# Uniform versus Priority Dropping for Layered Video*

Sandeep Bajaj Lee Breslau Scott Shenker
{bajaj,breslau,shenker}@parc.xerox.com
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

**Abstract**

In this paper, we analyze the relative merits of uniform versus priority dropping for the transmission of layered video. We first present our original intuitions about these two approaches, and then investigate the issue more thoroughly through simulations and analysis in which we explicitly model the performance of layered video applications. We compare both their performance characteristics and incentive properties, and find that the performance benefit of priority dropping is smaller than we expected, while uniform dropping has worse incentive properties than we previously believed.

## 1 Introduction

A common question facing network designers is what functionality the Internet should offer, and whether to place that functionality in the interior of the network (in network routers) or at its edges (in hosts). When evaluating new network control mechanisms, it is not enough to merely consider network-centric criteria, such as the local effects on queue sizes at routers, or the end-to-end delays provided by a particular network architecture. Rather, because the ultimate purpose of the network is to support applications, one must evaluate the impact of the proposed mechanisms on application performance.[1] If the proposed mechanisms require significant additional complexity in the network infrastructure, their adoption should only be considered if they provide significant benefits to a large (or extremely important) segment of applications.[2]

For example, a priority dropping mechanism will increase the dropping probability of low priority packets while protecting high priority packets from frequent loss. However, in and of itself, the change in drop rates for different traffic does not indicate whether the control mechanism significantly improves the performance of applications. The desirability of drop priority depends crucially on its impact on application performance. That aspect of the issue, which has received comparatively little attention so far, is the focus of our paper.

This paper is devoted to the question of drop priority for the transmission of layered video in contexts where packet drops, rather than packet delays, are the primary determinant of application performance. In such cases, we ask: should the Internet retain the currently predominant *uniform* dropping mechanism, where all data packets are treated equally with respect to dropping, or should the Internet adopt a *priority* dropping mechanism where lower priority packets are dropped before higher priority ones? We specifically focus on the implications of these network dropping mechanisms for application performance.[3] To this end, we consider a class of layered video applications and introduce a simplified model of their performance; this model is extremely primitive, but nonetheless highlights several open questions about the characteristics of network video applications. Using both discrete event simulation and analytical models, we assess the impact of different network dropping schemes on application performance and on their incentive properties.

*This research was supported in part by the Advanced Research Projects Agency, monitored by Fort Huachuca under contracts DABT63-94-C-0073 and DABT63-96-C-0105. The views expressed here do not reflect the position or policy of the U.S. government.

[1] We will use the terms application performance and application utility interchangeably.

[2] Alternatively, one would consider adopting network designs that allowed a significantly higher level of link utilization without significant application performance degradation; in this case no single user is substantially better off in terms of application performance, but the increased utilization implies that there are many more such satisfied users. Of course, the desirability of any new network level mechanism depends on the level of mechanistic complexity introduced; however, we adhere to the philosophy that one should only consider adoption after the significance of the benefit is demonstrated, and so the consideration of implementation complexity comes after the analysis of the potential benefits. We only address the latter (the potential benefits) and not the former (the implementation complexity) in this paper.

[3] We do not discuss the mechanisms for implementing drop priority, such as whether these priority levels are indicated in packets in TOS bits, or signaled as part of the multicast join message.

Much to our chagrin, we find that many of our previously held opinions and assumptions about the performance of packet dropping algorithms and the incentives they provide to applications are either wrong, or are only true in the context of specific network conditions. In particular, we find that while priority dropping performs better in general than uniform dropping, the magnitude of the difference is smaller than we expected. At the same time, we find that uniform dropping has worse incentive properties than we thought.

We hasten to note that we do not offer specific conclusions about whether or not the Internet should offer drop priority for layered video applications. Our results are ambiguous on this point – there are some conditions where drop priority yields significant benefits, and other conditions where it doesn't– and there are many relevant issues (such as implementation complexity) that are outside of our ken. Instead of making a specific recommendation about priority dropping, our goal is to illustrate more generally the basic approach of considering application performance in conjunction with network control mechanisms.

In the next section we provide background about the question of network dropping algorithms that motivated this work, and identify three central questions to be addressed. In Section 3 we outline our general model of application utility, along with the simulation and theoretical frameworks used in this study. In Sections 4 and 5 we present the results of our investigations into the performance and incentive aspects of priority and uniform dropping. We conclude with a discussion in Section 6.

## 2 Background

A key challenge in sending video over the Internet is matching the transmission rate to the currently available bandwidth. One approach is to employ rate adaptive coding algorithms [2, 3] in which the parameters of the coding algorithm can be adjusted periodically to match the available network bandwidth. However, this strategy is problematic for multicast transmission [12, 14] since, in general, there is not a single bottleneck bandwidth available between the source and all receivers; some paths will have more available bandwidth than others. Choosing a single transmission rate (at any instance in time) will either unnecessarily constrain those receivers with higher bandwidth paths or congest the lower bandwidth paths leaving some receivers with high loss rates. In response, Deering [5] and others [9, 11, 15, 16] proposed using layered coding algorithms[4] to transmit video data, striping different layers across different IP multicast groups [4]. In this scheme, a receiver only subscribes to those groups for which sufficient network capacity exists. Thus, in a heterogeneous network, each receiver is able to adjust its level of subscription to receive an appropriate amount of traffic.

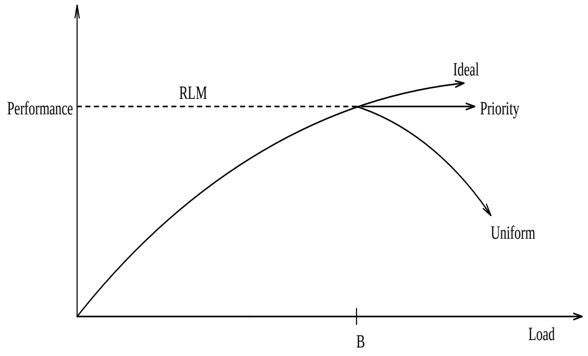


Figure 1: A representation of application performance as a function of load with perfectly smooth traffic. The maximal performances of priority and uniform dropping are the same, and RLM achieves this optimal level. The uppermost curve depicts the performance level achieved if the link had infinite bandwidth. The load level marked by $B$ denotes the bandwidth of the bottleneck link.

Such a layered encoding and transmission scheme lends itself to the use of priority dropping. Packets belonging to the base layer of a hierarchically encoded video stream can be marked as high priority while packets belonging to each successive enhancement layer can be marked as successively lower priority. During times of congestion, the network can preferentially drop the low priority packets, protecting the base layer from significant loss. Shifting loss away from more important and towards less important packets improves the picture quality. For this reason, priority dropping (for layered encodings) was considered by many in the Internet community to be an extremely promising approach for digital video.

However, in their recent work on Receiver-driven Layered Multicast [12] (RLM), McCanne *et al.* argue against this use of priority dropping.[5] Their argument is best illustrated by Figure 1 (based on a similar figure in their paper) which plots the application performance (which, in the case of digital video, is synonymous with picture quality) as a function of offered load for uniform and priority dropping when capacity is fixed. In addition, the uppermost curve shows the quality of the picture as a function of load with unlimited link capacity (and therefore no packet drops.) Underlying this figure is the implicit assumption that there is a bottleneck rate below which there is no dropping, and above which all excess packets are dropped. That is, the output rate equals the input rate whenever the input rate is below the bottleneck rate, and the output rate is equal to the bottleneck rate whenever the input rate is greater than the bottleneck rate. As the figure

[4] Layered coding algorithms, such as described in [9, 11], partition the encoded signal into several *layers*. The base layer encodes a fairly primitive rendering of the image, and higher layers encode increasingly finer enhancements to the image. Layered coding algorithms thus provide several different levels of encoding simultaneously. A receiver can choose how may layers to receive, and the more layers that are available to decode, the higher the resulting picture quality.

[5] While the introduction of this paper makes remarks about the inadvisability of priority dropping (as we outline below), we should note that in this paper RLM is also described as a way to cope with the current uniform dropping infrastructure (regardless of what one thinks of priority dropping). Moreover, the bulk of the paper is devoted to the design of the RLM algorithm itself – an admirable exercise in elegance and scalability – and the discussion of the relative merits of uniform and priority dropping is clearly not the central focus of the paper.

shows, up to the bottleneck load the application performance of the two dropping schemes is equal because no packets are dropped in either case. Beyond the bottleneck load, the performance of uniform dropping degrades because no additional packets are transmitted, and packets are dropped uniformly from all layers; the performance of priority dropping remains constant since only packets from the lower priority levels are dropped. Thus, as illustrated by Figure 1, uniform and priority dropping both attain the same maximal performance, and the performance curves only differ when excess load is offered.

