regardless of the number of other packets queued. We will show in Section 3.2 how, as a result, the performance objectives of typical network applications can be guaranteed. If the number of classes is large, statistical multiplexing enables a less conservative approach, but we address the worst case here where there are a small number of classes. The priority of best-effort traffic is required to be $< G$, where G is a constant. For guaranteed traffic, if and only if $T_i \geq L_i$ for some constant limit $L_i : L_i \geq 0$, then $P_i \geq G$, thereby giving the guaranteed class i packets priority over all best-effort traffic. $\tau_j : \forall j$ and $P_j(T_j) : \forall j$ are chosen so that all class i packets with priority $\geq G$ are transmitted within τ_i packet transmission times, so T_i never exceeds the limit L_i. To insure this even in the worst case, consider the performance of class i traffic when all FIFOs other than FIFO i have an infinite number of packets queued, and no class i packet can ever be transmitted until its priority is $\geq G$. All packets with priority $< G$, including all best-effort traffic, can then be ignored. A class i packet can never be transmitted until T_i is incremented to L_i, and the packet must be transmitted before T_i would be incremented to $L_i + 1$. This is equivalent to a system considered by Liu and Layland [26] in which class i packets arrive periodically with period τ_i (for all i), and each class i packet must be transmitted before the arrival of the next class i packet. They showed that this can be achieved if jobs are scheduled earliest-deadline-first as long as load does not exceed 1, i.e. $\sum_{j \in R} 1/\tau_j \leq 1$, where R is the set of classes containing guaranteed packet streams.

Liu and Layland also showed that it can be guaranteed that no deadline will be missed with static priority scheduling, i.e. when P_i equals the same constant $\geq G$ whenever $T_i \geq L_i$, which is perhaps a more valuable result, given its simplicity. It is assumed here that all packet transmission times are equal, which we define to be unit length. (This assumption is valid for ATM networks; for other cases, the scheme can be extended.) All deadlines are met with static priority scheduling as long as priorities are *rate-monotonic*, which means that $\tau_j > \tau_k$ implies that $P_j < P_k$, and as long as the load from the set R of guaranteed packet streams meets the following restriction.

$$\sum_{j \in R} 1/\tau_j \;\leq\; |R| \cdot (2^{1/|R|} - 1)$$

where $1 \;\geq\; |R| \cdot (2^{1/|R|} - 1) \;>\; .69$ for all values of $|R|$. Thus, as long as the admission control algorithm prevents load from going too high, timely service of guaranteed traffic is insured. The fact that as much as 30% of the bandwidth may be unavailable for allocation to guaranteed traffic is often acceptable, because some bandwidth should probably remain available for best-effort traffic (e.g. ABR) any way, just as a GPS system might allocate a minimum fraction of bandwidth to best effort. Moreover, a congestion or flow control mechanism can then throttle the transmission

rate by 30% without unduly affecting the performance of the guaranteed traffic. However, it should be noted that the equations above are worst-case bounds, and it is generally possible to guarantee performance at much greater loads [27]. For example, if $\tau_j : \forall j$ are required to be powers of 2, a load of 1 from guaranteed streams can be supported.[1]

3.2 Calculating Delay Bounds

When bandwidth is allocated in this manner, much is known about the maximum queueing delay that can be experienced by a class i packet, as will now be demonstrated. This will form the basis of admission control, because performance must be guaranteed even after a new stream is admitted. Let packet 0 be a class i packet that arrives when there are no other class i packets queued, so $T_i = 0$. Packet $k : k > 0$ is defined to be the class i packet that arrives after packet $k-1$, and it arrives while packet $k-1$ is still queued. Because the FIFO has not been empty since packet 0 arrived, T_k must equal the number of times the counter has been incremented minus the number of class i packets transmitted. Let $D_k : k \geq 0$ be the queueing delay experienced by packet k. Because packet k must be transmitted before T_i would exceed L_i, it must not be incremented more than $L_i + k$ times before packet k is transmitted, i.e.

$$D_0 < (L_i + 1)\tau_i$$

$$D_k < (L_i + 1 + k)\tau_i - X_k$$

where X_k is the time between the arrivals of packet 0 and packet k. The distribution of X_k depends on the arrival process.

We will apply this bound with three arrival processes. Beginning with the simplest, in the special case of constant-bit-rate (CBR) periodic arrivals where $X_k = k\tau_i$, the maximum delay is $(L_i + 1)\tau_i$ for all packets, i.e., a packet's queueing delay (rounded to a multiple of τ_i) equals τ_i times the number of tokens queued T_i when the packet is transmitted, and that number must be $\leq L_i$.

As a more general example, consider the case where an application requesting admission for a packet stream specifies the *extended peak load*, ρ_p and I_p, which is also the extended peak arrival rate, since departure rate was defined to be 1. Extended peak load is defined as follows; no more than $m = \rho_p I_p$ packets will arrive during any peak averaging interval of duration I_p. In the special case where $m = 1$, this is the same as the previous case; otherwise, traffic can be burstier. The extended peak load traffic description, or a variation of it, has been used for many admission control algorithms, e.g. [13, 14, 15, 17]. With this description, $X_k \geq \lfloor k/m \rfloor I_p$. Let $\tau_i = 1/\rho_p$, so $I_p = m\tau_i$.

$$D_k \;<\; (L_i + 1 + k)\tau_i - \lfloor k/m \rfloor m\tau_i \;\leq\; (L_i + 1 + k)\tau_i - [k/m - (m-1)/m]m\tau_i = (L_i + m)\tau_i$$

In words, the delay bound for an arbitrary arrival stream is the sum of two values. One is proportional to the maximum token counter value L_i, as with CBR traffic. The other is proportional to m, which is roughly the maximum burst length. Thus, $D_k < (L_i + m)\tau_i = L_i/\rho_p + I_p$ for all k. ρ_p is a function of I_p and the arrival process of the given traffic class. Of course the arrival process

[1] This can be proven as follows. Let classes be numbered such that $P_j > P_{j+1} : \forall j$, implying that $\tau_j \leq \tau_{j+1} : \forall j$. An arriving class i packet can only miss its deadline if τ_i consecutive transmission periods are all devoted to packets from classes $< i$, which means τ_i such packets must arrive during this period. Since all periods are a power of 2, τ_i is a multiple of $\tau_j : \forall j < i$. Thus, exactly τ_i/τ_j class j packets arrive during this period of length τ_i. Consequently, the class i packet can miss its deadline only if $\sum_{j=1}^{i-1} \tau_i/\tau_j \geq \tau_i$, which means that $\sum_{j=1}^{i-1} 1/\tau_j \geq 1$. Therefore, no packet will miss its deadline unless the load from guaranteed packet streams exceeds 1.

cannot be controlled, but I_p can be chosen arbitrarily. For predefined application types as described in Section 3.1, the relation between ρ_p and I_p can be determined in advance, so the network is free to select the value of I_p that optimizes performance, possibly considering the current network state. Some tradeoffs involved in selecting I_p will be described as we consider some specific applications.

Alternatively, we can characterize the arrival process with parameters σ_L and ρ_L, where the amount of traffic arriving in a period of duration t is bounded by $\sigma_L + \rho_L t$. ρ_L is a measure of average arrival rate, and σ_L is a measure of burstiness. This was assumed in [20, 21], and is appropriate for a stream that is policed with a Leaky Bucket [28]. Furthermore, like extended peak load, it is always possible to select parameters such that the constraint is met. Delay bounds will be derived for $k \leq \sigma_L - 1$, and $k \geq \sigma_L$. For $k \leq \sigma_L - 1$, $X_k \geq 0$, so

$$D_k < (L_i + 1 + k)\tau_i - X_k \leq (L_i + 1 + k)\tau_i \leq (L_i + \sigma_L)\tau_i$$

For $k \geq \sigma_L$, $X_k \geq (k + 1 - \sigma_L)/\rho_L$. Let $\tau_i = 1/\rho_L$.

$$D_k < (L_i + 1 + k)\tau_i - X_k \leq (L_i + 1 + k)\tau_i - (k + 1 - \sigma_L)\tau_i = (L_i + \sigma_L)\tau_i$$

Thus, $D_k < (L_i + \sigma_L)\tau_i = L_i/\rho_L + \sigma_L/\rho_L$ for all k. Once again, the delay bound is the sum of two values. The first is proportional to the maximum token counter value L_i, as would be expected if the token counter T_i were a measure of queueing delay. The second (σ_L) measures traffic burstiness.

