Scalable QoS Provision Through Buffer Management

R. Guérin, S. Kamat, V. Peris, and R. Rajan
IBM, T.J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598, USA
{guerin,sanjay,vperis,raju}@watson.ibm.com

Abstract

In recent years, a number of link scheduling algorithms have been proposed that greatly improve upon traditional FIFO scheduling in being able to assure rate and delay bounds for individual sessions. However, they cannot be easily deployed in a backbone environment with thousands of sessions, as their complexity increases with the number of sessions. In this paper, we propose and analyze an approach that uses a simple buffer management scheme to provide rate guarantees to individual flows (or to a set of flows) multiplexed into a common FIFO queue. We establish the buffer allocation requirements to achieve these rate guarantees and study the trade-off between the achievable link utilization and the buffer size required with the proposed scheme. The aspect of fair access to excess bandwidth is also addressed, and its mapping onto a buffer allocation rule is investigated. Numerical examples are provided that illustrate the performance of the proposed schemes. Finally, a scalable architecture for QoS provisioning is presented that integrates the proposed buffer management scheme with WFQ scheduling that uses a small number of queues.

Keywords: Buffer Management, Rate Guarantees, Scheduling, Sharing, Fairness

1 Introduction

Provision of service *guarantees*, especially rate guarantees, is becoming increasingly important in packet networks, and in particular the Internet. This is caused by both the heterogeneity of requirements from new applications, and the growing commercialization of the Internet. For example, the latter often translates into the specification of "Service Level Agreements," that define contracts, e.g., rate guarantees, between users and the network. The introduction of such service guarantees means, that contrary to the current *best-effort* networks which treat all flows equally, the network now needs to differentiate between flows. Support for such differentiation requires that the network control the amount of resources that each flow or set of flows is allowed to consume. The network resources whose consumption is to be controlled, consist primarily of buffers and link bandwidth, with buffer management and scheduling being the associated mechanisms.

The mechanisms used to provide such control do, however, come at a price, which has two main components: the storage and processing of the state information associated with service guarantees; and the cost of making the per packet admission and transmission decisions required to enforce service guarantees. These two dimensions are obviously not independent, but the more significant one appears to be the per packet cost. This is primarily because it directly impacts the speed and scalability of the associated mechanisms. In particular, as link speeds keep increasing, the "processing" time available for each packet decreases in proportion. This is particularly significant as high speed mechanisms are often implemented in hardware, and designed for worst case operation, i.e., the smallest possible packet size (for a 32-byte packet, this corresponds to about 107 nanoseconds on an OC-48 link).

In that context, packet scheduling costs are usually of a greater concern than those associated with buffer management. This is because with most buffer management schemes, the decision to admit or drop an incoming packet can be made based on a fixed amount of state information. Specifically, this usually consists of some global state information, e.g., the total buffer content, as well as additional state information specific to the flow to which the packet belongs, e.g., the number of packets the flow currently has in the buffer. For example, this is true for threshold based mechanisms [2], schemes such as Early Packet Discard (EPD) [7, 9], Random Early Discard (RED) [3], and Fair RED (FRED) [5].

Scheduling decisions, on the other hand, require both flow specific state information, e.g., the last transmission time of a packet from the flow, and operations involving *all* the other flows currently contending for access to the link. The latter is typically in the form of insertion and deletion operations in a sorted list of packets waiting for transmission. For example, in the case of algorithms such as Weighted Fair Queuing (WFQ) [6] or rate controlled Earliest Deadline First (EDF) [4], the sorted list consists of departure deadlines for packets from each active flow, where the departure deadline for a flow is computed so as to ensure specific rate and/or delay guarantees. Reliance on a sorting operation that grows with the number of flows (or service guarantees) can be a major impediment to scalability as speed increases. As a result, it is desirable to devise approaches that limit this exposure, even if at the cost of some decrease in performance guarantees or increase in the cost of other system components that are less critical for scalability purposes.

One possible direction is to lower the cost of sorting by allowing some coarsening of the information to be sorted. This is the approach of [8], which achieves a reduction from $\log N$ to $\log\log N$ in complexity, where N is the number of flows to be sorted. Another direction is to avoid reliance on sorting altogether as in the

Rotating Priority Queue (RPQ) proposal of [10]. This is the direction we pursue in this paper, where we take it to its extreme configuration of limiting scheduling support to that of a simple FIFO queue. In that context, we propose a scheme that provides rate guarantees to individual flows (or set of flows) by relying solely on buffer management mechanisms, i.e., packet admission decisions.

As mentioned earlier, buffer management operations typically require only a constant amount of processing and state information, compared to the sorting operations associated with a WFQ-like scheduler. Furthermore, scheduling seeks to provide rate guarantees to flows by controlling the transmission opportunities that each individual flow gets. However, there is little benefit in guaranteeing transmission opportunities to a flow, if it has no packets waiting because another misbehaving flow is occupying the entire buffer space. Thus, buffer management is required independently of any scheduling, if rate guarantees are to be provided. As a result, an approach that can provide rate guarantees by relying primarily on simple buffer management is attractive as it removes much of the complexity associated with scheduling.

As with the schemes of [8] and [10], there is obviously some "penalty" associated with the simplification of providing rate guarantees only through buffer management. As we shall see in Section 2, this penalty is primarily in terms of looser delay guarantees to individual flows and an increase in the amount of buffer space required to achieve a given link utilization.

Tight delay guarantees are clearly important to some real-time applications. However, in the environments with which this paper is concerned, namely, very high-speed links, even the worst case delays are likely to be sufficiently small and tolerable to most real-time applications. For example, the worst case delay caused by a 1MByte buffer feeding an OC-48 link (2.4Gbits/sec) is less than 3.5msec. In general, scheduling mechanisms that can provide tight delay guarantees are most appropriate on lower speed links, where not only queuing delays can be significant, but also scalability constraints are less of an issue. As a result, we believe that trading-off some control on delay guarantees for a simpler implementation and better scalability represents a reasonable design choice. Similarly, the need for larger buffers does translate into additional cost. However, those costs are containable, e.g., the price of memory has been regularly decreasing, and furthermore the scheme we propose offers some flexibility in shifting cost between buffer and bandwidth.

In the rest of this paper, we describe further the scheme we propose for ensuring rate guarantees, and identify properties of interest. Section 2 establishes the basic results relating buffer allocation and rate guarantees. In particular, it provides an explicit expression linking the amount of buffer allocated for a flow, to the rate it is guaranteed to receive. The result is given for both constant rate and bursty flows. Section 3 investigates the trade-offs involved when providing rate guarantees by relying solely on buffer management. This includes the impact on conformant and non-conformant flows of lowering the buffer size below what is needed to ensure rate guarantees. In addition, even when buffer sizes are sufficient to ensure the rate guarantees of conformant flows, providing efficient and fair access to excess resources is also of interest. Both aspects are investigated, and comparisons to what can be achieved using more sophisticated scheduling mechanisms, e.g., WFQ, are also provided. Based on the understanding obtained in Section 3, Section 4 investigates the potential benefits of hybrid schemes, that combine limited scheduling with the buffer management based approach of the paper. The section reviews some of the basic design choices of such combinations, and explores their potential benefits. Finally, Section 5 briefly summarizes the results of the paper.

2 Rate Guarantees for FIFO schedulers

Consider a number of flows being multiplexed onto a link using a simple FIFO scheduling policy. It is well known that with such a scheduling policy, misbehaving or aggressive flows can easily starve compliant flows. In other words, FIFO scheduling does not, by itself, provide sufficient isolation between flows. The problem we shall address in this section is that of controlling arrivals into the buffer, in order to ensure that all flows receive their share of link bandwidth even if some flows misbehave. In this section, we shall consider a very simple buffer management policy – that of *logically* partitioning the entire buffer into portions that may be considered reserved for particular flows. The partitioning is called *logical* because it is enforced by assigning a specific buffer occupancy threshold to each individual flow rather than by physically allocating buffer regions to flows. A packet belonging to a flow is admitted if it would not raise the flow's buffer occupancy beyond its assigned threshold. It is dropped otherwise.

Clearly, enforcing such a policy requires only a constant number of operations, irrespective of the number of flows involved. We first address the question of reserved buffer allocation, i.e., how to compute buffer occupancy thresholds for flows given their traffic profiles in order to ensure lossless service to each flow with no assumptions on the behavior of other flows. The answer to this question will immediately yield an admission control policy and a corresponding schedulability region. We then go on to compare the price we pay in terms of buffer requirements of this FIFO scheduling combined with a simple buffer management policy to that of a more sophisticated WFQ-like scheduling mechanism.