As discussed by McCanne *et al.* [12], this behavior has implications for the incentives faced by users. Under uniform dropping, users maximizing their own performance would restrain their usage to the bottleneck rate. Priority dropping, on the other hand, does not provide any penalty for sending low priority packets that are dropped inside the network, and so users (at least based solely on application performance) would not mind sending faster than the bottleneck rate.[6] Thus, uniform dropping provides better performance-based incentives than priority dropping.

Priority dropping, however, has the advantage that the application need not determine the bottleneck rate precisely to achieve optimal performance. In contrast, to achieve the optimal performance with uniform dropping, the application must identify the bottleneck rate precisely because performance degrades if the transmission rate is higher (or lower) than the bottleneck rate.

RLM [12] is a host-based algorithm that responds to current network conditions (as measured by packet drops) to achieve the appropriate bandwidth level. Members leave a level (*i.e.*, leave the multicast group associated with that level) when their overall drop rate (across all levels) becomes too high; in addition, they periodically perform experiments by joining a level to test if the drop rate with the addition of the new level is sufficiently low. RLM's innovative design coordinates the activities among the various members of the multicast group so that these experiments don't interfere with each other. Under the conditions explored in [12], where the traffic is rather smooth (*i.e.*, CBR-like), RLM successfully achieves the bottleneck bandwidth, and does so in a scalable manner for large multicast groups. We have indicated this on Figure 1 by marking RLM's performance level at the peak of the priority and uniform dropping curves.

One can summarize (with appropriate additional caveats) the three basic tenets of the arguments against priority-dropping: (1) uniform dropping achieves (essentially) the same optimal level of performance as priority dropping (*i.e.*, the peaks of the curves are nearly the same), (2) RLM achieves (close to) optimal performance, and (3) uniform dropping provides incentives for users to reach the optimal operating point whereas priority dropping does not provide performance-based incentives to constrain usage. While these arguments certainly have an initial appeal, we must admit that after reading [12] we were not entirely convinced by them. In the spirit of full disclosure we now discuss the intuition that (mis)guided us as we embarked on our research program.

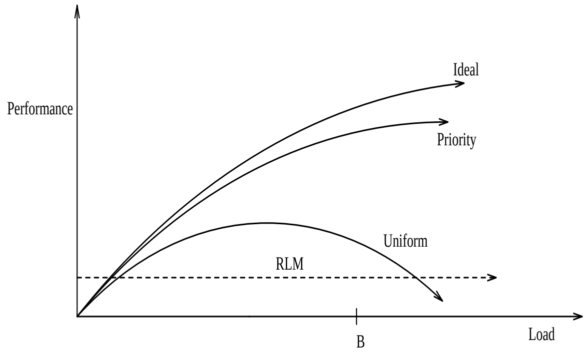


Figure 2: A representation of our intuition about application performance as a function of average load with bursty traffic. Priority dropping achieves a significantly higher maximal performance level than uniform dropping. Moreover, RLM's performance is significantly below the maximal performance achievable under uniform dropping. As before, the uppermost curve depicts the performance level achieved if the link had infinite bandwidth, and the load level marked by $B$ denotes the bandwidth of the bottleneck link. Note that since the x-axis represents average load and traffic is bursty, the performance of priority dropping can increase beyond the bottleneck rate.

The assumption underlying Figure 1 is that packet dropping occurs only when the bottleneck rate is exceeded, and then all excess packets (*i.e.*, all packets exceeding the bottleneck rate) are dropped. This is a reasonably good approximation when network traffic is quite smooth, like CBR traffic, but Internet traffic is not typically CBR-like. In fact, Internet traffic is characteristically quite bursty [8, 10], and bursty traffic results in packet drops even when long-term average transmission rates are well below bottleneck rates. With bursty traffic, there will be some packet drops at all significant utilization levels, thereby producing a performance gap between priority and uniform dropping even below the bottleneck rate. Consequently, our initial hypothesis when undertaking this work was that in the presence of bursty traffic (similar to what we envision is present in the Internet), priority dropping would significantly outperform uniform dropping. By this we mean the optimal performance achievable with priority dropping would be much higher than the highest performance achievable with uniform dropping; in essence, we imagined that with bursty traffic the performance curves would look more like Figure 2 than like Figure 1.

RLM uses packet drops as a signal of congestion, joining and leaving levels based on the current dropping rate. In this manner, RLM attempts to match its transmission rate to the bottleneck bandwidth. When traffic is bursty, however, the bandwidth available at

[6] The impact of users not constraining their usage of lower priority packets is not clear, nor can it be evaluated in the simple settings we investigate here. If there is only a single bottleneck in the network, then there are no ill-effects of sending lower priority packets that will be dropped at that bottleneck. Negative consequences only arise in more complicated settings where these destined-to-be-dropped packets congest one or more links upstream of the bottleneck. It is not clear how significant a phenomena this is.

the bottleneck (and perhaps even the location of the bottleneck itself) can fluctuate significantly. Moreover, these fluctuations occur on time scales much shorter than the response times of host-based mechanisms (which require at least the round trip time between the bottleneck and the receiver to respond). We assumed that any attempt to adjust, at the endpoint, to rapid fluctuations at the router was doomed to fail. Consequently, our initial hypothesis was that, under bursty conditions, adaptive mechanisms implemented at the endpoints would perform significantly worse than the optimal performance under uniform dropping, and certainly much worse than the performance achieved with priority dropping. As shown in Figure 2, our initial assumption was that RLM, or indeed any similar adaptive algorithm used with uniform dropping, would result in performance significantly below the peak of the uniform dropping curve.

Lastly, there is the issue of incentives. The performance curves in Figures 1 and 2 depict the performance of a single application given a fixed (smooth in Figure 1 and bursty in Figure 2) traffic load. The peaks of these curves identify points that are optimal for that individual application. However, in constructing network mechanisms we are typically more interested in achieving global or social optima, where the total performance – the performance of all applications – is optimized. There are many models of congested systems where the individually optimal point, often called the Nash equilibrium [7], is quite far from the socially optimal outcome [13]; the reasoning in [12] about the incentives provided by uniform dropping did not make the necessary distinction between individually and socially optimal outcomes. While uniform dropping provides performance-based incentives to constrain usage (and priority dropping did not), this does not at all imply that under uniform dropping the joint behavior resulting from applications seeking their individually optimal points will result in socially optimal, or close to socially optimal, outcomes. Despite this fact, our initial assumption about incentives was that uniform dropping's incentive properties were still far superior to those of priority dropping.

Thus, our initial assumptions were quite different from those embedded in the arguments against priority dropping advanced by McCanne *et al.* in [12]. To resolve these fundamentally conflicting intuitions and arguments, we set out to answer three basic questions about uniform and priority dropping:

1. Do uniform and priority dropping achieve the same optimal performance? To what extent does the answer depend on the burstiness of traffic, and is the difference ever significant?

2. Can RLM (as an example of a control mechanism used with uniform dropping) achieve the optimal performance under uniform dropping? Again, does the answer depend on the burstiness of traffic?

3. What incentives do these different dropping mechanisms present to individual applications? Do operating points where each user is individually optimal differ from the socially optimal operating point?

To address these questions, we use simulation and analysis of simple models of layered video applications transmitting over a network. The simulation models are, by necessity, crude approximations because we know little about the traffic and performance characteristics of future layered video sources, and even less about the likely nature of future background traffic. The analytical models suffer from these same drawbacks but, in addition, tractability requires even further simplification. Thus, we make no pretense that the models we used are *realistic* in any precise sense; they merely illustrate some of the basic issues involved, and highlight areas where future research is needed. These simulation and analytical models are presented in the following section.

## 3 Simulation and Analytical Models

In this section we describe our simulation and analytical models of applications and the network. For our purposes here, an application has two somewhat distinct aspects: its offered load (the nature of the packet stream the application transmits over the network) and its performance characteristics (how application performance depends on the network service those packets receive). The basic assumptions about the performance characteristics embedded in the analytical and simulation models are quite similar in spirit, and we describe them first. We then describe the network and traffic aspects of the simulation model, and then those of the analytical model.

### 3.1 Basic Assumptions about Performance of Layered Video

We represent the performance of an application by a *utility function*. This function maps the service received into some performance (or utility) level delivered to the end user. Below we describe a model for layered video, our canonical application, but it may well apply to other layered applications.