Although it is beyond the scope of this paper, similar bounds hold for a network of queues, even though the arrival process at the second queue may differ from that of the first.[2] Indeed, the advantages of the Priority Token Bank are even greater in a network of queues, as shown in [3]. For example, packets that arrive to the first queue at a constant rate may arrive at the next queue in a burst. However, this can occur only if the packets at the end of the burst were transmitted "early," i.e. they were not maximally delayed. Thus, the fact that packets at the end of a burst often experience larger queueing delays at the second queue is not a problem.

3.3 Appropriate Parameters for Sample Applications

We now address the selection of parameters for some typical applications. For a given ρ_p, a large I_p means that the packet stream can be burstier, so there is some incentive to select a small I_p. ρ_p is relatively insensitive to I_p when a class consists of a single packet voice stream, so a small value (relative to the maximum tolerable queueing delay) can be chosen for I_p. For example, in the simple case where arrivals are periodic, the best choice for I_p would be the period, and ρ_p would be its inverse. Let M be the maximum tolerable delay, and let the desired loss rate be 0. To achieve the desired maximum delay, assuming the maximum delay exceeds the period $M \geq I_p$ as is typical,

$$\tau_i = 1/\rho_p$$

$$L_i = \rho_p(M - I_p) = \rho_p M - 1$$

Equivalently, for the model based on Leaky Bucket policing, σ_L should be small (i.e. 1), $\tau_i = 1/\rho_L$, and $L_i = \rho_L M - \sigma_L = \rho_L M - 1$. Thus, the token counter limit is roughly proportional to the

[2] For example, if all class i packets follow the same source destination path, as would be the case for some (but perhaps not all) classes, $D_k < (\sum_{j=1}^{N} L_i^{(j)} + N - 1)/\rho_p + I_p + \sum_{j=2}^{N} P^{(j)}$. $L_i^{(j)} : 1 \leq j \leq N$ is the value of L_i in the jth queue, N is the number of queues along the source-destination path, $P^{(j)} : 2 \leq j \leq N$ is one packet transmission time plus propagation delay from queue $j-1$ to j, and τ_i is the same for all of these queues.

maximum tolerable delay. If the load from the new stream ($1/\tau_i = \rho_p = \rho_L$) plus the load from guaranteed packet streams that have already been admitted would exceed the amount of bandwidth that can be allocated (which, as described earlier, ranges from .69 to 1), then the call is rejected. Otherwise, the call can be accepted with the above parameters. Although we have required that, whenever $T_i \geq L_i$, P_i must equal a constant that is $\geq G$ and whose value is selected so that priorities are rate monotonic, no assumptions about P_i have been made when $T_i < L_i$. Thus, a loss rate of 0 is still guaranteed if $P_i = 0$ (where 0 is the smallest priority value) when $T_i < L_i$, yielding a step function for $P_i(T)$. As a result, all best-effort traffic and some guaranteed classes would have a greater priority than class i traffic as long as $T_i < L_i$.

Many applications, such as packet voice, can tolerate some packet loss with little degradation in performance. For example, a number of researchers have indicated that a loss rate of around 5% could be tolerated in packet voice, given that the loss probabilities of all packets are independent, although of course a lower loss rate is preferred. This fact can be exploited to improve the performance of other traffic by dropping some voice packets when those voice packets are otherwise likely to have a priority > 0, i.e., when T_i is great. This also means that it is only necessary to allocate enough bandwidth to accommodate those packets that would not be dropped, so blocking probabilities are thereby reduced. A simple way to achieve this is to drop every d_ith class i packet whenever $T_i > D_i$, where D_i and d_i are constants. (Alternatively, one could drop each packet with probability $1/d_i$.) However, it may be preferable for the application to mark packets as candidates for dropping based on the importance of the information they carry, and there is a bit for this purpose in the ATM packet header. It has been reported that by marking packets in this manner using an embedded code [29], or by marking packets based on the types of sounds that they carry [30], loss rates as high as 20% and 16%, respectively, can be tolerated with only slight degradation in voice quality. With this variation of the dropping mechanism, whenever the packet at the head of FIFO i is marked as droppable and $T_i \geq D_i$, that packet should be dropped.

An application such as bulk data transfer may have a performance objective of transmitting all B packets from a file within a fixed period. As before, $\tau_i = 1/\rho_p$ and $L_i = \rho_p(M - I_p)$, where ρ_p is the peak load over an averaging interval of duration I_p. Recall that any $I_p : I_p \leq M$ can be selected. In this case, there is an implicit tradeoff in the selection of I_p (and therefore ρ_p), since, assuming that all B packets arrive simultaneously, $\rho_p = B/I_p$. Thus, selecting a larger I_p means less bandwidth ρ_p need be allocated, thereby reducing the blocking probability of future requests. However, this also reduces L_i, thereby increasing the frequency at which class i priority $P_i \geq G$, and degrading the performance of other traffic. (The same effect is apparent in the model associated with Leaky Bucket policing, where ρ_L must be chosen such that $L_i = \rho_L M - \sigma_L > 0$, and a large ρ_L increases L_i but also increases the bandwidth required.) A demonstration of this effect will be shown in Section 4, which presents performance results in a few scenarios. In a bulk data transfer application, unlike typical voice and video applications, it is also useful to complete the transmission of all packets in a time significantly less than their maximum allowable queueing delay when possible, both because this is preferable to the user and because it frees up the allocated bandwidth a bit earlier, thereby decreasing the blocking probability. Consequently, the priority P_i when $T_i < L_i$ should be greater than that of a class j voice packet when $T_j < L_j$, and greater than the priority of relatively unimportant best-effort traffic, but still less than the priority of urgent best-effort traffic. For example, when $T_i < L_i$, then $P_i = G/2$, and when $T_i \geq L_i$, then P_i equals a constant $\geq G$ (such that priorities $\geq G$ are rate-monotonic). For a large data transfer, M is likely to be fairly large, so $P_i(T)$ when $T \geq L_i$ is probably not much larger than G.

For some applications, it is not maximum delay but mean delay that matters. Clearly perfor-

mance analysis for a given arrival process is not possible in a system as complicated as a Priority Token Bank if analysis is not possible in a simple FIFO. We assume that such analysis is complete, and a function $Q(\rho)$ is known (at least for predefined application types) which gives expected queueing delay as a function of the load ρ allocated to the packet stream, when circuit switching via simple time-division multiplexing with periodic service is used. At request time, the application specifies the greatest mean delay D that would be tolerable. The network determines a load ρ_r where $D \geq Q(\rho_r)$. The Priority Token Bank must then guarantee that no class i packet will experience a queueing delay worse than $D - Q(\rho_r)$ more than it would have with periodic service with period $1/\rho_r$. If class i packets were served periodically, then, with packets numbered as before, packet k could experience a delay as great as $(k+1)/\rho_r - X_k$. With the Priority Token Bank, as shown above, queueing delay cannot exceed $(L_i + 1 + k)\tau_i - X_k$. Thus, the desired performance can be achieved if the parameters are set as follows.

$$\tau_i = 1/\rho_r$$

$$L_i = \rho_r(D - Q(\rho_r))$$

$\rho_r = 1/\tau_i$ can be thought of as the amount of bandwidth allocated to class i. Recall that any $\rho_r : D \geq Q(\rho_r)$ can be selected. Implicit in the exact selection of ρ_r is the same tradeoff as described when selecting the interval I_p for a file transfer. A large ρ_r increases the blocking probability of future requests. However, by making more bandwidth available to class i when it is really needed, class i can be given low priority even more frequently, thereby improving the performance of other guaranteed and best-effort traffic. Class i gets priority $> G$ less often because the token counter limit L_i is increased when ρ_r is increased (which also decreases $Q(\rho_r)$).