2.1 Rate guarantees based on peak rates

First, let us consider the case of two flows sharing a finite buffer of size B, and being multiplexed onto a link of capacity R using a FIFO scheduler. Flow 1 has peak rate ρ_1, while flow 2 is potentially aggressive, and could swamp the first flow if its arrival into the buffer is unregulated. We address the problem of logically partitioning the buffer size B into two portions, B_1 and B_2, that correspond to the maximum occupancy levels allowed for flows 1 and 2, respectively, so as to ensure that flow 1 never loses a packet.

Intuitively, it seems clear that flow 1's share of the buffer should be at least as large as its share of the bandwidth, i.e.,

$$\frac{B_1}{B} \geq \frac{\rho_1}{R}.$$

We will assume a fluid model of flows to demonstrate the correctness of our results. In the fluid model, each flow is comprised of infinitesimal bits that are served on a FIFO basis. At any time instant t, let $Q_1(t)$ and $Q_2(t)$ denote the respective buffer occupancy levels of flows 1 and 2. Also, let $A_1(t)$ and $A_2(t)$ denote their corresponding cumulative volume of traffic admitted into the buffer by time t.

Suppose that $B_1 = \frac{B\rho_1}{R}$, and $B_2 = B - B_1$. Initially, the buffer is empty. Let $u > 0$ be the first time at which flow 1 loses packets, so that at time u we must have, $Q_1(u) = B_1$. Consider then the bit that is the "oldest" one among flow 1's bits in the buffer at time u. This bit must have arrived at some earlier time instant v, such that $A_1(v) = A_1(u) - B_1$. On arrival, this bit sees sees $Q_1(v)$ bits of flow 1 and $Q_2(v)$ bits of flow 2 already in the buffer. As we assumed that flow 1 first experiences bit losses at time u, we have

$$Q_1(v) < B_1 = \frac{B\rho_1}{R} \tag{1}$$

Under the FIFO discipline, the bit arriving at time v will not spend more than $(Q_1(v)+Q_2(v))/R$ time in the queue. This implies that

$$u - v \leq (Q_1(v) + Q_2(v))/R.$$

Furthermore, arrivals from flow 1 between times v and u are bounded above by $\rho_1 \cdot (u-v)$. Consequently,

$$\begin{aligned} Q_1(u) &\leq \rho_1 \cdot (u-v) \\ &\leq \rho_1 \cdot \frac{Q_1(v)+Q_2(v)}{R} \\ &< \rho_1 \cdot \frac{B_1 + (B - B_1)}{R} \\ &= \frac{B\rho_1}{R}. \end{aligned}$$

This shows that the buffer occupancy of flow 1 never exceeds $B\rho_1/R$.

The main result of this buffer management approach can then be summarized in the following proposition.

Proposition 1 *Consider N flows admitted into a common buffer that is served by a FIFO scheduler with service rate R. It is given that flow i requires a guaranteed service rate ρ_i. If the flow is peak-rate conformant, a buffer occupancy threshold of $B\rho_i/R$ is sufficient to guarantee lossless service.*

Proof: The proof for flow i simply follows from the earlier discussion of the case of 2 competing flows by considering the traffic of all flows other than flow i as a single virtual flow.

Remark 1 *While the above proposition assures us that conformant flows do not lose packets, it is important to look at losses suffered by non-conformant flows, as well. In this context, we assert that if a flow exceeds its negotiated peak rate, then it will not be penalized excessively, i.e., it will have more bits delivered (up to any time) than had it been a lower volume[1] conformant flow. To see this, imagine coloring all conformant bits of a flow green and non-conformant bits red. If we pretend that green bits always have strict priority over red bits, then the conformant portion of the flow does not "see" the non-conformant portion, and hence never suffers loss. We can equivalently carry out an accounting stratagem of interchanging the colors of an arriving green bit for that of the earliest red bit in the buffer. This shows that at least as many bits get through as there are conformant bits.*

Next, we show through an example, that assigning buffer thresholds in proportion to rates is not only sufficient but also necessary.

Example 1 *Consider two flows, the first conformant to a peak reservation of ρ_1 and the second greedy. As before we assume a FIFO scheduler and a total buffer size of B, of which flow 1 can occupy at most $B_1 = \frac{B\rho_1}{R}$, while flow 2 is entitled to the rest, i.e., $B_2 = B - B_1$. The cumulative arrival process of the first flow into its buffer is given by $A_1(t) = \rho_1 \cdot t$. The greedy nature of the second flow means that its arrival process is such that $Q_2(t) = B_2$, for all $t \geq 0$.*

We examine the dynamics of this system at a sequence of times $t_0(=0), t_1, t_2, \ldots$. These are successive times at which the buffer content of flow 2 "clears", i.e., the last bit of flow 2 to arrive into buffer at time t_0 leaves at time t_1, and so on.

[1] A flow with arrival process A is said to be lower volume than another with arrival process $\hat{A}$ if $A(t-s) \leq \hat{A}(t-s)$ for any times s, t, with $t \geq s$.

Let us denote by R_i^1 and R_i^2 the service rates received by flows 1 and 2 respectively during the interval between t_{i-1} and t_i.

First, it is apparent that between the times t_0 and t_1, flow 1 receives no service, i.e., $R_1^1 = 0$, and flow 2 receives service at the link rate, i.e., $R_1^2 = R$. This is because of the FIFO service discipline and our assumption, that at time t_0 we start out with no bits of flow 1 in the buffer and flow 2 occupying its maximum threshold of $B2$. Thus, $t_1 = B_2/R$ is the first time instance where the buffer contains some bits of both flows that compete for service having arrived simultaneously at an earlier time.

Also, note that flow 2 seeks to greedily always occupy its maximum allowed buffer share. However, it can arrive into the buffer only at the rate at which it is being served. Thus, during the first interval (t_0, t_1), it replenishes itself in the buffer at rate R. However, in the second interval (t_1, t_2), it replenishes itself at a lower rate, since it shares the link capacity with some of the flow 1 traffic that arrived in the previous interval.

Now, at time t_1, $Q_1(t_1) = \rho_1 B_2/R$. Moreover, bits of flows 1 and 2 are interspersed with respect to the order in which they should receive service. The last bit of flow 2 in the buffer at time t_1 will depart after $(Q_1(t_1) + Q_2(t_1))/R$ time, i.e.,

$$t_2 = t_1 + \frac{Q_1(t_1)}{R} + \frac{Q_2(t_1)}{R} = t_1 + \frac{\rho_1(t_1 - t_0)}{R} + \frac{B_2}{R}.$$

The relative rates of service of the two flows after time t_1 is then in proportion to their rates of arrival into the buffer, i.e., bits of flow 1 will be drained at the rate $R_2^1 := \frac{\rho_1}{\rho_1+R}R$, while the remaining rate of the server $R_2^2 := \frac{R}{\rho_1+R}R$ will be dedicated to the second flow. Note that $R_2^1 < \rho_1$, so that even after time t_1, flow 1 is not receiving its guaranteed rate ρ_1. However, as we shall see, flow 1 asymptotically achieves is guaranteed rate, and does so without losing any bit, assuming the above buffer allocation.

In general, flow 1 receives service at the rate R_i^1, and flow 2 receives rate R_i^2 between times t_{i-1} and t_i. The buffer occupancy of flow 1 at time t_i is $\rho_1 \cdot (t_i - t_{i-1})$, while that of flow 2 is always B_2. Thus,

$$t_{i+1} = t_i + \frac{\rho_1 \cdot (t_i - t_{i-1})}{R} + \frac{B_2}{R}.$$

If we set $l_i := t_i - t_{i-1}$, the above may be written as

$$l_{i+1} = \frac{\rho_1}{R} l_i + \frac{B_2}{R}.$$

Moreover,

$$R_i^2 = B_2/l_i, \quad i = 1, 2, \ldots,$$

and

$$R_i^1 = R - R_i^2.$$

It is easy to see that,

$$\begin{aligned} lim_{i\to\infty} l_i &= \frac{B_2}{R - \rho_1} \\ lim_{i\to\infty} t_i &= \infty \\ lim_{i\to\infty} R_i^1 &= \rho_1 \\ lim_{i\to\infty} R_i^2 &= R - \rho_1 \end{aligned}$$

In other words, the flow 1 asymptotically fills its maximum allowed share of buffer, but obtains the long term rate ρ_1 despite the aggressiveness of the flow 2.