The canonical representation of layered video we use throughout this paper consists of some fixed number $L$ of layers. Each layer is characterized by a traffic stream and a potential value it provides to the application. One expects the *per bit* value to decrease with additional layers; the most value is derived from the information encoded in the base layer, and the relative value of bits in the enhancement layers decreases. For each layer $l$ let $a(l)$ denote the bandwidth and $f(l)$ denote the per-bit value. We assume that $f(l)$ is a non-increasing function.[7] If all packets are successfully received (we assume delay is not an issue, only dropping, so we ask only *if*, not *when*, packets are received), then the total utility is merely $\sum_{l=1}^{L} a(l)f(l)$. Define $F(l) = \sum_{k=1}^{l} f(k)a(k)$ to be the total utility of all layers up to $l$.

Our utility functions describe, in the presence of loss, how much value an application derives from the set of packets it actually receives. Since utility is a subjective measure that depends in large part on human

[7] As $l$ increases, the value of layer $l$ decreases. Hence, higher layers (in terms of their index) will have lower drop priority and lower layers will have higher drop priority.

perception (so it may vary from person to person) and on the characteristics of the coding algorithm used (so it may vary from implementation to implementation if different encoding schemes are used), the range of applicability of any particular utility function is limited. Rather than choose a single utility function and claim that it accurately represents the truth about application utility, we instead examine a family of extremely simple utility functions in an attempt to understand what impact different utility functions have on the results of our study.

We assume that the utility of each layer is independent of the other layers' utilities.[8] The utility of a given layer is a function of the loss experienced in that layer; we represent this by a non-decreasing function $g(z)$ with $g(0) = 0$ and $g(1) = 1$, where $z$ is the fraction of packets received in the layer. Thus, if $d(l)$ is the fraction of packets dropped in layer $l$ then the total utility is given by:

$$\sum_{l} a(l)f(l)g(1-d(l)) \tag{1}$$

To explore the impact of different utility functions, we use functions of the form $g(z) = z^m$, $m > 0$. The nonlinearities for $m \neq 1$ could be due to characteristics of the coding algorithm, human perceptual factors, or both.

### 3.2 Simulation Framework

We use discrete event simulation to study the performance of uniform and priority dropping. Our simulator, which used version 2 of the *ns* network simulator as a starting point and added new functionality as needed, incorporates the utility functions described above, as well as the relevant source models and network control mechanisms.[9] Below we describe the application source models, the router queueing and dropping algorithms, and the network topology.

**Source models**: We use an abstract layered source model that captures two essential characteristics of layered video traffic: (1) the instantaneous traffic in each layer varies over time, and (2) there is high correlation between the instantaneous traffic in each layer (one might expect factors such as motion or scene change that lead to changes in bit rate to have impact across layers). The layered source model is built out of individual layers. We first describe the traffic in the base layer ($l = 1$) and then describe the traffic generated by the higher layers.

We divide time into discrete intervals of length $\Delta_1$, and let $t$ be the index of the time intervals. Let $n_t$ be the number of packets sent in time interval $t$. All $n_t$ packets are sent back-to-back at a starting time chosen at random from a uniform distribution within the interval. In every time interval $t$, the rate $n_t$ is selected independently from the following random distribution: $n_t = 1$ with probability $\frac{P-1}{P}$, and $n_t = PA+1-P$ with probability $\frac{1}{P}$. This model produces hi-low sources that generate either $n_t = 1$ packet per interval or $n_t = PA + 1 - P$ packets per interval (and $A$ continues to describe the average number of packets per interval). Note that when $P = 1$, this model produces CBR-like traffic with $n_t = A$. Increasing $P$ yields increasingly bursty traffic. Throughout our simulations, we use $\Delta_1 = 1$ second, and $A = 4$ packets per interval. All packets have size $s = 1000$ bytes.

The higher layers are slight modifications of this model. We impose the requirement that $f(l)a(l) = f(1)a(1)$ so that all layers contribute equal value; for convenience, we assume $\frac{f(1)}{f(l)}$ is an integer. Then, for each time interval of the base layer (of length $\Delta_1$), we create $\frac{f(1)}{f(l)}$ subintervals of length $\Delta_l = \Delta_1 \frac{f(l)}{f(1)}$. As in the base layer, a certain number $n$ of packets are sent back-to-back in each of these subintervals, starting at some uniformly distributed starting time. Inter-layer correlations are captured by using the same value of $n_t$ in each subinterval $\Delta_l$ (for all layers $l$) of a given base interval $\Delta_1$; that is, a number $n_t$ is chosen for each time interval $t$ for the base layer, and this $n_t$ is used to govern the transmissions in each subinterval (for each higher layer) of the interval $t$. Thus, as $n_t$ varies randomly, all layers adjust their sending rates in concert.

For our simulations, we typically use $a(l) = 2^{l-1}a(1)$ and $f(l) = 2^{1-l}f(1)$, so each layer potentially contributes a unit of value.[10] With $\Delta_1 = 1$ second, $A = 4$ packets per interval, and $s = 1000$ bytes, we have average transmission rates per layer of 32kbps, 64kbps and 128kbps, etc.

**Dropping and Scheduling Algorithms**: Under uniform dropping, all packets have equal drop probability. We use tail drop in most of our experiments; a packet arriving at a full queue is dropped, otherwise it is queued for later transmission. We also use Random Early Detection [6] (RED) as our dropping algorithm in some experiments.

The priority dropping algorithm is slightly more complex and involves dropping on both input and on output. When a packet arrives and the queue is full, rather than dropping the arriving packet (as with drop tail) the arriving packet is queued and a packet (the latest in the queue) of the lowest priority among those packets in the queue is dropped. Dropping on output is also performed in order to prevent transmission of already queued low priority packets from causing later drops to higher priority packets. Dropping on output uses a threshold parameter $\zeta$, with $0 \leq \zeta \leq 1$. A packet in layer $l$ at the front of the queue is dropped if the total number of packets in the queue of higher priority than $l$ is greater than $\zeta$ times the total number of buffers; otherwise, the packet is transmitted. This heuristic allows low priority packets to be sent when the probability is small that they will cause subsequently arriving higher priority packets to arrive at

[8] In [1] we also consider utility models that capture dependency between layers, where the dropping rate in one layer may effect the utility of another layer. Such dependencies arise in many coding algorithms.

[9] The *ns* simulator is available at http://www-mash.cs.berkeley.edu/ns. Our extensions to the simulator, and the simulation scripts we ran to generate the results in this paper can be found at ftp://ftp.parc.xerox.com/pub/net-research/breslau/dropping.

[10] The decreasing per-bit value per layer is fundamental to our model. However, the grouping of bits into layers is arbitrary. We chose to hold the per-*layer* value constant, implying an increasing rate per layer. Our analytical model is not burdened by this somewhat arbitrary decision.

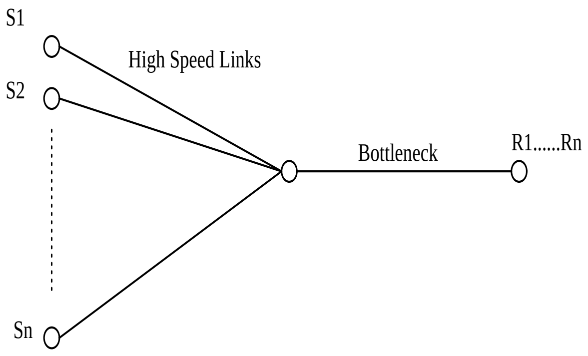


Figure 3: Simulation Topology

a full queue. We used the value $\zeta = 0.6$ in our simulations (both here, and in the subsequent places where parameter $\zeta$ is invoked).[11]

In the experiments described in this paper, we used a simple FIFO scheduling algorithm with the two dropping algorithms described above. We have also simulated uniform and priority dropping with Fair Queueing. Results of these experiments are reported in [1].

**Topology**: We use a very simple topology in our experiments (shown in Figure 3) to assess the impact of different dropping strategies across a single bottleneck link. There are $n$ sources and receivers. There is a single bottleneck link, and high speed links (10 Mbps) connect each source to the bottleneck. All dropping occurs at the bottleneck link. Note that we do not address the issue of multiple receivers per source since it is irrelevant to priority dropping, and RLM deals gracefully with large groups for uniform dropping. Thus, for clarity, we reduce the problem to its bare essentials by focusing on single receiver groups.

Inputs for each experiment include the number of sources, the traffic parameter $P$ used for each source, the bottleneck link bandwidth, and the number of buffers $b$ at the bottleneck link. Unless otherwise stated, our simulations use eight sources and a bottleneck bandwidth of 4 Mbps. We use $P = 1$ or $P = 5$ to produce smooth or bursty traffic, respectively, and $b = 60$ or $b = 20$ to test the effect of buffer size on our results.