Finally, we consider best-effort traffic. The only restriction on best-effort priorities is that they should always be less than G. (The same can be said of guaranteed traffic when $T_i < L_i$.) With packet streams for which performance is measured in mean delay, simple static priority is appropriate [31], i.e., P_i always equals some constant > 0 and $< G$. For other types of performance objectives, P_i can be made to vary with T_i much the same way it would vary with queueing delay in algorithms such as Cost-Based Scheduling (CBS) [8], as described in Section 2. Indeed, precisely what makes the Priority Token Bank effective is that T_i is an indirect, approximate measure of the queueing delay the packet at the front of FIFO i has already experienced. This is why, for example, it was appropriate for traffic with maximum delay restrictions for P_i to be high when T_i is high, and therefore the queueing delay of the oldest packet is approaching the stated upper limit, and P_i is low otherwise. In a best-effort packet with a performance objective based on loss rate, P_i would similarly be small when T_i is small, and P_i would grow with T_i.

4 Performance Results

In this section, we evaluate the performance of the Priority Token Bank (PTB) through comparison with alternative algorithms that can also be used to guarantee that performance objectives will be met. The scheduling algorithms proposed most frequently in the context of high-speed networks are static priority (SP), where guaranteed packet streams are given greater priorities, and the polling-based methods (POLL) such as [13]-[22]. The results shown apply to work-conserving polling-based algorithms unless explicitly stated to the contrary. (Under the assumptions below, where individual packets are of negligible size and bandwidths are infinitely divisible, all work-conserving algorithms such as weighted fair queueing, generalized processor sharing, and round robin, will perform the

same.) Comparisons will be made with an idealized MARS system of unlimited complexity, i.e. MARS will set polling parameters for an unlimited number of future frames, each of minimal duration. Comparisons will also be made with Cost-Based Scheduling (CBS), which is very effective but more difficult to implement, the virtual clock (VC), and an occupancy-based approach (OCC). (These acronyms will be used for figure labels.) As described in Section 2, it is not clear whether or how the Leaky Bucket alone (i.e. without some other scheduling algorithm as in [20, 21]) could be used to meet the performance objectives of an arbitrary number of heterogeneous packet streams, so the Leaky Bucket alone is not considered in this section. Earliest deadline first is also not included explicitly, because in the scenarios described it would be necessary to give the packets of one traffic class infinite deadlines making the algorithm equivalent to static priority.

One measurement criteria is the performance that can be achieved for one class of (guaranteed or best-effort) traffic under the constraint that the performance objectives of another class of (guaranteed) traffic are met, which is a good way to compare scheduling algorithms. The other criterion used for comparison is the *schedulable region* [24], which some systems have used as a basis for admission control. We use it here to compare the loads possible with the various algorithms. In figures showing these schedulable regions, the axes are the load ρ_1 from class 1 traffic and the load ρ_2 from class 2 traffic. Performance requirements can be met with a given algorithm at a load $\rho_1 = x$, $\rho_2 = y$, if the point (x, y) is in the schedulable region, which is the region bounded by the axes and the curve associated with the given algorithm. If a new call would push the load outside the schedulable region, the call must be blocked. Otherwise, there is a choice to be made.

For each algorithm and each scenario, parameters are selected to optimize performance. As examples, we consider four scenarios in these comparisons, all of which share the following assumptions. (Of course, the Priority Token Bank is still applicable without these assumptions.) The channel bandwidth is 150 Mb/s. Performance objectives relate to the queueing delay within a single queue, which is appropriate if queueing delay at the network access point exceeds queueing delay within the network, as discussed in Section 1. (Scenarios where total queueing delay is comparable i.e. around 30 ms, but is distributed across a number of bottleneck queues, will be discussed in [3].) There are two classes of traffic. The loads from classes 1 and 2 are ρ_1 and ρ_2, respectively. Class 2 traffic consists of data bursts of exponentially distributed length with mean β. Bursts arrive according to a Poisson process, and their mean queueing delay Q_2 is the performance measure. Burst lengths are assumed to be large compared to packet length, which is considered to be negligible. (After all, transmission time of one 53-byte ATM cell is very small.) Consequently, bursts can be preempted at any time, and if a packet stream remains exactly constant bit rate (CBR) with load ρ throughout a period of duration t, the amount of data arriving is exactly $t\rho 150$ Mb/s. Also, polling-based algorithms and MARS can give each stream precisely the desired fraction of the bandwidth, regardless of frame length. Consequently, two work-conserving (or two non-work-conserving) polling-based algorithms can differ only in how packets are ordered within a frame, and as long as the frame length is relatively small, the algorithms' performance must be equivalent. Furthermore, the duration of a MARS frame is assumed to be negligible compared to the delays that can be tolerated, which should lead to ideal performance for MARS. Four types of traffic are considered for class 1, each of which is intended to approximate a potentially common network application: packet voice in Section 4.1, high-definition television (HDTV) in Section 4.2, standard video in Section 4.3, and bulk data transfer in Section 4.4. In each case, with the Priority Token Bank, class 1 has greater priority than class 2 if and only if $T_1 \geq L_1$ so that class 1 packets are in danger of not meeting their performance objectives, i.e. $P_1(L_1) > P_2(T_2) > P_1(T_1)$ for all T_2 and for all $T_1 < L_1$. This would be appropriate if class 2 were best-effort, or if it were guaranteed and $\tau_2 \geq \tau_1$.

There are exact analytic tools to determine performance in some scenarios (for SP, OCC, CBS, and PTB with 0% loss in Section 4.1) [32, 33], and efficient simulation tools for other scenarios [34]. For all simulation results, the 95% confidence interval is, at worst, within 5% of the values shown.

4.1 Voice Traffic

Let class 1 consist of many independent voice packet streams multiplexed together, which would not be bursty. Thus, we consider here a CBR source for which it must be guaranteed that maximum delay not exceed some threshold M. As mentioned in Section 3, with a CBR source, the queueing delay of the oldest class 1 packet (rounded to the nearest τ_i) is proportional to the number of class 1 tokens queued. Since the size of each packet was assumed to be negligible, so are interarrival periods, and therefore τ_1. Consequently, transmitting a voice packet whenever the number of tokens queued reaches the limit L_1 is equivalent to transmitting whenever the queueing delay of the oldest packet exactly reaches its maximum, or when the number of queued packets reaches a maximum. Thus, in this case, the Priority Token Bank is equivalent to scheduling algorithms that are based on queueing delay such as CBS, and to occupancy-based scheduling. Furthermore, since voice packets are always transmitted before their deadline, voice packets are only given a greater priority than class 2 packets when doing otherwise would lead to the loss of a voice packet, and packets are always transmitted when there are any queued, this algorithm is optimal with respect to minimizing the mean delay Q_2 of class 2 traffic under the constraint that no voice packets are lost.

We now consider performance with polling-based algorithms, the virtual clock, and static priority. Since data arrives at a constant rate and packet lengths are negligible, the amount of class 1 data arriving during any period of length t is exactly $t\rho_1 150$ Mb/s. Thus, if t is the frame length in a polling-based scheme, the same amount of voice data arrives in each frame, so it is always possible to transmit voice packets as they arrive without exceeding the frame limit. If instead voice packets had a higher static priority, they would similarly be transmitted right away. Consequently, in this case, polling is equivalent to static priority. Furthermore, static priority is also equivalent to the virtual clock when class 1 loss rate must be minimal. Class 1 losses can only be avoided if the virtual clock is incremented by an amount no greater than the packet interarrival time. Consequently, arriving voice packets always have the greatest possible priority and are transmitted immediately.