2.2 Rate guarantees based on token rates and burst sizes

Describing a flow through its peak rate alone results in an over-allocation of resources in the network. A popular alternative is to use a leaky bucket profile, i.e., describe the flow through a token rate ρ and burst size σ. We would like to use this richer knowledge of traffic burstiness to allocate reserved buffers for flows, so as to ensure lossless service for compliant traffic. The following proposition states the main result of this section.

Proposition 2 *Consider N flows, where flow i has a token rate ρ_i bits/second and a burst size σ_i, multiplexed on a constant rate link of rate R bits/second with a FIFO scheduler. If flow i is conformant, a reserved buffer allocation of $\sigma_i + B\rho_i/R$ is sufficient to guarantee lossless service to flow i.*

Note: The previous example can be easily extended to show that allocating less than $\sigma_i + B\rho_i/R$ to flow i will result in losses even if flow i remains conformant. This is achieved by having flow i transmit at rate ρ_i without transmitting its burst of σ_i until it fills the $B\rho_i/R$ portion of its buffer allocation[2], which it can according to the previous example, and then dumps its entire burst.

Proof : As before, let us consider two flows, one compliant and the other greedy, both being multiplexed onto a buffer of size B and served by a FIFO link scheduler, with a link capacity of R. Assume that $A_1(t)$, the function describing the arrival process of the first flow, is right continuous and (σ_1, ρ_1) constrained, i.e.,

$$A_1(t) - A_1(s) \leq \sigma_1 + \rho_1 \cdot (t - s) \quad 0 \leq s \leq t. \tag{2}$$

We claim that a buffer threshold of size $B_1 = \frac{B\rho_1}{R} + \sigma_1$ is sufficient to serve flow 1 in a lossless manner[3].

It is useful to associate a process called the *burst potential* process, with a flow. This process, denoted $\sigma_i(t)$ for flow i, is defined as

$$\sigma_i(t) := \inf_{s \leq t}\{A_i(s) + \rho_i(t - s) + \sigma_i\} - A_i(t). \tag{3}$$

This process describes the size of the token pool in the leaky bucket of the flow at a given time and thus captures the potential burstiness of the flow's arrival process at that instant. In other words, the RHS of the above equation denotes the maximum number of bits that could arrive for flow i instantaneously in a burst. In particular, from equations (2) and (3) it is easy to deduce that for any times $0 \leq v \leq u$,

$$\begin{aligned} A_1(u) + \sigma_1(u) &= \inf_{s \leq u}\{A_1(s) + \rho_1(u - s) + \sigma_1\} \\ &\leq \inf_{s \leq v}\{A_1(s) + \rho_1(u - s) + \sigma_1\} \\ &\leq \inf_{s \leq v}\{A_1(s) + \rho_1(u - v) + \rho_1(v - s) + \sigma_1\} \\ &\leq \rho_1(u - v) + \inf_{s \leq v}\{A_1(s) + \rho_1(v - s) + \sigma_1\} \\ &= A_1(v) + \sigma_1(v) + \rho_1(u - v). \end{aligned} \tag{4}$$

Define $M(t) := Q_1(t) + \sigma_1(t) - \sigma_1$.

Claim: With M defined as above, $-\sigma_1 \leq M(t) < \frac{B_2\rho_1}{R - \rho_1} := \hat{M}$ for all $t \geq 0$.

The first inequality is obvious. To show the second, note that $M(0) = 0$, as the initial burst potential of flow 1 is $\sigma_1(0) = \sigma_1$, and its buffer occupancy $Q_1(0)$ is zero.

We demonstrate the second inequality, i.e., $M(t) < \hat{M}$, for all t, by contradiction. Let us assume that this inequality is violated, so that there exists a smallest value of $u > 0$ such that $M(u) = \hat{M}$. (It is possible to show that $M(t)$ is continuous, and hence such a time does exist.) Look at the arrival time v of the 'oldest' bit of flow 1 in the buffer. It is easy to see that $v < u$, for if not, then all arrivals of flow 1 must have occurred at the instant u. But there can be no more than σ_1 such arrivals, which implies that $Q_1(u) + \sigma_1(u) \leq \sigma_1$, in turn implying that $M(u) < \hat{M}$, a contradiction. Thus, $v < u$, $M(v) < \hat{M}$, which implies

$$Q_1(v) < \sigma_1 - \sigma_1(v) + \frac{B_2\rho_1}{R - \rho_1}$$

Let $D_1(t)$ denote the cumulative departure process of the first flow. As there may be bits that arrived at time v that have not departed by time u, we have $A_1(v) \geq D_1(u)$. Let $\delta := A_1(v) - D_1(u)$. In other words, δ denotes the number of bits of flow 1 that arrived into the buffer at time v but have not yet departed at time u.

Now, we have

$$\begin{aligned} Q_1(u) + \sigma_1(u) &= A_1(u) - D_1(u) + \sigma_1(u) \\ &\leq A_1(v) - D_1(u) + \sigma_1(v) + \rho_1(u - v) \\ &\leq \delta + \sigma_1(v) + \rho_1(u - v). \end{aligned}$$

Now, $R \cdot (u - v) \leq Q_1(v) - \delta + Q_2(v)$. Consequently,

$$\begin{aligned} &Q_1(u) + \sigma_1(u) \\ &\quad\leq \delta + \sigma_1(v) + \frac{\rho_1}{R}(Q_1(v) - \delta + Q_2(v)) \\ &\quad< \delta + \sigma_1(v) + \frac{\rho_1}{R}\left(\sigma_1 - \sigma_1(v) + \frac{B_2\rho_1}{R - \rho_1} - \delta + B_2\right) \\ &\quad= \left(1 - \frac{\rho_1}{R}\right)(\delta + \sigma_1(v) - \sigma_1) + \sigma_1 + \frac{B_2\rho_1}{R - \rho_1}. \end{aligned}$$

To complete the proof, simply observe from equation (3), that for any $\epsilon > 0$,

$$A_1(v) + \sigma_1(v) \leq A_1(v - \epsilon) + \sigma_1 + \epsilon\rho_1$$

Taking limit as $\epsilon \to 0$, and rearranging terms

$$\sigma_1 \geq A_1(v) - A_1(v-) + \sigma_1(v) \geq \delta + \sigma_1(v).$$

Now, combining this inequality with the above one for $Q_1(u) + \sigma_1(u)$, we get

$$M(u) = Q_1(u) + \sigma_1(u) - \sigma_1 < \hat{M}$$

which contradicts our assumption that $\exists u, M(u) = \hat{M}$.

2.3 FIFO vs. WFQ: Worst Case Buffer Requirements

We now examine the tradeoff between a purely buffer based resource management scheme, and a scheduling mechanism such as WFQ, to provide lossless rate guarantees to individual flows. Suppose that we have N flows, where the i-th flow has a traffic profile of (σ_i, ρ_i). Suppose that all flows are conformant to their envelope. In this case, a WFQ scheduler would require a buffer of σ_i bits for flow i in order to provide lossless service[4]. Thus, the minimum total buffering requirement for a WFQ scheduler would be σ_i. Another way of saying this is that the set of N flows with envelopes

[2] Or rather until it gets arbitrarily close to it.

[3] Assume that $B \geq B_1$, i.e, $B \geq \frac{R}{R - \rho_1}\sigma_1$. We shall show later that a buffer of at least this size is required for lossless service.

[4] We ignore packetization in making this calculation.

$\{(\sigma_i, \rho_i)\}$ is WFQ-schedulable on a link of rate R bits/second with a fully partitioned buffer of size B bits if

$$R \geq \sum_{i=1}^{N} \rho_i, \tag{5}$$

$$B \geq \sum_{i=1}^{N} \sigma_i. \tag{6}$$

In other words, equation (5) specifies the bandwidth constraint that must be met to accept a new flow, while equation (6) gives the corresponding bandwidth constraint. Both constraints need to be satisfied for the flow to be accepted. In other words, if a new request is rejected because the constraint of equation (5) is violated, then the scheduler is deemed to be bandwidth limited. Conversely, it considered to be buffer limited if the new request is rejected because the constraint of equation (6) is not met.