### 3.3 Theoretical Framework

We augment our simulation study with theoretical analysis. We employ an extremely crude model that captures some of the essential features but leaves out many details. The model complements the simulations since the theoretical model can incorporate a wider variety of traffic models (in terms of their drop rates as a function of load) and the layers are infinitesimally small (avoiding effects due to the large size of the lower priority layers). Below we describe the model; the results of the analysis are presented in the following sections.

**Source Models**: Rather than having a finite number of discrete layers (labeled by $l$), we model each flow as having a continuum of layers (labeled by $x$), each with an infinitesimal unit of bandwidth (*i.e.*, $a(x) = 1$). Let $r_i$ denote the highest layer being sent by flow $i$. Since each layer consumes a unit of bandwidth, $r_i$ is also the total bandwidth sent by the $i$'th flow. The assumption that different flows send the same amount of bandwidth in each layer is clearly limiting, but it greatly simplifies our analysis and does not directly undercut our central concern, that being the tradeoff between priority and uniform dropping. As before, $f(x)$ is the potential value for each layer (if all packets are received in that layer), and $F(x) = \int_0^x dy f(y)$ denotes the total potential value for all layers up to $x$.[12]

The performance of the network, from the perspective of the $i$'th flow, is characterized by $d_i(\vec{r}, x)$ which is the drop fraction of packets in flow $i$, layer $x$, and where $\vec{r}$ describes the transmission rates of all flows. Note that the quantity $d_i(\vec{r}, x)$ can be interpreted as the drop rate (packets dropped per unit time), or the drop fraction (fraction of packets dropped), given that each layer consumes a single unit of bandwidth. The continuum version of the utility function given by Equation 1 is:

$$u_i = \int_0^{r_i} dx f(x) g(1 - d_i(\vec{r}, x)) \qquad (2)$$

**Dropping Algorithms**: We consider two forms of dropping behavior: uniform and priority dropping. For uniform dropping the basic principle is that there are no distinctions between layers as far as dropping is concerned: $d_i(\vec{r}, x) = d_i(\vec{r}, y)$ for all $x, y$. For priority dropping the basic assumption we make is that the dropping rate for higher priority packets is completely unaffected by the presence of lower priority packets. In reality, priority dropping schemes will never achieve this perfection, but this assumption makes the model tractable. This means that $d_i(\vec{r}, x)$ is independent of all $r_j$ as long as $r_j \geq x$ (and this applies to $j = i$ as well) and so, in particular, $d_i(\vec{r}, x) = d_i(\vec{r} \wedge x, x)$ (using the notation that $a \wedge b = \min[a, b]$ and that $(\vec{a} \wedge b)_i = \min[a_i, b]$).

**Scheduling Algorithms**: The basic behavior of the queueing system is represented by an increasing and convex ($D'' \geq 0$) function $D : \Re_+ \mapsto \Re_+$. $D(T)$ is the drop rate (packets dropped per unit time) resulting from a total traffic load of $T$; the drop fraction is given by $\frac{D(T)}{T}$. We make the basic approximation that the total drop rate depends only on the total traffic load (and is not a function of the individual flow rates).[13] Thus, we must have:

$$\sum_i \int_0^{r_i} dx\ d_i(\vec{r}, x) = D(\sum_i r_i)$$

The canonical example we use for the function $D$ is that of an M/M/1/b queue with unit service rate

[11] Designing an ideal priority dropping mechanism was not a goal of this research. We fully expect that one could improve on the priority dropping algorithm outlined here. However, experimentation has shown that this algorithm does accomplish our main goal which is to achieve high throughput while protecting higher priority packets from loss resulting from the transmission of lower priority packets.

[12] Note that since $a(x) = 1$, $f(x)$ is both the per-bit value and the per-layer value and we can leave out the term $a(x)$ in the expression for $F(x)$.

[13] Note that since the throughput is bounded, we must have $\lim_{T\to\infty} D'(T) = 1$, and so $0 \leq D'(T) \leq 1$ for all $T$; this fact will be relevant later in the paper.

(modeling a bottleneck bandwidth of 1) and $b$ buffers, so $D(z) = z^{b+1}\frac{1-z}{1-z^{b+1}}$. In the limit of infinite $b$, this reduces to the perfect bottleneck model where $D(z) = (z-1)_+$ (where we use the notation $x_+ = \max[0, x]$). The buffer size parameter $b$ can also be seen as describing the smoothness of the traffic, with infinite $b$ describing infinitely smooth traffic.[14]

We now compute the functions $d_i(\vec{r}, x)$ for uniform and priority dropping assuming FIFO scheduling.[15] For uniform dropping, the loss rates are given by:

$$d_i(\vec{r}, x) = \frac{D(\sum_i r_i)}{\sum_i r_i}$$

For priority dropping the loss rates are:

$$d_i(\vec{r}, x) = D'(\sum_j r_j \wedge x)$$

The drop fraction of each layer (which is independent of the flow) is given by the incremental increase in the total drop rate when that layer is added on (ignoring all lower priority layers).

We now use these simulation and theoretical frameworks to address our three key questions.

## 4 Performance

In this section we address the two questions concerning performance: (1) Do uniform and priority dropping achieve the same optimal performance? and (2) Can RLM achieve the optimal performance level under uniform dropping? We begin with results from our simulation experiments.

### 4.1 Simulation Results

Most of our simulation studies of performance are presented in the following form. For a given choice of utility function and network scenario, we simulated the network with each source sending up to level $l$ for $l = 1, \ldots L$ under both priority and uniform dropping. In addition, we simulated all sources using RLM (with uniform drop). All simulations were run for 600 simulation seconds; data collected during an initial warmup period of 160 seconds was discarded. Per source utility was computed using as input the percentage of those packets sent during the 440 second period that were delivered to the receiver (in other words, we set $(1-d) = \frac{r}{s}$ where $s$ is the number of packets sent in the layer and $r$ is the number received). The data is presented on a single graph, with one curve describing uniform drop, one curve describing priority drop, and a horizontal line depicting the performance level achieved by RLM. The horizontal axis indicates the number of levels sent by each source (for the uniform and priority drop tests). Average utility per source is plotted on the vertical axis. Below we present the main results from our simulations experiments.

[14] Increasing the buffer size and decreasing the burstiness of traffic are roughly equivalent; they both decrease the dropping rate at a given level of throughput. In our theoretical model, we only vary the parameter $b$, but in our simulations we separately vary buffers and burstiness.

[15] See [1] for the analogous treatment of the Fair Queueing scheduling algorithm.

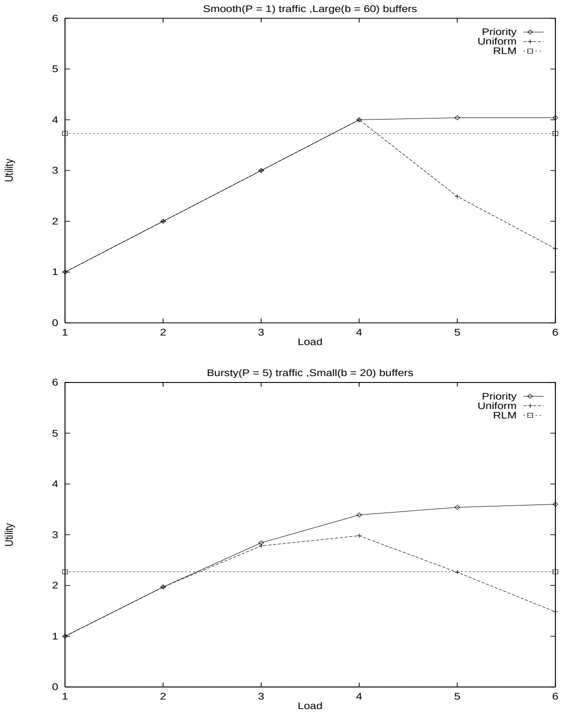


Figure 4: The top graph depicts the results with smooth ($P = 1$) traffic and large ($b = 60$) buffers. The peaks of the priority and uniform dropping curves are 4.04 and 4.00, respectively. RLM achieves a performance of 3.73. The bottom graph depicts the results with bursty ($P = 5$) traffic and small ($b = 20$) buffers. The peaks of the priority and uniform dropping curves are, respectively, 3.60 (priority) and 2.98 (uniform). RLM achieves a performance of 2.27.