We now compare the mean delay Q_2 achieved with the various algorithms. Let the maximum allowable queueing delay for class 1 traffic $M = 30$ ms. Figure 1 shows the queueing delay Q_2 of class 2 traffic as a function of the load ρ_2 from class 2 traffic, with the mean burst length $\beta = 500$ kb and with the load from voice $\rho_1 = .5$. Clearly, performance is much better with the Priority Token Bank, CBS, and occupancy-based scheduling than with polling, static priority, or the virtual clock. This is because the former algorithms give voice packets higher priority only when they are in imminent danger of missing their deadlines. The Priority Token Bank also outperforms MARS, although the difference is not great. MARS occasionally gives voice packets higher priority when it isn't necessary. Consider the case where load from CBR voice is .5, and there has been some class 2 traffic queued for an extended period. All voice packets are therefore being maximally delayed, and each class uses 50% of the bandwidth on the outgoing link. When the class 2 queue becomes empty, voice packets suddenly fill the outgoing link, causing them to be transmitted earlier than necessary. Then, another class 2 burst arrives before the voice queue can empty. Although it would be possible at that instant to give 100% of the bandwidth to class 2 without missing any voice deadlines, MARS will always use some of the bandwidth for voice. Consequently, MARS performance is always somewhat inferior to that of the Priority Token Bank with CBR traffic.

Also shown in Figure 1 is the mean delay Q_2 achieved with a Priority Token Bank which drops the fraction of voice packets that the application can tolerate when token counter $T_i = L_i$ (i.e., $D_i = L_i$). Results are shown where 5% of traffic can be dropped when $T_i = L_i$, as might be appropriate if the Priority Token Bank is selecting voice packets to drop arbitrarily, and 20% can be dropped, as might be appropriate if the application identifies the packets that could be dropped [29, 30]. Clearly the mean delay is greatly reduced when the Priority Token Bank allows some loss, particularly at high loads. Although short-term loss rates are allowed to be as high as 5% and 20%, respectively, the long-term average loss-rate is much smaller. Even with a total network utilization of .95, and a short-term loss as great as 20% for voice packets, the long-term loss rate is still around 4%. The fact that the Priority Token Bank has a simple built-in capability to drop these packets when and only when there is congestion is therefore another potential advantage.

Another important criteria for evaluating the algorithms is the schedulable regions which are shown in Figure 2, where the largest tolerable mean queueing delay for class 2 (excluding transmission time) is 5 ms. Results are shown when the greatest tolerable loss rate for CBR voice is 0%, 5%, and 20%. This time, the maximum delay M for class 1 traffic is only 20 ms. Clearly, CBS, the Priority Token Bank, and occupancy-based scheduling all carry much more traffic than the polling-based algorithms, the virtual clock, or static priority, even with no packet loss. When some packets can be dropped during periods of great congestion, they do even better. Performance with MARS is again slightly worse than that of the Priority Token Bank.

We now consider the effect of the maximum delay M for voice traffic. Figure 3 shows Q_2 as a function of M with $\beta = 500$ kb and $\rho_1 = \rho_2 = .45$. This figure shows that the relative performance of the Priority Token Bank is even better if queueing delays greater than 20 or 30 ms can be tolerated in voice streams. This would not be surprising; propagation delay across the U.S. would not exceed 30 ms, and a total delay of 60 ms is not great. Even the differences between the Priority Token Bank and MARS become quite pronounced when M is large, because the aforementioned scenario where only voice packets are queued when a class 2 burst arrives becomes more common.

4.2 HDTV Traffic

We now consider the case where the traffic for which performance must be guaranteed does not arrive at a constant rate; it is bursty. Since multiplexing numerous independent sources reduces burstiness (at least for traffic that can be described with traditional Markovian models), let us consider the worst case where class 1 traffic is generated by a single variable bit-rate (VBR) video packet stream. For one video source to use a significant portion of a 150 Mb/s channel, it must be HDTV. Maglaris et al [35] characterized the arrival process of VBR video with a fixed number of independent sources, each of which alternates between active and inactive periods, where the distributions of both are exponentially distributed. Data is generated at a rate proportional to the number of currently active sources. They discovered that a good approximation of the arrival process for VBR video can be achieved with 10 sources, and mean active and inactive periods of 386 ms and 765 ms, respectively. While there is current controversy over the most appropriate model for video, many researchers have proposed comparable Markovian fluid-flow models. As an example, we will assume here that this model is appropriate for HDTV, with data rate scaled such that mean rate is 22.5 Mb/s.

Let the maximum tolerable queueing delay M for video packets be 30 ms, and the loss rate be 0 for all algorithms. To determine the parameters for the Priority Token Bank, the peak load ρ_p must be known over some interval I_p. With this model, the probability that all 10 sources are active at a given time is $2 * 10^{-5}$, and the mean duration of these periods (until the load drops

to 90% of the peak) is 38.6 ms. Since the instantaneous peak rate is often generated even longer than the maximum tolerable delay, we must guarantee that the maximum queueing delay of 30 ms would not be exceeded even if the arrival rate always equaled the instantaneous peak rate of 67 Mb/s. Since ρ_p = 67 Mb/s independent of I_p, we should choose a negligible I_p. (It would be possible to allocate less than the instantaneous peak rate by selecting a larger I_p if burstiness could be reduced by multiplexing multiple video streams together.) As with CBR voice, arriving HDTV packets can always be transmitted in the current frame, so the polling-based approach is equivalent to giving HDTV packets a greater static priority. Furthermore, static priority and the polling-based algorithms are again both equivalent to the virtual clock: to prevent HDTV losses, the virtual clock must be incremented by an amount no greater than the declared interarrival period at peak rate. Consequently, a video packet's virtual clock will never exceed its arrival time, so it will always have the greatest possible priority. These algorithms are also equivalent to occupancy-based scheduling. If class 1 is not given high priority at all occupancy-levels, losses can occur when the HDTV arrival rate is low. The occupancy-based approach does poorly with bursty traffic such as HDTV, where arrival rates can be low.

Figure 4 shows the mean delay Q_2 of class 2 traffic achieved with the various scheduling algorithms as a function of the load ρ_2 from class 2 traffic, where β = 500 kb. Because the Priority Token Bank, CBS, and MARS give priority to video packets only when they are in danger of missing their deadlines, their performance is again better than that of polling, static priority, or the virtual clock. Occupancy-based scheduling also does poorly because HDTV is bursty. Burstiness also affects the Priority Token Bank and MARS, causing them both to give priority to HDTV even when the oldest video packet has not been maximally delayed. Thus, CBS performance is a bit better. Burstiness affected the Priority Token Bank somewhat more than MARS in this scenario, causing MARS to achieve slightly better performance this time.

4.3 Standard Video

We use the same VBR model for each stream as above, but average data rate for a single stream is 1 Mb/s. No packet loss is allowed with any algorithm. The optimal value of I_p is not known, but for simplicity, a negligible I_p was selected for the Priority Token Bank, and ρ_p was chosen such that the probability of load exceeding ρ_p is less than 10^{-7}. This "peak" rate also determines the parameters for the virtual clock, which must accommodate this rate without loss. Figure 5 shows the queueing delay Q_2 of class 2 traffic when class 1 consists of 45 independent VBR video sources (ρ_1 = .3). The Priority Token Bank greatly outperforms the polling-based algorithms and static priority, and performance is relatively close to that of CBS. Unlike the HDTV example, however, occupancy-based scheduling does well here, because the difference between the minimum and average data rates for video is not as great. Figure 6 shows the schedulable regions when class 1 traffic can tolerate a 5 ms mean queueing delay (excluding transmission time). A much greater load can be supported with the algorithms that give video packets high priority only when they are in danger of missing their deadlines: the Priority Token Bank, MARS, CBS, and occupancy-based scheduling.