On the other hand, to schedule the same set of flows using a FIFO scheduler combined with buffer management, not only do we need to satisfy the bandwidth constraint expressed in equation (5), but we also need to ensure that each flow has a reserved buffer share $B_i \geq B\rho_i/R + \sigma_i$. In other words, we must have

$$R \geq \sum_{i=1}^{N} \rho_i \tag{7}$$

$$B \geq \frac{B}{R}\sum_{i=1}^{N} \rho_i + \sum_{i=1}^{N} \sigma_i. \tag{8}$$

Equation (8) can be rewritten as follows

$$B \geq \frac{R}{R - \sum_{i=1}^{N} \rho_i} \sum_{i=1}^{N} \sigma_i. \tag{9}$$

Denoting the reserved link utilization by $u = \frac{\sum_{i=1}^{N} \rho_i}{R}$, this gives

$$B \geq \frac{1}{1-u} \sum_{i=1}^{N} \sigma_i. \tag{10}$$

Equation (10) points to the fact that in order to avoid being buffer limited in its call admission, the FIFO scheduler can require substantially more buffers than the WFQ scheduler. In particular, under FIFO scheduling, as the reserved link utilization goes to 1, the buffer requirements become unbounded. Note that the above derivation does not take the peak rate of the source into account. However, given a peak rate limit for the source a similar calculation can be carried out and identical results are obtained.

3 Tradeoffs between WFQ and FIFO

The previous section established basic results and properties on how to provide rate guarantees by relying solely on buffer management. Expressions were derived that relate buffer allocations to the corresponding rate guarantees. In addition, the amount of buffer needed to guarantee losslessness to a conformant flow was obtained and compared to what is required when a WFQ scheduler is used. It was shown that the greater simplicity of buffer management based scheme, came at the cost of potentially much higher buffer requirements, at least when the goal was to ensure losslessness.

Such a worst case comparison is certainly valuable and provides a useful benchmark. However, comparisons for more "practical" scenarios, e.g., when small losses are tolerated, are also of interest as are other performance measures such robustness to traffic fluctuations, sensitivity to buffer size, fairness in allocating idle bandwidth, etc. In this section, we perform such comparisons by means of simulation, and evolve a framework of performance measures to better characterize the behavior of our buffer management based approach. Based on those results, we also present simple modifications to the basic scheme, which allow us to "tune" the mechanism to achieve an operating point that balances different QoS measures.

3.1 Performance Measures Comparison Bases

A router is required to manage link bandwidth and buffers to achieve a reasonable tradeoff between different performance measures, which are often at odds. We consider three major objectives in evaluating link and buffer management schemes: (1) link utilization (2) delivery of rate guarantees to reserved flows (3) sharing of excess bandwidth. The other dimension of interest is the complexity and scalability of the solution used to achieve a given trade-off. As mentioned earlier, this was the primary motivation for our investigation of a buffer management based scheme for providing rate guarantees.

There is little need to emphasize that achieving high link utilization is a desirable goal. A straightforward approach to this problem is to admit as many packets as the buffer allows, and serve them in a work conserving manner. However, such a strategy is at odds with the second objective of providing a minimum level of service to reserved flows. Aggressive flows could swamp the buffers, and deprive conformant reserved flows of transmission opportunities. This suggests that we must, on occasion, prevent aggressive flows from occupying an excessive portion of the buffers as was embodied in the buffer management scheme described in the previous section. In carrying out such buffer management, one must also seek to apportion unreserved link capacity "fairly" among flows that can utilize it. There are many notions of fairness, and a flexible resource sharing scheme should be configurable to implement one that is suitable for the particular operational environment.

In order to evaluate our buffer management based scheme and the performance of several variants, we rely on several benchmarks. The first is a simple work-conserving FIFO scheduler with no buffer management. Such a scheme is commonly implemented in a best effort internet, and has the virtues of simplicity, scalability, as well as efficient utilization of link bandwidth. On the other hand, it is not capable of providing differentiated access to resources. Thus, neither are conformant flows protected, nor is excess capacity shared in a fair manner among competing flows. Our second benchmark is at the other end of the spectrum in terms of capabilities, and is based on a WFQ scheduler. Such schedulers, although relatively complex, are quite effective at providing rate guarantees to flows, even with relatively small buffers. Further, the WFQ automatically apportions excess bandwidth in proportion to rate reservations, and is, in this sense, fair to all flows. However, note that in order to deliver good QoS, it is important to also couple WFQ schedulers with effective buffer management schemes. As mentioned before, if access to the buffer is not regulated, it is easy for an aggressive flow to "capture" all spaces in the buffer, thus cornering all future transmission opportunities to itself. As a result, we also consider the performance of third benchmark, namely a WFQ scheduler combined with buffer management.

Flow	Peak rate (Mbits/s)	Avg rate (Mbits/s)	tkn bckt (KBytes)	tkn rate (Mbits/s)
0	16.0	2.0	50.0	2.0
1	16.0	2.0	50.0	2.0
2	16.0	2.0	50.0	2.0
3	40.0	8.0	100.0	8.0
4	40.0	8.0	100.0	8.0
5	40.0	8.0	100.0	8.0
6	40.0	4.0	50.0	0.4
7	40.0	4.0	50.0	0.4
8	40.0	16.0	50.0	2.0

Table 1: Traffic characteristics and reservation levels

In making comparisons between resource management schemes, a key aspect of interest is the sensitivity to the available total buffer size, especially when it is less than what is required to guarantee individual rates. In that context, we evaluate the performance of our buffer management scheme as the total buffer size varies, and compare it to the purely FIFO and WFQ schemes in terms of overall throughput, ability to provide rate guarantees to conformant flows, and ability to redistribute excess bandwidth. The aspect of efficient and fair redistribution of excess bandwidth is further investigated for buffer management schemes that are more aggressive in allowing non-conformant flows to access free buffers. The trade-off in this case is between higher link utilization, and the potential degradation of rate guarantees for conformant flows.

3.2 Buffer Thresholds

As indicated in Section 2.3, the buffer cost of providing rate guarantees through buffer management in a FIFO queue can be substantial when the (reserved) link utilization is high. It is, therefore, to be expected that in some cases the total amount of buffers available will be less than the sum of the buffer allocations required to provide strict rate guarantees to all flows. The scheme of Section 2 can be readily extended to handle this more general setting. This is done simply by mapping buffer allocations into thresholds, that determine when packets from a given flow should be dropped.

Specifically, flow i is assigned a threshold of size $\sigma_i + \rho_i B/R$, where σ_i and ρ_i are the token bucket size and token rate specified by the flow, B is the total number of buffers[5], and R the link rate. A packet is admitted if and only if there is room in the buffer and the queue size of its flow is less than the flow's threshold.

In the rest of this section, we study the performance of this threshold based buffer management with a FIFO scheduler, by comparing it to the benchmarks mentioned earlier. Specifically, we simulate and evaluate various performance measures for the following four schemes:

1. FIFO with threshold based buffer management.
2. WFQ with threshold based buffer management.
3. FIFO with no buffer management.
4. WFQ with no buffer management.

Simulation Setup: We simulate a link that is operating at 48Mb/s, which is a little over T3 capacity, over which a number of flows with various traffic patterns and rate guarantees are multiplexed. Each flow behaves as a Markov-modulated ON-OFF source and specifies a traffic profile (peak rate, token rate, and token bucket size).

[5] When the total number of buffers is larger than the sum of these thresholds, then all thresholds are appropriately scaled up so as to fully partition the buffer.

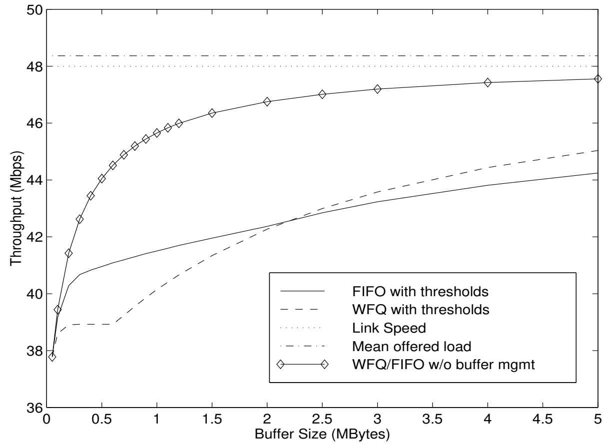

Figure 1: Aggregate throughput with threshold based buffer management.

When the source is in the ON state, the flow continuously transmits maximum size (500 bytes) packets at its peak rate. A subset of the flows are conformant and this is achieved by having their traffic regulated by a leaky bucket with parameters corresponding to their traffic profile.