#### 4.1.1 Burstiness and Buffer Size

We first explore the effect of burstiness and buffer size on performance, using a linear utility function ($g(z) = z$). We vary the parameter $P$ in our source model to control the burstiness of the traffic, and we vary the number of buffers, $b$, at the bottleneck link (4Mb) to control the ability of the switch to absorb bursts of packets. Figure 4 shows the results of two experiments, one where the number of packets sent per layer per interval is constant (*i.e.*, $P = 1$) and the buffers are large ($b = 60$, corresponding to 120 msec of buffering), and another where the traffic is burstier ($P = 5$) and the buffers smaller ($b = 20$).

For the case where the traffic is smooth and the buffers are large, the results are very much like those predicted in Figure 1. When offered load is less than the bottleneck link rate, uniform and priority dropping achieve the same utility. They both achieve the same maximum values, and then the performance of uniform dropping degrades as load increases, while the performance of priority dropping does not. RLM achieves nearly the same maximum utility.[16]

[16] The slight difference between RLM's utility and the maximum utility achieved by uniform and priority dropping is due to the absence of a mechanism to insure fairness in FIFO scheduling, and different flows can send at different rates (that is, the

Burstier data coupled with smaller buffers leads to markedly different results (as shown in the bottom graph in Figure 4). In this case, priority dropping achieves significantly higher average utility than uniform dropping (3.60 vs. 2.98).[17] The other striking difference between the smooth and bursty scenarios is the relative performance of RLM. Its utility with bursty traffic and smaller buffers is 2.27, around two-thirds of priority dropping and about three quarters of uniform dropping. With bursty traffic, there are enough losses, even at relatively lower levels of utilization to prevent RLM join experiments from succeeding. Thus, as we suspected, the results with bursty traffic look more like Figure 2 than like Figure 1.

One can relate these performance numbers to the equivalent amount of throughput being wasted. That is, with the exponential bandwidth per layer function $a(l) = 2^{l-1}a(1)$ we use in these simulations, the performance differential of between priority (3.60), uniform (2.98) and RLM (2.27) can be viewed as achieving useful throughputs of, respectively, 11.8, 6.92, and 4.08 (in units of the throughput of the base layer). The performance increase of 58% between RLM and priority dropping reflects roughly 1.3 additional layers (from $3.60 - 2.27 = 1.33$), which is equivalent to almost tripling the effective throughput. That is, while the priority dropping has a much higher effective throughput, the worth of these additional bits falls off so rapidly that the increase in utility is only 58%. If we use a linearly increasing bandwidth per layer $a(l) = l$ (see Figure 5) the performance numbers are 4.5 (priority), 3.71 (uniform), and 2.5 (RLM), corresponding to effective throughputs of 12.5, 8.84, and 4.50, respectively, again reflecting nearly a tripling of the effective throughput (and yielding an 80% increase in utility) when comparing priority to RLM.

The impact of burstiness on RLM's ability to adapt can be further demonstrated with the following contrived experiment where a single receiver performing RLM is adapting to available bandwidth in the face of bursty cross-traffic generated by a single on-off source with a sending rate of 6 Mbps, and exponentially distributed on and off times with average 100 msec. In this case, even though the utilization is only 65%, RLM is unable to utilize the available bandwidth and achieves a performance figure of only 1.19.

For completeness, we ran simulation of the two remaining combinations of buffer size and burstiness ($P = 1$ and $b = 20$; $P = 5$ and $b = 60$). The results (not shown here for lack of space) show that it is indeed *both* the burstiness of traffic and the size of buffers that account for the results described above. Bursty traffic presents opportunities for priority dropping to perform better than uniform dropping and also makes it more

receivers are not all subscribed to the same levels). This unequal allocation of resources leads to a slight decrease in total utility, as bits sent in lower priority layers by one source could be replaced by more valuable bits of another source. This performance degradation suffered by RLM disappears when a scheduling mechanism that allocates equal shares per flow, such as Fair Queueing is used.

[17] Note that this difference may be somewhat overstated in the sense that we only performed the uniform and priority tests with all senders sending the same number of layers. The best possible performance of uniform dropping may in fact occur when some flows are sending 3 layers and some are sending 4.

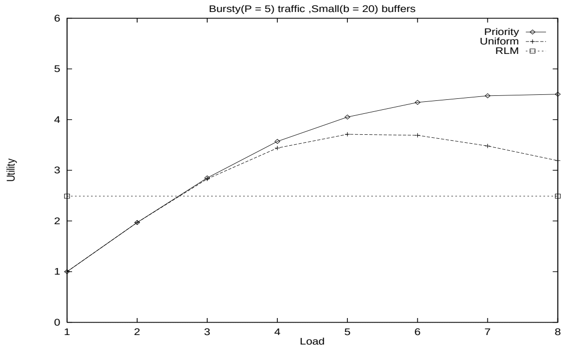


Figure 5: This graph depicts the results with bursty ($P = 5$) traffic and small ($b = 20$) buffers with $a(l) = l$. The peaks of the priority and uniform dropping curves are, respectively, 4.50 (priority) and 3.71 (uniform). RLM achieves a performance of 2.50.

difficult for the RLM adaptation algorithm to find the optimal operating level. As expected, large buffers can absorb burstiness and mitigate its ill-effects on the performance of uniform dropping and RLM.

### 4.1.2 Other Experiments

We conducted many additional simulation experiments to explore the impact of various parameters on our results. In particular we ran simulations with non-linear utility functions, different scheduling and dropping algorithms, varying bandwidth on the bottleneck link, and varying degrees of multiplexing. We briefly summarize those results here. A more detailed description is presented in [1].

To briefly summarize, in all of our experiments, priority dropping performed better than uniform dropping. As expected, this performance advantage increased with increased burstiness, decreased buffer sizes, decreased multiplexing, and decreased bandwidth per flow; the effect of nonlinear utility functions was ambiguous, sometimes increasing and sometimes decreasing the ratio. Using Fair Queueing instead of FIFO scheduling had little effect on results, since our simulations had fairly homogeneous flows; this is not a scenario in which fairness mechanisms can be expected to have an impact. In addition, we tested the RLM mechanism with RED [6], with little impact on results.

While we anticipated the direction of these effects of various parameters on the results, we thought their magnitudes would be much bigger. The performance advantage of priority dropping over the optimal uniform performance was quite modest; the biggest percentile advantage was roughly 27% (achieved in the case of reduced multiplexing). Results in the next section shed further light on the limits of the difference.

The performance advantage of priority dropping over RLM was somewhat more significant. For instance, in the case where the bottleneck capacity was reduced, priority dropping offered almost twice the performance of RLM. The performance advantage of priority over RLM had the same dependence on parameters as did the priority versus optimal uniform dropping, except that the effect of nonlinearities was no longer ambiguous. When the nonlinearity was quite extreme ($m = 100$) the performance advantage was essentially infinite (2.37 versus 0.02), but under more moderate nonlinearities ($m = 4$) the performance advantage was only 24%. While RLM did suffer under extremely bursty conditions, it was far more resilient than we had expected. This is partially due to the very rapid decrease in $f(l)$; for instance, if the per-flow bandwidth allowed roughly four levels per flow, then if RLM could utilize only 50% of the available bandwidth its performance disadvantage was a mere 25% (because it would get three out of the available four layers). When, as in Figure 5, the function $f(l)$ decreased less rapidly, the relative performance of RLM was significantly poorer.

### 4.2 Theory

We now turn to our theoretical model to provide additional insight about the performance results obtained through simulation. In particular, we derive results that are not bound to the particular layered source model used in simulations. To address the relative performance of uniform versus priority dropping, we first consider a single flow. The utility for uniform dropping is given by:

$$u(r) = g(1 - \frac{D(r)}{r})F(r) \qquad (3)$$

The utility for priority dropping is given by:

$$u(r) = \int_0^r dx f(x) g(1 - D'(x)) \qquad (4)$$

We start with a case that we can solve exactly: an M/M/1/b queue with $b = 1$, $g(z) = z^m$ for $m > 0$, and $f(x) = \frac{p}{(1+x)^{p+1}}$ for $p > 0$ (so $F(x) = 1 - \frac{1}{(1+x)^p}$). Here, $1 - d(r, x) = \frac{1}{1+r}$ in the uniform case, and $1 - d(r, x) = \frac{1}{(1+x)^2}$ in the priority case. Letting $u(r)$ denote the utility at a given sending rate $r$, in the uniform case we have $u(r) = (1+r)^{-m}(1 - \frac{1}{(1+r)^p})$ and so the maximizing $r$ value is given by $1 + r = (1 + \frac{p}{m})^{\frac{1}{p}}$ and the maximal utility is $u = \frac{p}{m}(1+\frac{p}{m})^{-(\frac{m}{p}+1)}$. In the priority case, we have $u(r) = \int_0^r dx p(1+x)^{-(p+1)}(1+x)^{-2m} = \frac{p}{2m+p}(1 - (1+r)^{-(p+2m)})$, and so the maximal utility is just $u = \frac{p}{2m+p}$. Let $R$ denote the ratio of maximal priority utility $u^{pri}$ to maximal uniform utility $u^{uni}$. Setting $w = \frac{p}{m}$ yields:

$$R = \frac{u^{pri}}{u^{uni}} = \frac{(1+w)^{(1+w^{-1})}}{2+w}$$

This is a decreasing function in $w$ so $R$ attains its maximal value in the limit $\frac{p}{m} \to 0^+$, and there $R = \frac{e}{2} \approx 1.359\ldots$[18]

We can numerically compute the ratio $R$ for more general cases, varying $b$, $m$, and $p$. We also consider the function $f(x) = \beta e^{-\beta x}$ and do similar computations. The results are summarized below.