4.4 Bulk Data Transfer

Finally, we consider the case where class 1 is for a bulk data transfer. All of the packets in a 30 Mb file must be transmitted within 500 ms. No packet losses are tolerable with any algorithm. Clearly the Priority Token Bank would require an allocation of at least .4, as would any other scheme, but as

described in Section 3, even better performance for class 2 traffic can be achieved if more bandwidth is allocated. Figure 7 shows the mean delay Q_2 of class 2 traffic as function of the load ρ_p allocated for the bulk data transfer with $\beta = 500$ kb, where file transfers arrive every $M = 500$ ms, so that there is always one file transfer underway. Curves are shown for $\rho_2 = .3$, $\rho_2 = .4$, and $\rho_2 = .5$. Obviously, if the possibility of blocking future calls is not considered, ρ_p should be chosen to be as great as possible. However, the slope of these curves is steepest when ρ_p is near the minimum, so a large improvement can be obtained by allocating a relatively small amount of extra bandwidth; there is little incentive to set ρ_p to 1. The performance gains from allocating extra bandwidth are especially significant when the load is high. The tradeoff of improving performance for existing calls versus reducing the blocking probability of future calls is explored in future work, where the ρ_p chosen depends on current load. However, the mere fact that this tradeoff is possible is an advantage of the Priority Token Bank over alternatives.

Figure 8 shows the mean delay Q_2 of class 2 traffic as a function of the load ρ_2 for several algorithms. We know from Figure 7 that the delay Q_2 experienced depends on the load ρ_p allocated for bulk data transfer. Therefore, we show the performance of the Priority Token Bank using three different values of ρ_p: $\rho_p = 1$, for which performance is the best, $\rho_p = .5$, and $\rho_p = .4$, which yields a blocking probability at least as good as any polling-based algorithm. Performance with the Priority Token Bank when $\rho_p = 1$ is as good as could be achieved with CBS or any other algorithm. In this scenario, virtual clock is equivalent to the Priority Token Bank with a worst-case allocation of .4; the Priority Token Bank does better than virtual clock with a greater allocation. (Virtual clock parameters must be set so that the class 2 transmission rate is .4.) MARS and the occupancy-based approach both do very poorly in this scenario. With the occupancy approach, the only way to insure that the file transfer is transmitted in time is to give it priority when any packets are queued, i.e. to make it static priority. Again, we see the problems of occupancy-based approaches for highly bursty traffic. Similarly, with MARS as previously defined, the only solution is to make MARS equivalent to static priority. It would not be too difficult to add new features to MARS that would allocate a portion of every frame to the file transfer, making MARS equivalent to polling. This would be an improvement, but is still far worse than the Priority Token Bank. As for the polling-based algorithms, unlike the previous scenarios, they don't all achieve the same performance. Results are shown for a work-conserving version (POLLW) and a non-work-conserving version (POLLN). Neither version can equal the Priority Token Bank. With a polling based algorithm, when there are both class 1 and class 2 packets queued, 40% of the packets transmitted must be class 1, even when class 1 packets are in no danger of violating performance requirements. With the Priority Token Bank or virtual clock, it is possible to transmit 100% class 2 packets when class 1 transmissions are already "ahead of schedule" with respect to meeting their performance requirements.

The advantages of the Priority Token Bank are further demonstrated in Figure 9, which shows the schedulable regions. Class 2 traffic can tolerate a mean queueing delay up to 5 ms. File transfers are initiated every 500 ms, but file size is a function of the actual load ρ_1 from file transfers. Results are shown with the Priority Token Bank for the case where the load allocated for class 1 $\rho_p = \rho_1$, $\rho_p = \min(1, \rho_1 + .1)$, $\rho_p = \min(1, \rho_1 + .2)$, and $\rho_p = 1$. (Clearly, in the case where $\rho_p = 1$, no guarantee is possible for class 2, but the results are still applicable if class 2 happens to be best effort.)

5 Conclusion

This paper has demonstrated a new mechanism, the *Priority Token Bank*, for admission control, scheduling, and policing. The performance achieved with the Priority Token Bank was compared to

alternative algorithms such as static priority, MARS [24], occupancy-based scheduling, the virtual clock [23], Cost-Based Scheduling (CBS) [8], and polling based algorithms [13]-[22] such as weighted fair queueing, generalized processor sharing (GPS), stop-and-go queueing, and hierarchical round robin. At a given load, the performance achieved with the Priority Token Bank was much better than performance with the virtual clock, static priority or the polling-based algorithms. Equivalently, the same performance could be achieved with the Priority Token Bank at a much greater load than is possible with these other algorithms, so more packet streams can be accommodated. There are scenarios where MARS or occupancy-based scheduling can slightly outperform the Priority Token Bank, but when the difference is great, it is the Priority Token Bank that performs better. Furthermore, performance with the Priority Token Bank was close to that of CBS, an algorithm which would be more difficult to implement.

Overall, although it is more complex to implement than the simplest schemes, the Priority Token Bank has significant performance advantages over competing approaches when operating in a single queue. Subsequent work [3] will demonstrate that these advantages are at least as great in a network of queues.

References

[1] Q. Wang, J. M. Peha, and M. Sirbu, "Optimal Pricing for Integrated-Services Networks," *Internet Economics*, J. Bailey and L. McKnight editors, MIT Press, 1997, pp. 353-76.

[2] J. M. Peha and S. Tewari, "The Results of Competition Between Integrated-Services Telecommunications Carriers," accepted to appear in *Information Economics and Policy*, Special Issue on Multimedia Economics.

[3] M. Lynn and J. M. Peha, "The Priority Token Bank in a Network of Queues," *Proc. IEEE International Conference on Communications (ICC)*, June 1997, pp. 1387-91.

[4] S. S. Panwar, D. Towsley, J. K. Wolf, "Optimal Scheduling Policies for a Class of Queues with Customer Deadlines to the Beginning of Service," *J. ACM*, Vol. 35, no. 4, pp. 832-44, Oct. 1988.

[5] P. P. Bhattacharya and A. Ephremides, "Optimal Scheduling with Strict Deadlines," *IEEE Trans. Automat. Contr.*, Vol. 34, no. 7, pp. 721-8, July 1989.

[6] J. M. Peha and F. A. Tobagi, "Evaluating Scheduling Algorithms For Traffic With Heterogeneous Performance Objectives," *Proc. IEEE Globecom,* Dec. 1990, pp. 21-7.

[7] J. M. Peha, "Heterogeneous-Criteria Scheduling: Minimizing Weighted Number of Tardy Jobs and Weighted Completion Time," *Computers and Operations Research*, vol. 22, no. 10, Dec. 1995, pp. 1089-1100.

[8] J. M. Peha and F. A. Tobagi, "Cost-Based Scheduling and Dropping Algorithms To Support Integrated Services," *IEEE Trans. Commun.*, Vol. 44, no. 2, Feb. 1996, pp. 192-202.

[9] J. M. Peha and F. A. Tobagi, "Implementation Strategies for Scheduling Algorithms in Integrated Services Packet-Switched Networks," *Proc. IEEE Globecom*, Dec. 1991, pp. 1733-40.

[10] H. G. Badr, I Mitrani, and J. R. Spirn, "An Adaptive Priority Queue," *Applied Probability - Computer Science: The Interface*, R. L. Disney and T. J. Ott, editors. ORSA-TIMS, 1982, Vol. 2, pp. 345-71.

[11] R. Chipalkatti, J. F. Kurose, and D. Towsley, "Scheduling Policies for Real-Time and Non-Real-Time Traffic in a Statistical Multiplexer," in *Proc. IEEE Infocom,* Apr. 1989, pp. 774-83.

[12] D. Lee and B. Sengupta, "Queueing Analysis of a Threshold Based Priority Scheme for ATM Networks," *IEEE/ACM Trans. Networking,* vol. 1, no. 6, pp. 709-17, Dec. 1993.