The traffic characteristics of each of the flows and their corresponding rate guarantees (token rate) are listed in Table 1. Flows 0 through 5 are conformant to their profile, i.e., their reservation matches their traffic profile as ensured by the leaky bucket regulators. Flows 6 through 8 are unregulated. Their token rate only corresponds to the floor or minimum rate they are guaranteed, but as can be seen their average rate is much higher. In addition, their average burst size also exceeds their token bucket by a factor of 5. In the case of a WFQ scheduler, the token rate is used to determine the weight used for the flow. For both WFQ and FIFO schedulers, the thresholds used for buffer management purposes are computed as described earlier based on both the token bucket and the token rate. Note that these settings apply to both conformant and non-conformant flows. By summing the token rate values of Table 1, it can be seen that the aggregate reserved rate is 32.8 Mb/s, or about 68% of the link capacity. On the other hand, because non-conformant flows generate substantial traffic in excess of their profile, the mean offered load is a little over 100% of the output link's capacity.

We averaged the results over 5 simulation runs and found the 95% confidence intervals for throughput measurements to be less than 2% of the corresponding values. In the case of the packet loss measurements most of the 95% confidence intervals were within 10% of the corresponding results.

Figure 1 presents the throughputs achieved by the four schemes listed earlier. The total buffer size is varied from 500 KBytes to 5 MBytes. As expected, the FIFO scheduler with no buffer management achieves 90% utilization with barely 500 KBytes of buffers, while both WFQ and our FIFO scheme with threshold based buffer management require more that 6 times that amount to achieve the same utilization. When we compare the losses suffered by conformant flows, which is a measure of flow isolation, across the four scenarios, Figure 2 shows that the scheduling policies (FIFO and WFQ) without any buffer management perform identically. This is essentially a reflection of the fact that in both cases, aggressive non-conformant flows are preventing the smaller conformant flows from receiving transmission opportunities by filling up the buffers

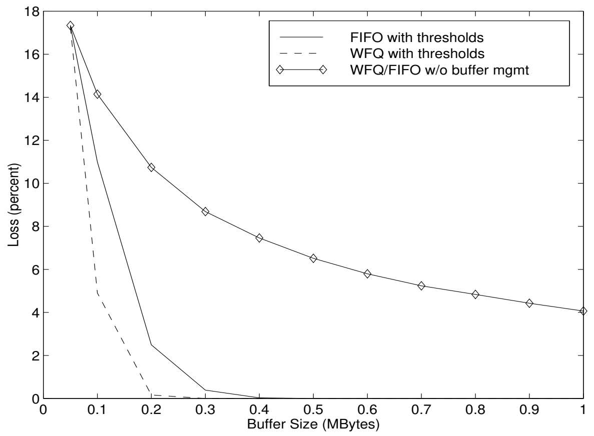

Figure 2: Loss for conformant flows with threshold based buffer management.

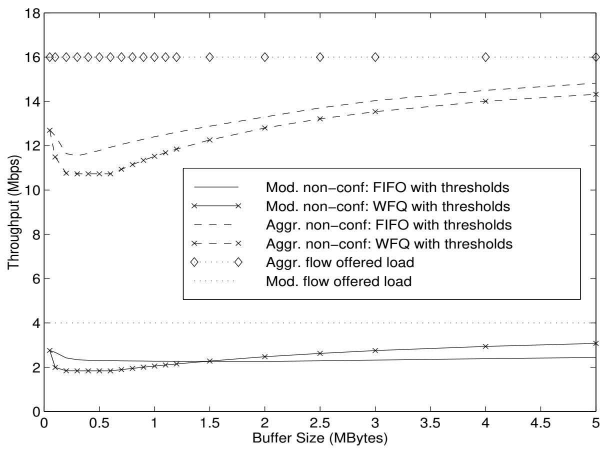

Figure 3: Throughput for non-conformant flows with threshold based buffer management.

whenever they burst. As these non-conformant sources are reasonably bursty, they result in periodic losses for the conformant traffic, but do not succeed in utilizing the link fully.

Note that as expected, policies which include buffer management are better at protecting flows. In addition, the threshold policy with FIFO scheduling is worse than WFQ with a threshold policy, in that the former requires 500 KBytes of buffer to achieve near 0 losses, while the latter merely requires 300 KBytes. This further confirms the trade-off between scheduling and buffer costs.

Finally, Figure 3 illustrates for the above scenario how the link bandwidth is shared by two non-conformant flows, flows 6 and 8, that differ in the amount of excess traffic they generate, with flow 8 generating substantially more excess traffic (see Table 1). The figure displays the expected behavior of WFQ with thresholds, where the two flows roughly share the excess bandwidth in the ratio of their reserved rates. rate reservations of the two flows, while none of the other policies consistently achieves such sharing. This includes the proposed FIFO with buffer management scheme, even when buffers are large enough to ensure rate guarantees. In the rest of this section, we investigate modifications to the buffer management scheme that aim at improving fairness in the sharing of excess

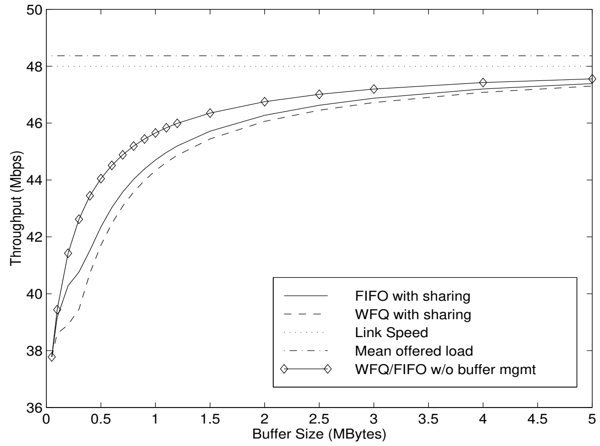

Figure 4: Aggregate throughput with Buffer Sharing.

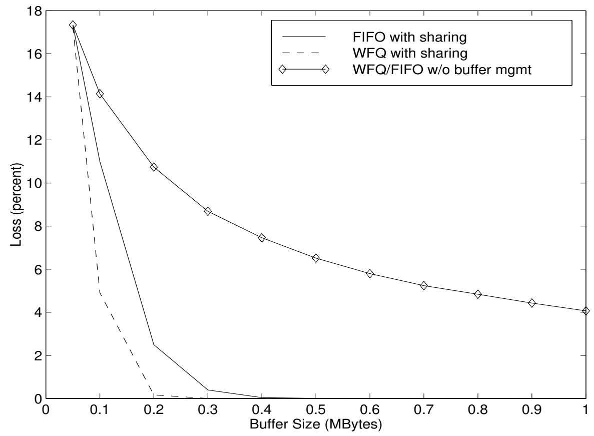

Figure 5: Loss for conformant flows in Buffer Sharing.

bandwidth, without affecting significantly the ability to provide rate guarantees.

3.3 Buffer Sharing

We now seek to improve the threshold based buffer management scheme so as to increase link utilization, and promote sharing of excess bandwidth. For that purpose, we define a buffer sharing scheme with thresholds, where the amount of buffers that need to be reserved for each flow is calculated identically as in the Fixed Partition case. The main difference with the Fixed Partition scheme is that we now allow active flows to access unused buffer space. In order to achieve fairness, we want the unused buffer space to be equally distributed among contending flows. However, in order to avoid any substantial impact to rate guarantees, we also reserve a certain *headroom* for flows that are below their threshold (and hence entitled to more buffer room). As a result, the buffers available for sharing are unused buffers from which the headroom has been subtracted. We denote those buffers as *holes*.

Access to buffers is controlled as follows. Whenever a packet arrives, we determine if the flow is below its threshold. If it is, we first attempt to use buffer space from the holes to accommodate the packet. If the space from the holes is insufficient, then buffer space

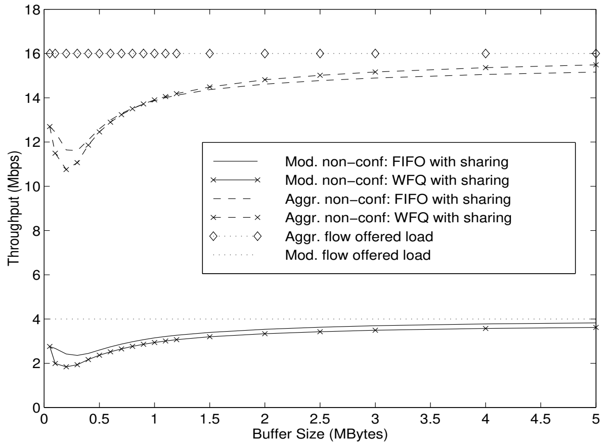

Figure 6: Throughput for non-conformant flows with Buffer Sharing.