Typically $R$ decreases with $b$; the performance advantage of priority dropping decreases as the traffic gets smoother (or, equivalently, the buffering gets larger). When varying the rate of decrease in $f(x)$ we find that the ratio decreases with $p$ (when $f(x) = \frac{p}{(1+x)^{p+1}}$) as long as $p \geq 0$ but has a peak at intermediate values of $\beta$ (when $f(x) = \beta e^{-\beta x}$). This is consistent with the intuition that priority dropping is of no use when the values $f(x)$ drop off too fast (so only the base layer is of significant value) or too slowly (so discriminating between layers is of little use). The dependence on $m$ is somewhat more subtle; for $b = 1$ the maximal ratio occurs for large $m$, but for greater values of $b$ the maximal ratio occurs for small $m$; we do not have an explanation for this behavior.

Note that in no case do we attain a higher value for $R$ than $\frac{e}{2}$. While the roughly 36% increase in performance achieved by priority dropping is certainly significant, we must admit that we had expected priority dropping to achieve higher levels of improvement under at least some conditions. We now conjecture that as long as $F$ and $D$ are smooth around the origin, this ratio is the highest possible. Our reasoning is as follows. We assume that for a given $g$ and $F$, the ratio $R$ is highest when $D(z) \approx z^2$ for small $z$. This is reasonable since higher powers (*e.g.*, $D(z) \approx z^3$) give lower ratios (as shown by our data for higher values of $b$), fractional powers (*e.g.*, $D(z) \approx z^{1.5}$) mean $D$ is not smooth near the origin contrary to our assumption, and $D(z) \approx z$ produces a smaller ratio (since $D'(z) \approx \frac{D(z)}{z}$ is roughly constant for small $z$ if the leading term in $D(z)$ is linear, and so the ratio is increased if we remove the linear term from $D$). Moreover, we assume that for such initially quadratic $D(z)$, the ratio $R$ increases when we substitute $(g(z))^2$ for $g(z)$, as suggested by our data for increasing $m$. Lastly, for $g(z) = z^m$ the performance for large $m$ does not depend on $F$ (as long as it is smooth near the origin, so $\frac{F(r)}{r}$ has a finite nonzero limit as $r \to 0^+$). Thus, it appears that the result that $R_{max} = \frac{e}{2}$ does not depend on the details of $D$ (aside from the leading term around the origin) or $F$, and is reached in the limit of $g(z) = z^m$ for diverging $m$. We are not able to prove this conjecture; whether or not $R_{max} = \frac{e}{2}$ remains the most interesting open theory question arising from our study.[19]

Thus, one of our preconceived notions, that of priority being able to achieve extremely large improvements over uniform dropping, is likely wrong. Further, using the same model, we find that priority dropping does not outperform uniform dropping in call cases. It is straightforward to show that if $g$ is a step function ($g(z) = 0$ if $x < \gamma$ and $g(z) = 1$ if $x \geq \gamma$ for some threshold $0 < \gamma < 1$) then uniform dropping outperforms priority dropping. However, one can show that

[18] Note that the case of $p = 0$ is ambiguous, so we only consider the limit. Setting $p = 0$ in the formula for $F$ yields a different answer than setting $p = 0$ in the formula for $f$ and then integrating; the limit is consistent with the formula derived from $f$, not $F$, in the case of $p = 0$.

[19] This bound is violated if we allow $D(z)$ to have a singularity at the origin, such as $D(z) \approx z^{1.5}$. In particular, when we look at the M/M/1/b formulae for $0 < b < 1$, where $D(z) \approx z^{1+b}$, we find that $R$ diverges as $m$ becomes infinite and $b$ vanishes. We don't know what such singular $D(z)$ functions would signify (perhaps extreme burstiness), or if they are accurate representations of reality, but we do not address such singular cases in this paper.

if $g$ is concave then priority dropping outperforms uniform dropping.

So far our theory has compared uniform and priority drop for only a single flow. We can extend our analysis to the multiple flow case without much additional complication. Consider the case where there are $n$ flows; when $\vec{r}$ denotes the transmission rates then the utilities are given by equations 3 and 4. It is straightforward to show that, in the uniform case, the total utility $\sum_{i=1}^{n} u_i$ is maximized when $r_i = r_j$ for all $i, j$. Similarly, in the priority dropping case the total utility is maximized when all $r_i$ are infinite. Consequently, the $n$-flow maximization problem with dropping function $D(z)$ yields the same optimality results, for both the uniform and priority dropping cases, as the 1-flow maximization problem with dropping function $\tilde{D}(z) = \frac{D(nz)}{n}$. Thus, the generalized form of our conjecture is that with an arbitrary number of flows (with the same conditions of smoothness on $D$ and $F$), the maximal ratio of the total utilities of priority dropping to uniform dropping is bounded above by $\frac{e}{2}$.

## 5 Incentives

The previous results compared the performance of priority dropping to the optimal uniform dropping performance, and to that achieved by RLM. However, in this discussion we assumed that system-wide optimality was the only goal. We now remove that assumption and address our third question: what incentives do different dropping mechanisms present to applications? This issue has two aspects: (1) the properties of Nash equilibria in the presence of performance-based incentives, and (2) the effect of nonperformance incentives on these Nash equilibria. We now discuss these two aspects in turn.

### 5.1 Nash Equilibria

We first assume that there are no usage incentives other than performance. When using priority drop users have no performance-based incentive to constrain usage, whereas they do when using uniform drop. In this section we demonstrate that this does not necessarily imply that uniform drop naturally leads to socially optimal operating points. We return to our theoretical model, initially considering FIFO scheduling, and investigate what happens when there are $n$ individually optimizing users; that is, we assume that users adjust their transmission level so as to maximize their own utility.

Consider the case of perfectly smooth traffic (modeled by setting $b = \infty$ in the M/M/1/b model), so $D(z) = (z-1)_+$. Then the maximal utility for a given user with priority dropping is at least $F(\frac{1}{n})$, with equality holding if all other users have $r_j \geq \frac{1}{n}$. Thus, the socially optimal point, and the Nash equilibrium point, is any vector $\vec{r}$ such that $r_j \geq \frac{1}{n}$ for all $j$.

For uniform dropping, the utility for a given user sending at rate $x$, with $r$ sent by everyone else, is $u(x) = g(\min[1, \frac{1}{x+(n-1)r}])F(x)$. First we compute the socially optimal outcome (in which we know $r_i = r_j$ as we argued earlier), so we maximize the function $u(r) = g(\min[1, \frac{1}{nr}])F(r)$. Note that for $nr < 1$, $u(r) = F(r)$ and $F(r)$ is nondecreasing, so a maximal point must exist with $nr \geq 1$. For $nr > 1$, assuming all first derivatives exist, we can calculate $u'(r) = g(\frac{1}{nr})F'(r) - g'(\frac{1}{nr})F(r)\frac{1}{nr^2}$. Note that $rF'(r) \leq F(r)$. If $g$ is concave, then we also have $g'(\frac{1}{nr}) \leq nrg(\frac{1}{nr})$ and thus $u'(r) \leq 0$ for $nr > 1$ and so at least one socially optimal operating point has $nr = 1$. However, there are cases with nonconcave $g$ where the socially optimal operating point has $nr > 1$. Consider, for instance, the case where $g(z)$ is a step function (with the step at $z = \gamma < 1$). Then, the socially optimal value is given by $nr = \frac{1}{\gamma} > 1$ with per-flow utility $F(\frac{1}{n\gamma})$.