[13] S. J. Golestani, "Congestion-Free Communication in High-Speed Packet Networks," *IEEE Trans. Commun.*, Vol. 39, no. 12, pp. 1802-12, Dec. 1991.

[14] S. J. Golestani, "A Framing Strategy for Congestion Management," *IEEE J. Select. Areas Commun.*, Vol. 9, no. 7, pp. 1064-77, Sept. 1991.

[15] A. Banerjea and S. Keshav, "Queueing Delays in Rate Controlled ATM Networks," *Proc. IEEE Infocom,* March 1993, pp. 547-56.

[16] K. Kümmerle, "Multiplexer Performance for Integrated Line and Packet-Switched Traffic," *Proc. ICCC,* 1974, pp. 517-23.

[17] C. R. Kalmanek, H. Kanakia, and S. Keshav, "Rate Controlled Servers for Very High-Speed Networks," *Proc. IEEE Globecom,* Dec. 1990, pp. 12-20.

[18] K. Sriram, "Dynamic Bandwidth Allocation and Congestion Control Schemes for Voice and Data Multiplexing in Wideband Packet Technology," *Proc. IEEE ICC*, Apr. 1990, pp. 1003-9.

[19] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Proc. ACM Sigcomm*, Sept. 1989, pp. 1-12.

[20] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Trans. Networking*, Vol. 1, no. 3, pp. 344-57, June 1993.

[21] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple-Node Case," *IEEE/ACM Trans. Networking*, Vol. 2, no. 2, pp. 137-50, Apr. 1994.

[22] D.D. Clark, S. Shenker, L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism," *Proc. ACM Sigcomm*, Aug. 1992, pp. 17-20.

[23] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks," *Proc. ACM Sigcomm,* Sept. 1990, pp. 19-29.

[24] J. Hyman, A. A. Lazar, G. Pacifici, "MARS: The Magnet II Real-Time Scheduling Algorithm," in *Proc. ACM Sigcomm,* Sept. 1991, pp. 285-93.

[25] T. Hsu and L. Kung, "A Hardware Mechanism for Priority Queue," *ACM Computer Arch. News*, Vol. 17, no. 6, pp. 162-9, Dec. 1989.

[26] C. L. Liu and J. W. Layland, , "Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment," *J. ACM*, Vol. 20, no. 1, Jan. 1973, pp. 46-61.

[27] J. P. Lehoczky, L. Sha, and Y. Ding, "A Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior," *Proc. IEEE Real-Time Sys. Symp.,* Dec. 1989, pp. 166-71.

[28] R. L. Cruz, "A Calculus for Network Delay, Part 1: Network Elements in Isolation" and "Part 2: Network Analysis," *IEEE Trans. Info. Theory*, Vol. 37, no. 1, Jan. 1991, pp. 114-31, 132-41.

[29] N. Yin, S. Li, and T. E. Stern, "Congestion Control for Packet Voice by Selective Packet Discarding," *IEEE Trans. Commun.*, Vol. 38, no. 5, pp. 674-83, May 1990.

[30] D. W. Petr, L. A. DaSilva, Jr., and V. S. Frost, "Priority Discarding of Speech in Integrated Packet Networks," *IEEE J. Sel. Areas Commun.*, Vol. 7, no. 5, pp. 644-56, June 1989.

[31] J. M. Harrison, "Dynamic Scheduling of a Multiclass Queue: Discount Optimality," *Operations Research*, Vol. 23, no. 2, March-April 1975, pp. 370-82.

[32] J. M. Peha, "Analysis of Scheduling Algorithms for Integrated-Services Networks using a Semi-Fluid-Flow Model," *Proc. IEEE Globecom,* Dec. 1992, pp. 330-4.

[33] J. M. Peha, "Evaluating Scheduling in Integrated-Services Networks using a Semi-Fluid Flow Model," accepted to appear in *Telecommunication Systems: Modeling, Analysis, Design, and Management.*

[34] J. M. Peha, "Simulating ATM Integrated-Services Networks," *Proc. 29th Annual IEEE/ACM/SCS Simulation Symp.*, Apr. 1996, pp. 162-71.

[35] B. Maglaris, D. Anastassiou, P. Sen, G. Karlsson, and J. D. Robbins, "Performance Models of Statistical Multiplexing in Packet Video Communications," *IEEE Trans. Commun.*, Vol. 36, no. 7, pp. 834-44, July 1988.

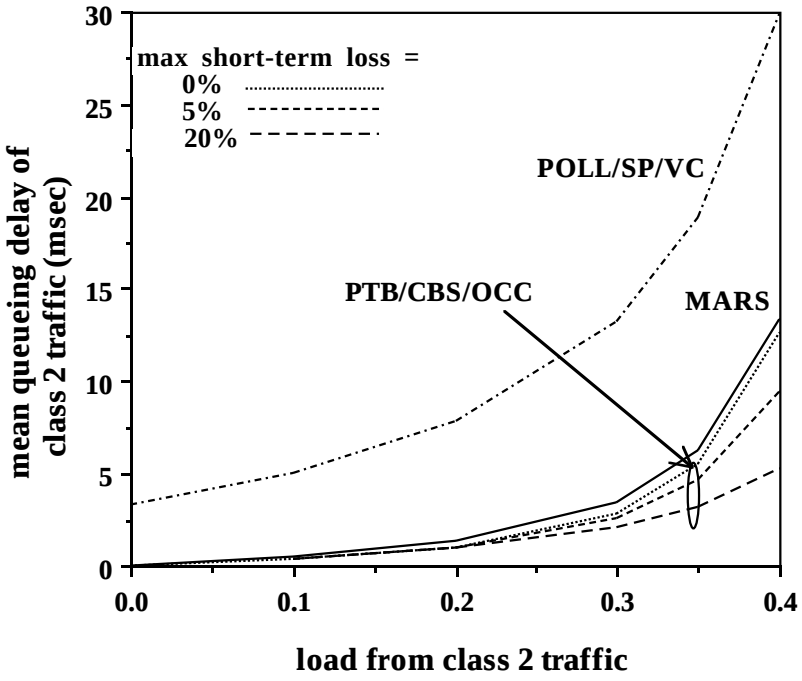

Figure 1: Mean delay Q_2 of class 2 traffic as a function of load ρ_2 from class 2 traffic with CBR voice class 1 traffic, $\beta = 500$ kb, $\rho_1 = .5$, $M = 30$ ms.

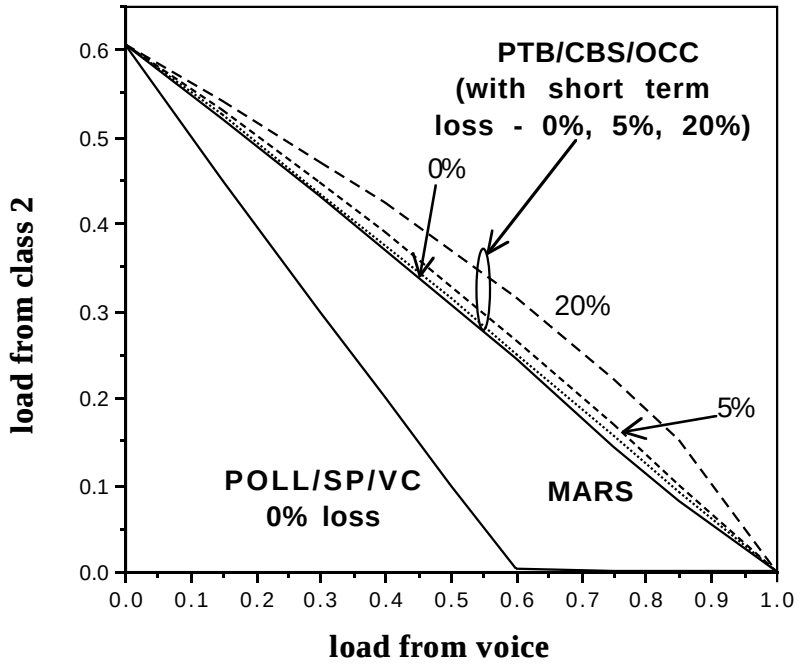

Figure 2: Schedulable regions: Load from class 2 traffic versus load from class 1 (CBR voice) traffic, $M = 20$ ms, $Q_2 \leq 5$ ms, Loss rate $\leq 5\%$, $\beta = 500$ kb.