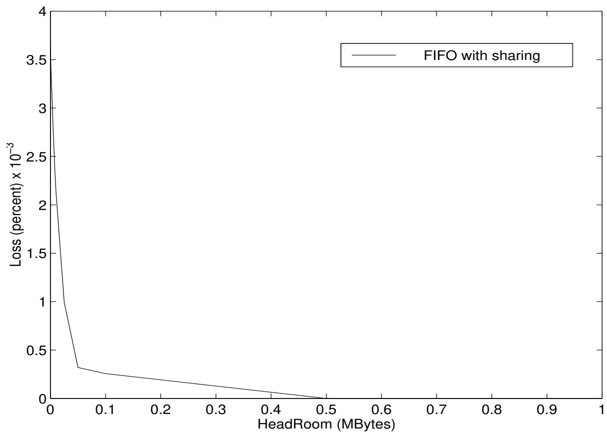

Figure 7: Effect of varying the headroom in terms of loss for conformant flows.

from the reserved headroom is used. If the available space is still insufficient, the packet is dropped. On the other hand, if the packet belongs to a flow which is above its threshold, the packet will be accommodated only if there is sufficient buffer space from the holes. Furthermore, in order to enforce some fairness in how holes are to be shared among flows, a packet is accepted only if the amount of buffer space occupied by the flow minus its reserved share, is less than the amount of remaining space in the holes. In other words, the amount of additional buffer space that a flow can grab, cannot exceed the amount of holes that are left. This sharing model is similar to the Dynamic Threshold scheme of [1]. The differences with the Dynamic Threshold scheme are the flow specific packet acceptance rules when a flow is below its threshold, and the use of a headroom to limit the amount of buffer space that can be shared.

When a packet departs, the holes and headroom counters are updated as follows:

```
headroom += packetlength;
holes += MAX(headroom - H, 0);
headroom = MIN(headroom, H);
```

This ensures that the amount of buffer space freed up by the packet departure is used preferentially to increment the headroom to a maximum of H, and only when this maximum value is reached is it applied to increasing the holes.

Simulation: We use a setting similar to the one previously described and compare FIFO scheduling coupled with the above buffer sharing scheme to WFQ with the same buffer sharing scheme. The goal is to investigate any improvement in sharing of excess bandwidth for the FIFO based scheme. Alternatively, we also want to assess the sensitivity of the scheme to the value H chosen for the headroom, and in particular how if affects the ability to provide rate guarantees to conformant flows.

In our simulation setup, we first choose a headroom of $H = 2$ MBytes. While studying link utilization, we recall our earlier baselines of FIFO/WFQ without any buffer management. As can be seen from comparing the throughput tradeoff in Figure 4 with Figure 1, we are quite successful in improving link utilization with the buffer sharing scheme. From Figure 5, it is apparent that this increase in throughput does not lead to worse protection for conformant flows. In Figure 6, we see that FIFO scheduling with buffer sharing based on thresholds successfully mimics WFQ in being able to distribute excess bandwidth in proportion to the reserved rate of the flow. The headroom provides a measure of protection for conformant flows, in that it acts as a reserved space for incoming flows that are within their thresholds. Increasing the headroom has the benefit of protecting conformant flows, while reducing the shared buffer space available for non-conformant flows. Figure 7 quantifies the protection offered to conformant flows as we vary the size of the headroom. In this instance, the size of the buffer is fixed at 1 MByte.

4 Extensions and Design Options

In this section, we discuss a possible extension to the schemes described in the previous sections. Specifically, at one extreme, we considered the use of a simple FIFO queue, with flow isolation and fairness provided solely through the use of buffer management. At the other end, we have considered an involved scheduling mechanism such as WFQ coupled with buffer management, to achieve the same objectives. Each of these solutions suffers from particular disadvantages. With the former, the penalty is paid in terms of increased buffer requirement while the main drawbacks of the latter involve issues of scalability and simplicity of implementation. A natural direction is then to explore what happens when the single FIFO queue is replaced by multiple FIFO queues, with a scheduler providing each queue its own rate guarantee. In each queue, the buffer management technique of the paper could then be used to further provide rate guarantees to individual flows. By keeping the number of such queues fixed and reasonably small, the overall architecture still remains scalable in terms of the total number of flows. We call such an architecture *hybrid,* and investigate its potential benefits.

There are some interesting questions and design parameters that arise in this architecture. For example, how many queues should we use to group the flows, how to assign flows to each queue, and how to determine the aggregate service rate to assign to each queue. In particular, the relation between increasing the number of queues and a potential decrease in the total buffer size required to provide a set of flows with rate guarantees, is not immediately apparent. Similarly, for a fixed number of queues, there may be a grouping of flows which results in the lowest possible total buffer requirement, and the identification of such a grouping and even its existence are again not simple issues. Clearly, such aspects need to be balanced with practical considerations that may prevent us from always enforcing an optimal flow grouping, i.e., as flows come and go, reassigning flows to different queues may not feasible. Nevertheless, gaining some basic understanding into these issues is of value as it may be enable us to devise practical schemes.

4.1 Rate allocations in a Hybrid System

In this sub-section, we provide some partial answers to the above questions. In particular, for a given number of queues and grouping of flows in each queue, we identify rate allocations to each queue that result in significant reduction in the possible total buffer requirement. While this does not necessarily result in an optimal grouping and rate allocation, it provides some insight into the kind of grouping that can lower the overall buffer size.

It is useful to introduce some additional notation here:

- R is the link rate as before.
- ρ and σ denote the sum of rates and bursts of all flows.
- k is the number of FIFO queues in which flows are classified.
- R_i denotes the rate at which the ith FIFO queue is served by a WFQ-like scheduler.
- $\hat{\rho}_i$ is the sum of the rate requirements of all flows in ith queue.
- $\hat{\sigma}_i$ is the sum of the burst requirements of all flows in the ith queue.
- B_i denotes the minimum amount of buffer space required for the ith queue. Assuming that more than one flow[6] is grouped into queue i, from equation (9) we have,

$$B_i = \frac{R_i \hat{\sigma}_i}{R_i - \hat{\rho}_i}. \tag{11}$$

- B_{hybrid} and B_{FIFO} denote the buffer requirements for the hybrid system using k FIFO queues and the earlier single FIFO queue, respectively.

$$B_{hybrid} = \sum_{i=1}^{k} \frac{R_i \hat{\sigma}_i}{R_i - \hat{\rho}_i} \tag{12}$$

and

$$B_{FIFO} = \frac{R\sigma}{R - \rho}. \tag{13}$$

Clearly, any rate assignment to the k queues should satisfy:

$$\sum_{i=1}^{k} R_i = R \text{ and } R_i \geq \hat{\rho}_i.$$

Thus, we may assign rates to individual queues as $R_i = \hat{\rho}_i + \alpha_i(R-\rho)$, where $0 < \alpha_i \leq 1$ and $\sum_{i=1}^{k} \alpha_i = 1$. This assigns each queue the minimum requested rate plus a fraction of the excess link capacity. Note that we assume here that the assignment of flows to queues is given, i.e., we are not attempting a joint optimization, and are simply trying to find the best rate assignment given an arbitrary partitioning of flows.

One plausible way to assign the excess available capacity may be to assign it in proportion to the total rate requirement of individual queues, i.e., $\alpha_i = \hat{\rho}_i/\rho$. However, it is easy to see that such an assignment does not reduce the overall buffer requirement, i.e., $B_{hybrid} = B_{FIFO}$. The allocation of excess available capacity must somehow take both the rate and burst requirements of different queues into account. Ideally, we would like to choose α_i's so that B_{hybrid} in equation (12) is minimized. The following proposition states how this can be achieved.