Next, we compute the Nash equilibrium. The equilibrium occurs when $\frac{du}{dx} = 0$ when $x = r$. Thus, at the equilibrium value $r$ we must have $g(\frac{1}{nr})F'(r) = g'(\frac{1}{nr})F(r)\frac{1}{(nr)^2}$. Consider the case where $g(z) = z^m$. Then, the socially optimal value is $nr = 1$ for all $m \geq 1$ and the Nash equilibrium is reached when $\frac{rf(r)}{F(r)} = \frac{m}{n}$. For the case where $f(x) = (1-x)_+$, the Nash equilibrium is $r = \frac{1-\frac{m}{n}}{1-\frac{m}{2n}}$. The per-flow utility at this Nash equilibrium is $n^{-m}\frac{\frac{m}{n}}{2(1-\frac{m}{2n})}(\frac{1-\frac{m}{2n}}{1-\frac{m}{n}})^m$. This per-flow utility, and the total utility, vanish in the limit of large $n$ or large $m$. The socially optimal per-flow utility is $\frac{1}{n}(1-\frac{1}{2n})$ for $m \geq 1$, and so the total utility $(1-\frac{1}{2n})$ approaches unity in the large $n$ limit.

These results show that even when traffic is completely smooth (*i.e.*, in the infinite $b$ limit), the Nash equilibria and the socially optimal operating points can be quite different; in particular, the Nash equilibrium can asymptotically (in the limit of large $n$) have zero total utility whereas the socially optimal point has full unit total utility. Thus, the performance-based incentives provided by uniform dropping do not lead to socially optimal, or even adequate, outcomes with FIFO scheduling. While we cannot solve these models exactly for burstier traffic, numerical computations for various choices of $b$ and $m$ for $f(x) = (1-x)_+$, $f(x) = \frac{p}{(1+x)^{p+1}}$ and $f(x) = \beta e^{-\beta x}$ show that these results continue to hold.

The intuition behind these results is that when a single user increases her usage, the penalty of that increased usage (in terms of an increased drop rate) is spread among all users, but the benefit (in terms of increased throughput) goes exclusively to the increasing user. Thus, the equilibrium occurs not when the benefit equals the penalty (which is the socially optimal point), but when $\frac{1}{n}$'th of the penalty equals the benefit, and this occurs only for much larger load levels (and this effect is magnified for larger $n$).

The above results show that the distinction between Nash and socially optimal operating points is significant when FIFO scheduling is used with uniform dropping. However, when uniform dropping is used with Fair Queueing, the first-order conditions for Nash equilibrium become identical to those for the socially optimal outcome. This is because the penalty for increased usage is born exclusively by the user with the highest sending rate, and so this user makes the socially optimal penalty/benefit tradeoff. Thus, with Fair Queueing, uniform dropping does indeed provide the proper

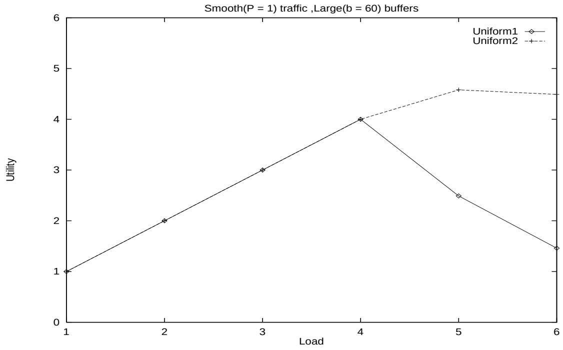


Figure 6: This graph compares the individually optimal and socially optimal operating points with $P = 5$ and $b = 60$. The curve 'Uniform1' describes the case when all 8 flows are sending the same number of levels (determined by the reading on the horizontal axis). Clearly the socially optimal operating point is when all flows are sending the first four levels. The curve "Uniform2' depicts the results when the 7 background flows are sending the first four levels, but the single test flow is sending up to the level determined by the reading on the horizontal axis. The utility of this flow is maximized when sending the first five levels. Thus, the socially optimal level is not a Nash equilibrium.

incentives.[20]

The simulations do not have exactly analogous behavior because the large granularity of the flow layers; each level consumes a significant amount of bandwidth, particularly the higher levels, and so typically one finds multiple equilibria rather than the single equilibrium points found in the simple theory models. Nonetheless, we can observe, as shown in Figure 6, that the socially optimal operating points are not Nash equilibria. The socially optimal result is to have each flow sending the first four levels, but when the background flows are held fixed at that level, a single test flow maximizes its utility by sending the first five levels.

One response to these dire results about Nash equilibria is that users won't be selfish and will just use RLM, which seems to achieve fairly reasonable results even for large $n$. This may very well be true. If such compliance is to be expected, then one can also expect users to use RLM with priority dropping to restrain usage, thereby removing the one problem with priority dropping. For instance, in the scenario with $P = 5$ and $b = 60$, RLM with priority dropping achieves a performance of 3.19 (versus 2.96 with uniform dropping) and constrains usage to only sending levels 1 through 3, plus occasionally level 4. The performance of RLM with priority dropping can be further improved by adjusting the dropping level at which RLM leaves a level (or considers a join experiment to have failed), thereby tuning the tradeoff between performance and bandwidth usage.

The behavior of RLM in the context of priority dropping is relatively insensitive to the burstiness of the traffic because the operating point need not be precisely at the bottleneck bandwidth. Thus, priority dropping is only a problem if we are worried about incentives, and in that case both priority and uniform dropping have problems (with FIFO scheduling).

Another response to the results about Nash equilibria is that there may be other sources of incentives that might make the Nash equilibria more palatable. These other incentives also presumably apply to priority dropping, and so usage would be constrained in both cases. We now address these other incentives.

### 5.2 Nonperformance Incentives

Above we assumed that the only incentives were related to the performance of the application itself. However, users clearly face other forms of incentives. The total bandwidth available to a user is limited to the speed of their access line, and for many users this access line is fairly slow (either a modem, or a shared T1); in addition, some hosts are quite limited in their network I/O. For any of these reasons, traffic from one application can have a performance impact on the user's other applications; *e.g.*, one's video traffic might congest one's own web traffic. Moreover, in some cases the Internet access charges are based, at least in part, on usage. Thus, we assume that there is some slight disincentive to send traffic. This seems compatible with reality given that most users do not usually send or receive traffic that they know is worthless. It is important to note that these incentives can either be applied to the sender or the receiver of the traffic, depending on the application (who is deriving value from the application) and the context (whose access line is congested).

We model this usage-constraining incentive by applying a cost $c$ for each unit of bandwidth. While this conjures up the idea of a per-bit price, that is not what is implied; in *reality* this usage-constraining incentive can come from any one of a number of nonmonetary sources, but it is most easily *modeled* by assigning a cost for transmission. If the incentive is placed on the sender, the cost is applied for each unit of bandwidth *sent*, and if the incentive is placed on the receiver, the cost is applied for each unit of bandwidth *received* (*i.e.*, we assume that there is no receiver disincentive for packets that never arrive). A user maximizes utility minus cost. In the case of perfectly smooth traffic, an infinitesimally small $c$ applied to the sender will provide incentive to operate at the bottleneck throughput; the excess traffic is completely worthless, and any nonzero disincentive is enough to throttle throughput back to the bottleneck capacity. However, if the incentive is present at the receiver, there is no constraint on usage, since no additional traffic arrives. Thus, applying incentives at the receiver involves somewhat of a paradox; we wish to prevent receivers from joining layers that will be mostly dropped, but need to do so by applying incentives only for those packets that are *not* dropped.

We can extend our theory model to look at this question for more general traffic loads, still with a single flow. We first consider only priority dropping, and assume that $c$ is the cost of the incremental layer $r$ (it

[20] While typically Fair Queueing does provide better incentives than FIFO, in general when users are heterogeneous the Nash equilibria under Fair Queueing are not socially optimal. This result holds in our example because users have the same utilities and flow structure. See [13] for a fuller discussion.

doesn't matter what the "charge" is for other layers, that will just be a fixed cost in these equations, ignoring income effects[21]). If the incentives are applied at the sender, the maximality condition is $u'(r) = c$ but if incentives are applied at the receiver, the maximality condition is $u'(r) = c(1 - D'(r))$. From the expression in equation 4, we find that the resulting equation is:

**Sender Incentives** $f(r)g(1 - D'(r)) = c$

**Receiver Incentives** $f(r)g(1-D'(r)) = c(1-D'(r))$

The relevant question is whether or not usage is stabilized for any nonzero $c$; by stabilized we mean that all solutions for $r$ are finite for any nonzero $c$. When the sender is "charged" usage is stabilized if either $\lim_{r\to\infty} f(r) = 0$ or $\lim_{z\to 0^+} g(z) = 0$. When the receiver is "charged" usage is stabilized if $\lim_{z\to 0^+} \frac{g(z)}{z} = 0$ and $D'(z) < 1$ for all finite $z$. Note that these are sufficient but not necessary conditions, but we expect them (or other sufficient conditions) to hold quite widely.