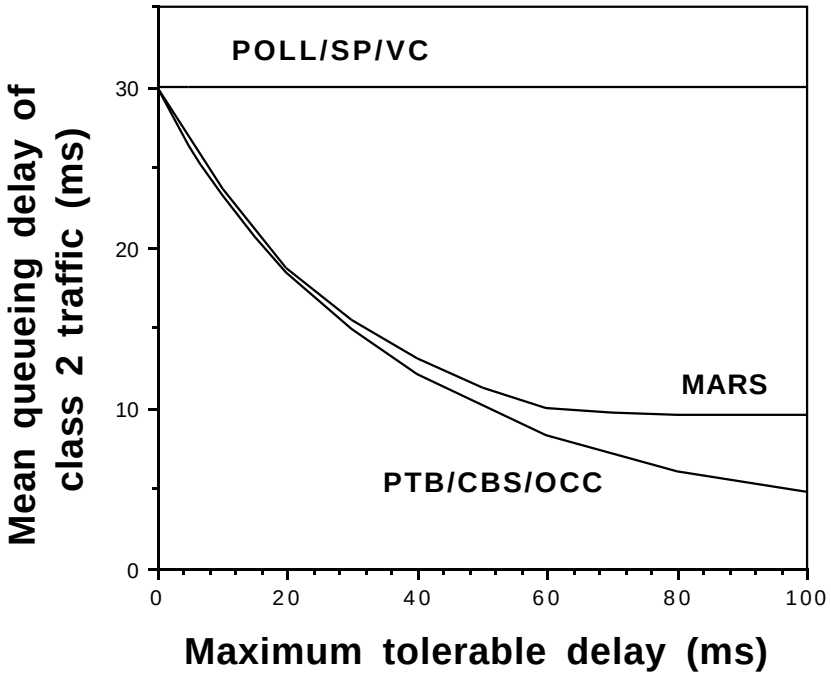

Figure 3: Mean delay Q_2 of class 2 traffic as a function of maximum tolerable delay M for CBR voice, $\rho_1 = \rho_2 = .45$, $\beta = 500$ kb.

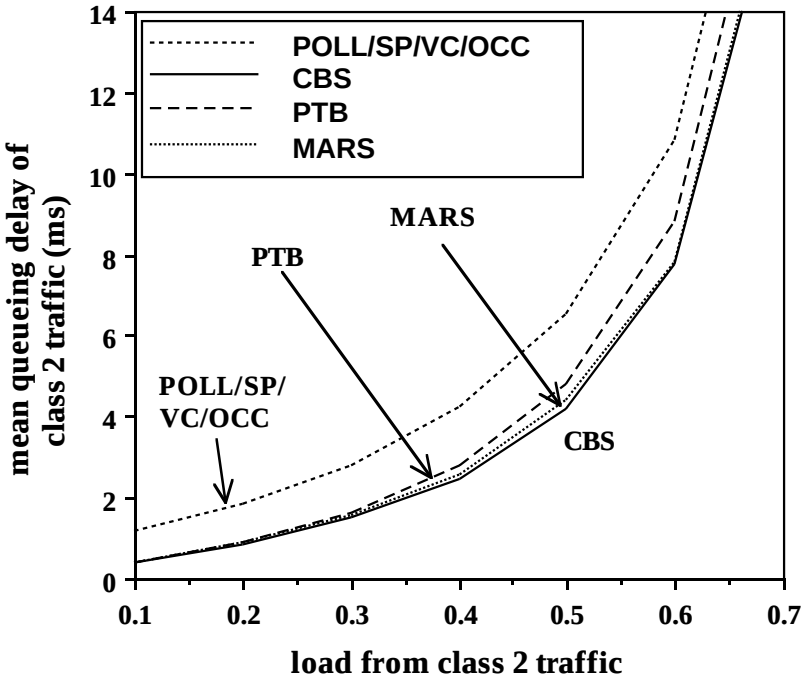

Figure 4: Mean delay Q_2 of class 2 traffic as a function of load ρ_2 from class 2 traffic with VBR HDTV class 1 traffic, $\beta = 500$ kb, $M = 30$ ms.

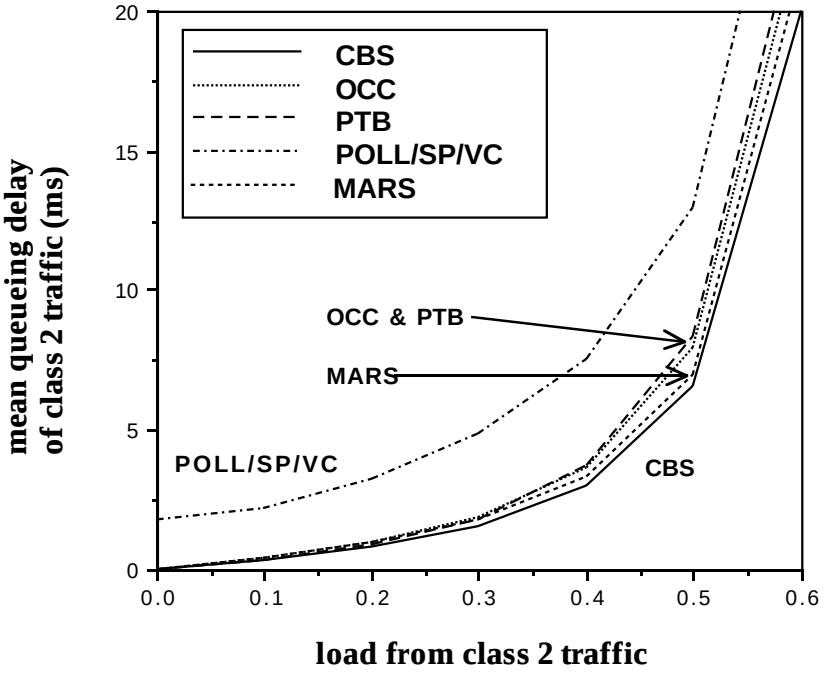

Figure 5: Mean delay Q_2 of class 2 traffic as a function of load ρ_2 from class 2 traffic when class 1 consists of 45 1 Mb/s VBR video streams, $\beta = 500$ kb, $M = 30$ ms.

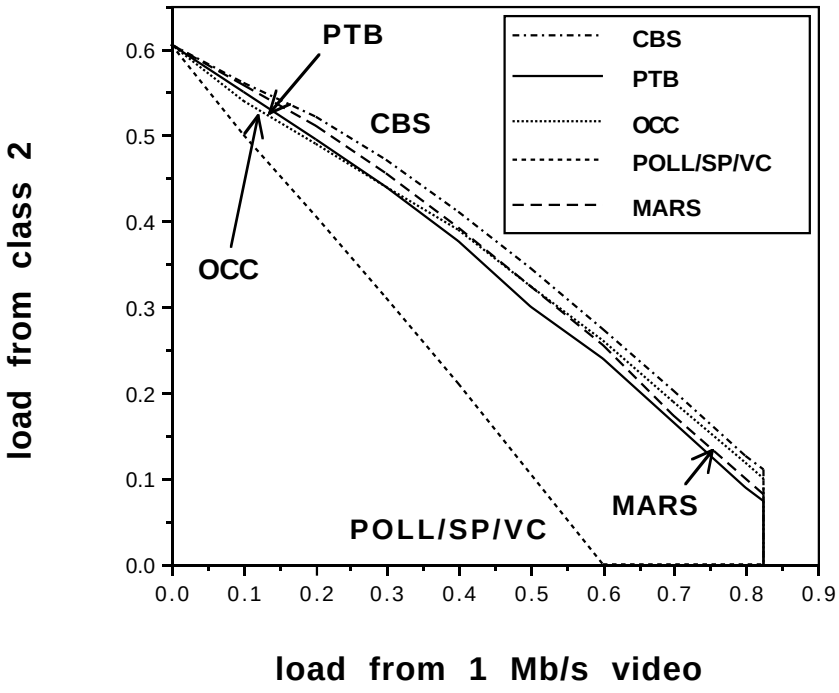

Figure 6: Schedulable regions: Load from class 2 traffic versus load from class 1 (1 Mb/s VBR video) traffic, $M = 30$ ms, $Q_2 \leq 5$ ms, $\beta = 500$ kb, Loss rate = 0.