[6] For a single flow the buffer requirement is simply the burst size of the flow

Proposition 3 *In a hybrid system with k queues, the total buffer size required to provide rate guarantees to individual flows is minimized if link rate R is partitioned among queues as $R_i = \hat{\rho}_i + \alpha_i \cdot (R - \rho)$ where α_i is chosen as*

$$\alpha_i = \frac{\sqrt{\hat{\sigma}_i \hat{\rho}_i}}{\sum_{i=1}^{k} \sqrt{\hat{\sigma}_i \hat{\rho}_i}}. \tag{14}$$

Proof: Substituting for R_i in equation (12), the expression for the buffer requirement of the hybrid system is as follows

$$\begin{aligned} B_{hybrid} &= \sum_{i=1}^{k} \frac{\hat{\sigma}_i \hat{\rho}_i + \hat{\sigma}_i \alpha_i \cdot (R-\rho)}{\alpha_i \cdot (R-\rho)} \\ &= \sigma + \frac{1}{(R-\rho)} \cdot \sum_{i=1}^{k} \frac{\hat{\sigma}_i \hat{\rho}_i}{\alpha_i}. \end{aligned}$$

Let $f(\alpha_1, \alpha_2, \ldots, \alpha_k)$ be defined as

$$f(\alpha_1, \alpha_2, \ldots, \alpha_k) = \sum_{i=1}^{k} \frac{\hat{\sigma}_i \hat{\rho}_i}{\alpha_i}. \tag{15}$$

It can be verified that B_{hybrid} is minimized by minimizing the function $f(\cdot)$, which attains its minimum at $(\alpha_1, \ldots, \alpha_k)$ defined by equation (14), by ascertaining that for arbitrary $\{\delta_i\}_{1 \leq i \leq k}$, such that $\alpha_i + \delta_i > 0$ and $\sum_{i=1}^{k} \delta_i = 0$, the difference $f(\alpha_1 + \delta_1, \alpha_2 + \delta_2, \ldots, \alpha_k + \delta_k) - f(\alpha_1, \alpha_2, \ldots, \alpha_k)$ is non-negative.

The following claim states by how much this specific rate assignment can reduce the overall buffer requirement of the hybrid system with respect to the single queue FIFO scheduling approach.

Claim: If in the hybrid scheme, rates to queues are assigned as

$$R_i = \hat{\rho}_i + \frac{\sqrt{\hat{\sigma}_i \hat{\rho}_i}}{S} \cdot (R - \rho), \tag{16}$$

where $S = \sum_{i=1}^{k} \sqrt{\hat{\sigma}_i \hat{\rho}_i}$, then the difference in buffer requirement between a single FIFO queue and the hybrid system with k queues is

$$B_{FIFO} - B_{hybrid} = \frac{\sum_{i,j=1}^{k} (\sqrt{\hat{\sigma}_i \hat{\rho}_j} - \sqrt{\hat{\sigma}_j \hat{\rho}_i})^2}{(R - \rho)}. \tag{17}$$

For the rate assignment of equation (16), an individual queue's buffer requirement is given by equation (11) which can be rewritten as

$$B_i = \hat{\sigma}_i + \frac{S\sqrt{\hat{\sigma}_i \hat{\rho}_i}}{R - \rho}. \tag{18}$$

By summing over all queues, we get the total buffer requirement as

$$B_{hybrid} = \sigma + \frac{S^2}{R - \rho}. \tag{19}$$

From equations (19) and (13) we obtain

$$B_{FIFO} - B_{hybrid} = \frac{R\sigma}{R-\rho} - \sigma - \frac{S^2}{R-\rho}$$

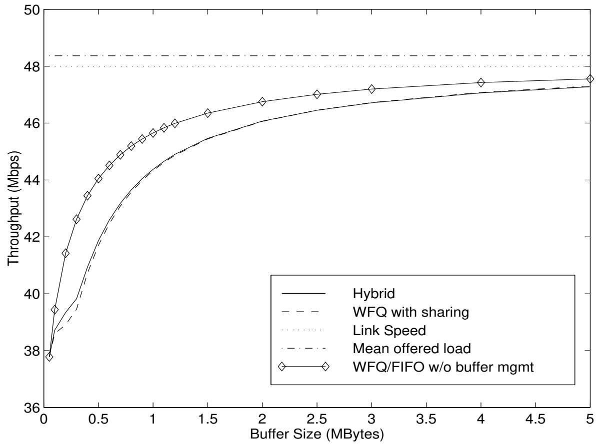

Figure 8: Hybrid System, Case 1: Aggregate throughput with Buffer Sharing.

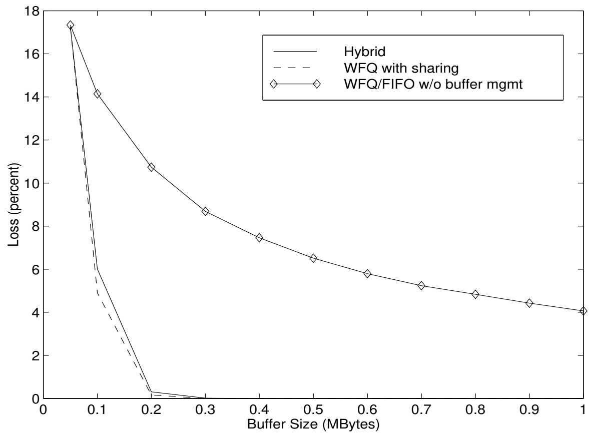

Figure 9: Hybrid System, Case 1: Loss for conformant flows with Buffer Sharing.

$$
\begin{aligned}
&= \frac{\sigma\rho - S^2}{R-\rho} \\
&= \frac{(\sum_{i=1}^{k}\hat{\sigma}_i)(\sum_{i=1}^{k}\hat{\rho}_i) - (\sum_{i=1}^{k}\sqrt{\hat{\sigma}_i\hat{\rho}_i})^2}{(R-\rho)} \\
&= \frac{\sum_{i,j=1}^{k}(\sqrt{\hat{\sigma}_i\hat{\rho}_j} - \sqrt{\hat{\sigma}_j\hat{\rho}_i})^2}{(R-\rho)}.
\end{aligned}
$$

Since the numerator in the above expression is a sum of non-negative terms, the difference $B_{FIFO} - B_{hybrid}$ is also non-negative. This result indicates that with proper rate assignment, one can potentially reduce the overall buffer requirement by splitting a set of flows served by one FIFO queue into more FIFO queues. In the extreme, we have one flow per queue, and the hybrid system reduces to a pure WFQ system. The choice of a given number of queues is primarily dictated by the implementation complexity that can be tolerated for the scheduler, i.e., the size of the sorted list that needs to be updated after each packet transmission. Once the number of queues has been fixed, the result also suggests that grouping flows such that one queue has significantly lower rate and burst requirements compared to another is beneficial. While it is clearly not

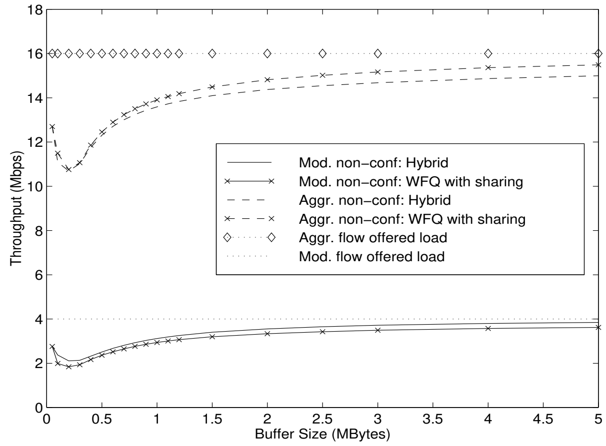

Figure 10: Hybrid System, Case 1: Throughput for non-conformant flows with Buffer Sharing.

practical to continuously shuffle flows between queues to maintain this property, it suggests a potentially useful broad classification of flows. For example, low bandwidth and burstiness IP telephony flows could be assigned to one queue, while higher bandwidth and burstiness video on demand streams would be mapped onto another queue.

4.2 Hybrid Systems: Performance Tradeoffs

To conclude this section, we consider two examples of hybrid systems and compare their behavior. For both examples, we again assume the 48 Mbits/sec link speed of the previous section.

Case 1: 9 Flows

In this scenario, we consider the performance of the hybrid system for the same simulation example with 9 flows considered in the previous section. In this case, we group the flows into 3 queues. The 3 small conformant flows 0, 1 and 2 are grouped into queue 1, the next 3 large conformant flows 3, 4 and 5 into queue 2, and the three non-conformant flows 6, 7 and 8 into queue 3. Having grouped flows in this manner, we compute the weighting factors α_i as given by equation (14). These determine how excess bandwidth is to be allocated to queue i, and its minimum buffer requirement B_i^{min} based on equation (18). Given a buffer of size B, we then partition the buffer amongst the queues in proportion to their minimum buffer requirement, i.e., queue i gets a buffer of size $B_i = B\frac{B_i^{min}}{\sum_{i=1}^{3} B_i^{min}}$. An individual flow j within queue i is then allocated a threshold of $\sigma_j + \frac{\rho_j}{\hat{\rho}_i} \cdot B_i$, where as before $\hat{\rho}_i$ is the rate allocated to queue i.