For the example treated in Section 4 with $b = 1$ and $f(x) = p(1+r)^{-(p+1)}$ and $g(z) = z^m$, we find:

**Sender Incentives** $1 + r = c^{\frac{-1}{p+2m+1}}$

**Receiver Incentives** $1 + r = c^{\frac{-1}{p+2m-1}}$

If we have $p \geq 1$ and $m > 0$ then usage is always stable in both the sender and receiver incentive cases; the usage levels decrease with increasing $p$ and $m$.

Thus, for a wide range of conditions, usage stabilizes even if incentives apply only for received packets (although at higher levels of usage than if incentives apply for sent packets). While there is clearly no stabilization for $b = \infty$ when receivers are "charged" (this case violates the $D'(z) < 1$ assumption), numerical computations show that stabilization occurs for all finite $b$, and for other choices for $f$ and $g$. The point is that with even slightly bursty traffic, some fraction of packets get through and so joining a layer that is almost completely useless (because of the high drop rates) is discouraged because the user is "charged" for the packets that do get through.

These nonperformance incentives also improve the nature of the Nash equilibrium under uniform dropping. We can return to our simple model and compute the Nash equilibrium after adding a small "charge" $c$ for usage. One way to compare the relative effectiveness of these nonperformance incentives in the priority and uniform dropping cases is to look at the values of $c$ for which they achieve $r$ values that are comparable to the socially optimal $r$ value for $c = 0$.[22] In the cases we computed, for sender "charging", priority dropping required lower levels of nonperformance incentives to restrain usage to these levels. Thus, these incidental sender incentives can more easily restrain usage in priority dropping than prevent the poor Nash equilibria under FIFO service with uniform dropping. When receivers incur these nonperformance incentives, the usage levels of uniform and priority dropping are roughly comparable (with the $c$ values needed to restrain usage for priority dropping being slightly higher than those needed for uniform dropping). Note that, as expected, both uniform and priority dropping required significantly higher levels of these incentives when the receiver incurred them.

In light of these results, at the very least both uniform and priority dropping have problems with incentives, and one might make a reasonable case that priority dropping, because it is more easily restrained by sender-incurred nonperformance incentives, actually has better incentive properties than uniform dropping.

## 6 Discussion

This paper is devoted to a comparison of uniform and priority dropping in the fairly narrow context of layered video applications. Our results are both humbling and ambiguous. "Humbling" because some of our preconceived notions were wrong, and "ambiguous" because the results do not provide a clear answer to whether adopting priority dropping would provide significant benefit to layered digital video applications. Priority dropping certainly does result in higher performance than uniform dropping when $g$, the utility function, is concave. However, contrary to our expectations, the performance advantage of priority dropping over the optimal uniform dropping performance is quite modest. Moreover, we conjecture that there is a universal upper bound of roughly 36% on this performance gap.

However, the real point of comparison for uniform dropping is not the optimal point on the uniform dropping curve, which is an ideal point that we cannot reliably achieve in practice, but instead is the performance level achieved by an actual endpoint adaptation algorithm. For the comparisons in this paper, we used the RLM algorithm as an example of such an algorithm since we are aware of no algorithm that performs better. Here too we were surprised; the RLM adaptation algorithm is far more resilient than we expected. Under some rather extreme conditions – very bursty background traffic or high degrees of nonlinearity – RLM performed quite poorly; however, contrary to our initial expectations, RLM managed to achieve fairly adequate performance in a very broad range of less extreme conditions. Our current explanation for this is that when $f(l)$ decreases rapidly (as was the case in our simulations), one can use only a small fraction of the available bandwidth and still attain reasonable performance, since most of the value lies in the first few layers. While our expectations about the performance differences were irrationally exuberant, we do not want to minimize the fact that priority dropping achieved performance improvements of 50% to 100% over RLM in many settings. Thus, the performance improvements offered by priority dropping are indeed significant, and the conjectured bound of roughly 36% is not an indication of the relative performance of priority dropping to RLM, or to any other endpoint adaptation algorithm.

[21] Income effects are where the marginal utility of money depends on the total amount.

[22] This is a somewhat arbitrary comparison, but it is asking how large do these nonperformance incentives have to be to restrain usage to a given level, and choosing the socially optimal level for $c = 0$ as that level seems like a reasonable choice.

While the performance properties of uniform drop were unexpectedly good, the incentive properties of uniform drop with FIFO service were, at least in theory, surprisingly poor. Moreover, these incentive properties, or more particularly the performance at the Nash equilibrium, were especially bad with a large population of flows. Using Fair Queueing instead of FIFO largely alleviates these incentive problems for uniform dropping.

In contrast, the incentive aspects of priority dropping may not be as bad as advertised. While priority dropping does provide poor performance-based incentives, even minimal amounts of nonperformance incentives will rectify the situation. In fact, when these usage-constraining incentives are incurred at the sender and when FIFO service is used, usage is more easily constrained in the priority dropping case than in the uniform dropping case.

Thus, we end up with somewhat of a paradox. If one takes the incentive issues seriously, then priority dropping may be better than uniform dropping, at least with FIFO service. But if one conjectures that instead users will be well-behaved and use RLM with uniform dropping in spite of their own personal incentives, then one could just as easily conjecture that users would use RLM (or some similar protocol) with priority dropping, which would yield better overall performance at the same level of bandwidth consumption. Thus, the argument that incentives are the reason to prefer uniform dropping to priority dropping is only valid if one believes that Fair Queueing (or some other protocol that enforces fairness) is used, or that the sender-based nonperformance usage constraining incentives, whatever their origin, are vanishingly small.

We hasten to note that this is only an initial study of this rather fundamental question. There are many unresolved questions about the performance and incentive properties of uniform and priority dropping. However, we think the most pressing open questions left by our study are those concerning the nature of application utility functions. We do not, nor does the research community we suspect, have a sense of whether our utility models capture the essential aspects of reality. Knowing how to best model application utility would provide much-needed guidance to network designers in their analysis of network performance, and may lead to more definitive answers to the questions posed here.

## References


[1] Sandeep Bajaj, Lee Breslau, and Scott Shenker. Uniform versus priority dropping for layered video – extended version. Preprint, June 1998.

[2] Jean-Chrysostome Bolot and Thierry Turletti. A rate control mechanism for packet video in the internet. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Toronto, Canada, June 1994.

[3] Jean-Chrysostome Bolot, Thierry Turletti, and Ian Wakeman. Scalable feedback control for multicast video distribution in the internet. In *Proceedings of ACM Sigcomm*, pages 58–67, London, England, August 1994. ACM.

[4] Stephen Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, 1991.

[5] Steve Deering. Internet multicast routing: State of the art and open research issues. *Multimedia Integrated Conferencing for Europe (MICE) Seminar at the Swedish Institute of Computer Science, Stockholm*, October 1993.

[6] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[7] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, Cambridge, Massacachusetts, 1991.

[8] M. W. Garrett and W. Willinger. Analysis, modeling and generation of self-similar VBR video traffic. *Computer Communications Review*, 24(4), October 1994. SIGCOMM '94 Symposium.

[9] D. Hoffman and M. Speer. Hierarchical video distribution over internet style networks. *Proceedings of IEEE International Conference on Image Processing, Lausanne, Switzerland*, pages 5–8, September 1996.

[10] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic. In Deepinder P. Sidhu, editor, *Proceedings of ACM Sigcomm*, pages 183–193, San Francisco, California, September 1993. ACM. also in *Computer Communication Review* 23 (4), Oct. 1992.

[11] S. McCanne and M. Vetterli. Joint source/channel coding for multicast packet video. *Proceedings of IEEE International Conference on Image Processing, Washington, DC*, pages 25–28, October 1995.

[12] Steven McCanne, Van Jacobson, and Martin Vetterli. Receiver-driven layered multicast. In *Proceedings of ACM Sigcomm*, pages 117–130, Palo Alto, California, August 1996.

[13] Scott Shenker. A game theoretic analysis of switch service disciplines. *Transactions on Networks*, 3(6):819–831, 1995.

[14] Thierry Turletti. The INRIA videoconferencing system IVS. *Connexions*, 8(10):20–24, October 1994.

[15] Thierry Turletti and Jean-Chrysostome Bolot. Issues with multicast video distribution in heterogeneous packet networks. *Proceedings of Sixth International Workshop on Packet Video Portland,OR*, September 1994.

[16] M. Vishwanath and P. Chou. An efficient algorithm for hierarchical compression of video. *Proceedings of IEEE Interantional Conference on Image Processing, Austin,TX*, November 1994.