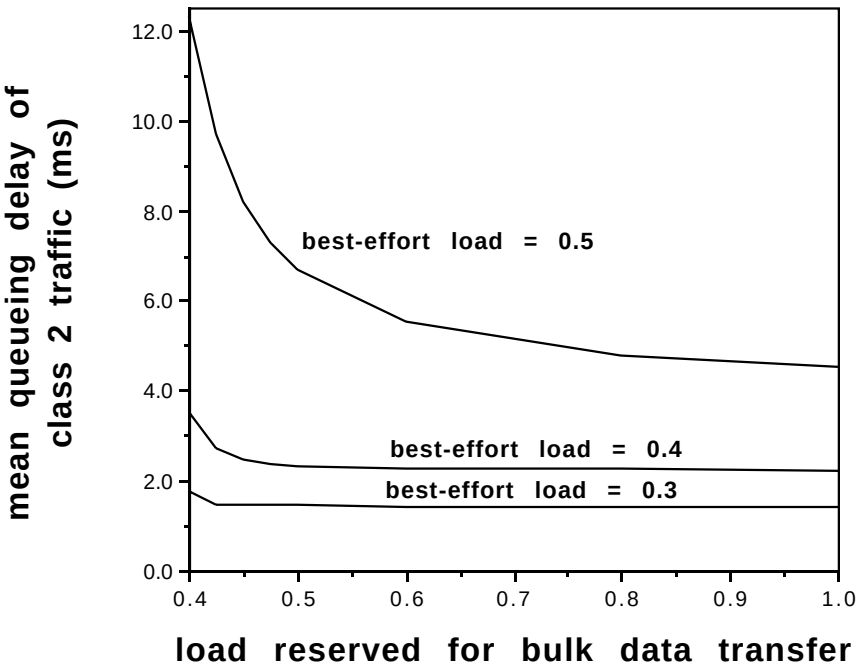

Figure 7: Mean delay Q_2 of class 2 traffic as a function of load ρ_p allocated for bulk data transfers, $\beta = 500$ kb, $\rho_2 = .3$, .4, and .5.

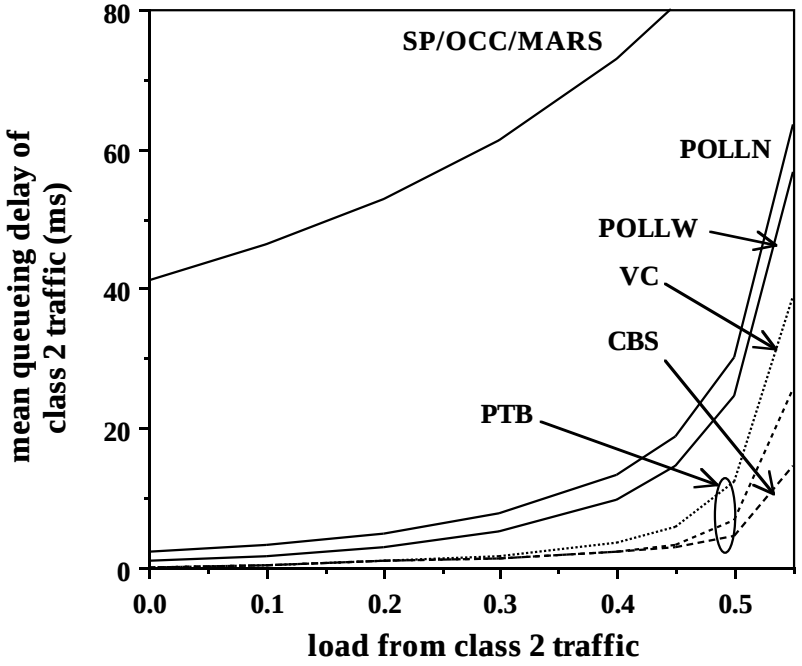

Figure 8: Mean delay Q_2 of class 2 traffic as a function of load ρ_2 from class 2 traffic with bulk data transfer class 1 traffic, $\beta = 500$ kb, $\rho_1 = .4$, .5, and 1.

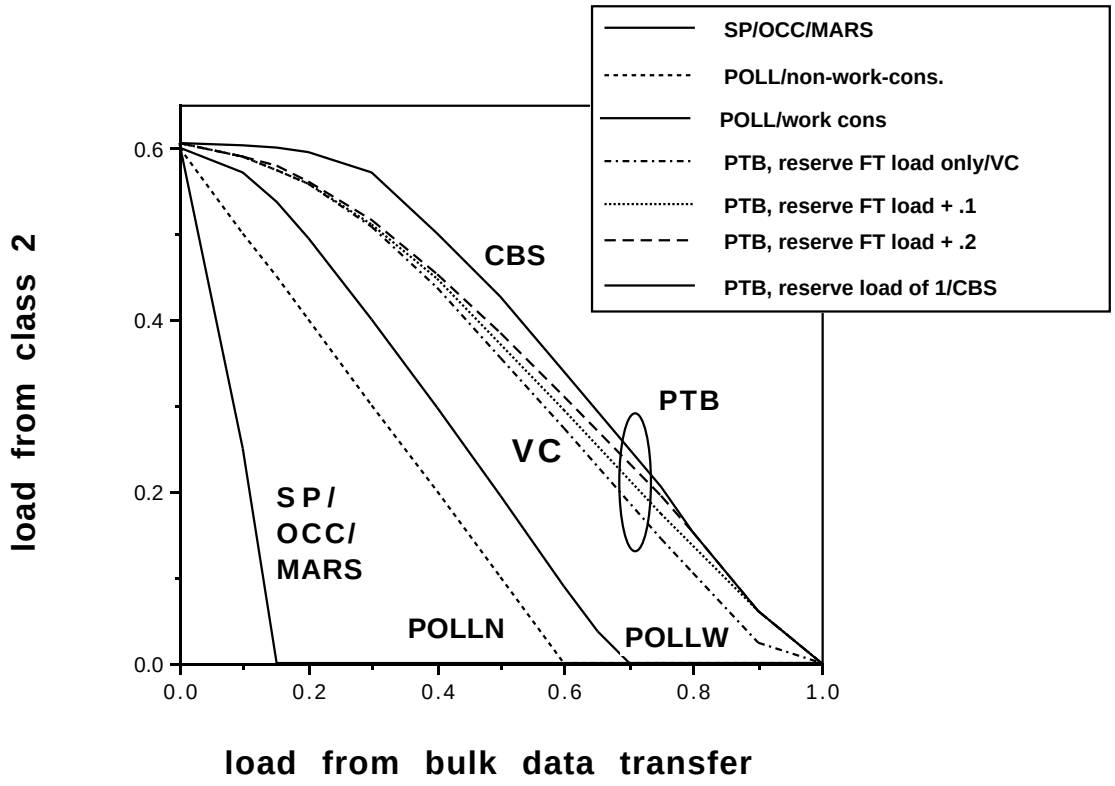

Figure 9: Schedulable regions: Load from class 2 traffic versus load from class 1 (Bulk File Transfer) traffic, Maximum delay for file transfer is 500 ms, $Q_2 \leq 5$ ms, $\beta = 500$ kb.