Figures 8, 9, and 10 illustrate the performance tradeoffs of using the hybrid system. It is clear from these simulations that the performance of the 3-queue hybrid system is very close to that of WFQ with buffer sharing which maintains separate queues for each flow. However, this is not entirely unexpected, since we only have 3 flows in each queue.

Case 2: 30 Flows

In order to explore a more realistic example, we consider next a hybrid system with 30 flows grouped in three queues as shown in Table 2.

In this system, the first 10 flows (0–9) are *conformant* to their requested token rates and token bucket sizes. The next 10 are *mod-*

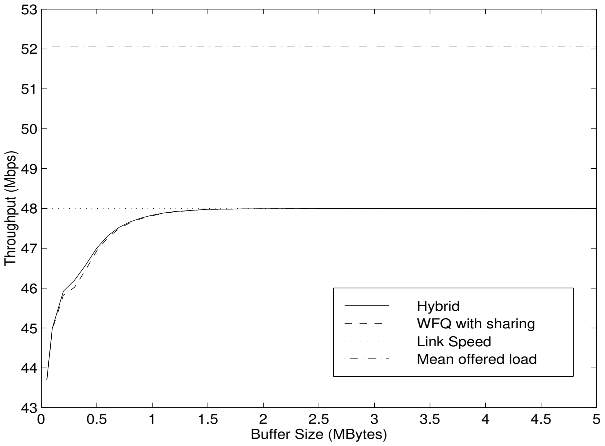

Figure 11: Hybrid System, Case 2: Aggregate throughput with Buffer Sharing.

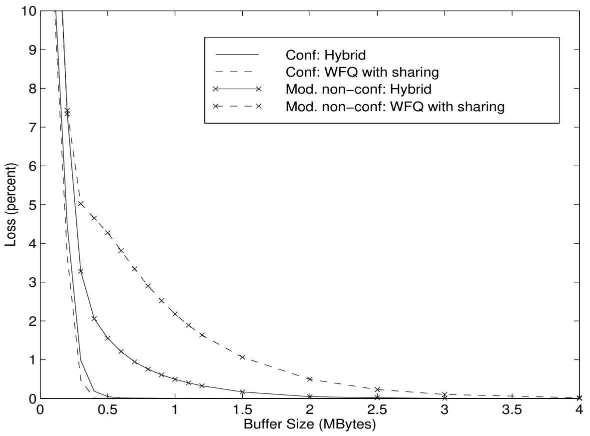

Figure 12: Hybrid System, Case 2: Loss for conformant and moderately conformant flows with Buffer Sharing.

erately non-conformant, in that their mean rate and average burst size conform to their specified token parameters. However, they are Markov modulated ON-OFF sources with which are not reshaped by a token bucket, and their traffic can, therefore, temporarily exceed their traffic profile. The last 10 flows are aggressive, in the sense that their actual arrival rates are over 8 times their requested reservation rates, and in addition their average burst size is 500KBytes which is way in excess of their token bucket.

The results are shown in Figures 11, 12, and 13. It is clear from these simulations that the performance of the hybrid system remains close to that of WFQ with buffer sharing, even for this larger number of flows.

5 Conclusion and Future Work

In this paper, we have established how rate guarantees can be provided by simply using buffer management. Exact expressions were provided that associate rate guarantees with buffer allocation in a simple FIFO queue. The efficiency of the scheme was investigated in terms of both the buffer size required to provide rate guarantees, and the ability of the scheme to enforce guarantees while allowing

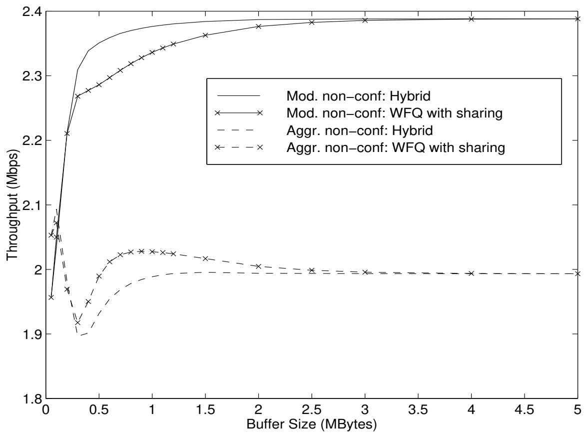

Figure 13: Hybrid System, Case 2: Throughput for non-conformant flows with Buffer Sharing.

Flow	Peak rate (Mbits/s)	Avg rate (Mbits/s)	tkn bckt (KBytes)	tkn rate (Mbits/s)
0-9	8.0	0.6	15.0	0.6
10-19	24.0	2.4	30.0	2.4
20-29	8.0	2.4	35.0	0.3

Table 2: Case 2: Traffic characteristics and reservation levels

efficient sharing of idle resources. The performance of the scheme was compared to that of a scheduler based scheme, i.e., WFQ, and the associated trade-offs were identified. A hybrid scheme where the FIFO queue is replaced by a small number of queues served by a WFQ scheduler was also investigated, and some potential benefits of grouping flows into separate queues were identified.

This combination of buffer management and limited scheduling, appears capable of a broad range of trade-offs between efficiency and complexity. However, understanding and defining the best possible combinations is an area that requires additional work. Another aspect of interest is the ability to provide different bandwidth sharing models through simple modifications to the buffer management scheme. Specifically, buffer management can provide a single mechanism to both enforce rate guarantees and control sharing of idle bandwidth. In contrast, most scheduling mechanisms, e.g., WFQ, usually imply a specific sharing model which cannot be easily adjusted without affecting the scheduling mechanism itself.

For example, in Section 3, it was shown how going from complete buffer partitioning to an approach with greater flexibility in sharing free buffers, resulted in different allocation of excess bandwidth across flows. Specifically, when full partitioning was used, excess bandwidth was redistributed in proportion to each flow's reserved rate, while active flows received, in addition to their reserved rate, an equal share of the excess bandwidth when buffer sharing was allowed. This represents only one example from a wide range of options available for controlling bandwidth sharing through buffer management. For example, one could also envision allowing adaptive flows to share buffers with reserved flows, while non-adaptive ones would be prevented from doing so. This would provide adaptive flows with greater access to available bandwidth without impacting reservations, and without entirely shutting off non-adaptive flows from accessing idle resources. However, understanding fully how variations in buffer sharing translate into bandwidth sharing requires more work.

References

[1] A. K. Choudhury and E. L. Hahne. Dynamic queue length thresholds in a shared memory ATM switch. In *Proceedings of INFOCOM*, pages 679–687, San Francisco, CA, April 1996.

[2] I. Cidon, R. Guérin, and A. Khamisy. Protective buffer management policies. *IEEE/ACM Trans. Networking*, 2(3):240–246, June 1994.

[3] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Networking*, 1(4):397–413, August 1993.

[4] L. Georgiadis, R. Guérin, V. Peris, and K. N. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Trans. Networking*, 4(4):482–501, August 1996.

[5] D. Lin and R. Morris. Dynamics of random early detection. In *Proceedings of SIGCOMM*, pages 127–137, Sophia Antipolis, France, September 1997.

[6] A. K. J. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*. PhD thesis, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, February 1992. No. LIDS-TH-2089.

[7] A. Romanow and S. Floyd. Dynamics of TCP traffic over ATM networks. *IEEE J. Sel. Areas Commun.*, 13(4):633–641, May 1995.

[8] S. Suri, G. Varghese, and G. Chandranmenon. Leap forward virtual clock: A new fair queueing scheme with guaranteed dealy and throughput fairness. In *Proceedings of INFOCOM*, pages 558–566, Kobe, Japan, April 1997.

[9] J. Turner. Maintaining high throughput during overload in ATM switches. In *Proceedings of INFOCOM*, pages 287–295, San Francisco, CA, April 1996.

[10] D. Wrege and J. Liebeherr. A near-optimal packet scheduler for QoS networks. In *Proceedings of INFOCOM*, pages 577–585, Kobe, Japan, April 1997.