Proportional Differentiated Services, Part II: Loss Rate Differentiation and Packet Dropping

Constantinos Dovrolis Parameswaran Ramanathann
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, WI 53706-1691
dovrolis@ece.wisc.edu parmesh@ece.wisc.edu

Abstract

The proportional differentiation model was proposed in [1] as a target for controllable and predictable relative differentiated services. [1] considered only the delay differentiation aspect of the model, and focused on packet scheduling mechanisms. In this paper, we extend the proportional differentiation model in the direction of loss rate differentiation. Several previous mechanisms for buffer management and packet dropping, such as complete buffer partitioning, partial buffer sharing, or multi-class RED, are not suitable for relative differentiated services. We propose and evaluate two dropping mechanisms that closely approximate the proportional loss rate differentiation model. The two droppers, $PLR(\infty)$ and $PLR(M)$, differ in the time interval over which the loss rates are measured and proportionally adjusted. This difference results in several trade-offs, in terms of implementation complexity, accuracy, and ability to deal with nonstationary traffic loads. We also re-evaluate the delay differentiation that the Waiting Time Priorities (WTP) scheduler of [1] can achieve in the presence of finite buffers and packet losses; this study extends the results of [1] providing further insight on the behavior of WTP in heavy load conditions. Finally, we examine the coupled effect of delay and loss rate proportional differentiation on the throughput of bulk-transfer TCP connections.

1 Introduction

The IETF Differentiated Services (DiffServ) working group was formed in early 1998 with the goal to standardize a few per-hop types of prioritized packet forwarding, called Per-Hop Behaviors (PHBs). The original vision was that PHBs at the network core can be combined with sophisticated mechanisms at the network edges, such as admission control, per-flow classification, or per-flow marking, in order to construct absolute or relative end-to-end services [2]. Since the DiffServ PHBs operate on large traffic aggregates, instead of individual flows, this architecture is expected to be more scalable than the 'flow-centric' IntServ architecture. The early efforts in DiffServ focused on the Premium service [3], which promises an end-to-end service with very small queueing delays and practically no losses, and on the Assured service [4], which offers a bandwidth profile to the user, if the network is appropriately provisioned, through edge marking and priority dropping in the core. The Premium and Assured services have been analytically studied in [5] and [6], and a refinement of the Assured service has appeared in [7]. Several other approaches for scalable differentiated services have been also proposed. The SCORE architecture uses QoS-related information carried in the packet header (similar to the MPLS technology), in order to provide fair-queuing [8] or guaranteed delay [9] services; SCORE requires per-flow state only at ingress routers. With a similar architecture, [10] proposed a scheme in which resource management and admission control are performed only at egress routers, without any coordination between backbone routers or per-flow management. The Asymmetric Best Effort scheme [11] provides two service classes: one for delay-sensitive applications and another for throughput-sensitive applications.

In the last two years much have been learned about the nature of the 'Internet QoS' problem, and about the limitations of the proposed solutions. In the case of the Premium service, it has been shown that the use of aggregate scheduling can lead to quite large delay bounds, with significant buffer requirements [12]. In the case of the Assured service, [13] has shown that it is hard to achieve high service assurance, coarse spatial granularity, and high network utilization at the same time. The Assured service has also been studied in the context of TCP bulk transfers in [14] and [15]; the main result is that under certain conditions it is impossible to provide a certain bandwidth profile to a TCP connection. Additionally, in practice, both the Premium and Assured services may need some kind of route-pinning techniques [13]. However, even with these research advances, most of the network providers today are reluctant to deploy QoS mechanisms (IntServ or DiffServ). Although it is early to draw conclusions, our belief is that the major reason for this reluctance is the complexity that is associated with most of the QoS architectures. This complexity is not only in building routers for such services, but also in the required pricing infrastructure, the interdomain policies and their monitoring, and on even more practical concerns, such as the maintenance and debugging of QoS-capable networks. It is for this reason that we focus on simpler QoS architectures, which can be incrementally deployed, providing a benefit to the network provider even if they are only used on individual resource-constrained links.

Such a simple approach in the DiffServ framework is the *relative differentiated services* [16]. In this approach, the network traffic is grouped into N classes of service which are ordered, such that *Class i is better (or at least no worse) than Class $(i-1)$ for $1 < i \leq N$, in terms of local (per-hop) metrics for the queueing delays and packet losses.* The *Class Selector PHBs*, recently standardized by the IETF [17], follow this model of relative service differentiation. In this context, applications and users do not get an absolute service level assurance, such as an end-to-end delay bound or bandwidth, since there is no admission control and resource reservations. Instead, the network assures that higher classes will be relatively better than lower classes, and so it is up to the applications and users to select the class that best meets their requirements, cost, and policy constraints. In [1], we proposed the *proportional*

This work was supported by a research fund from Cisco Systems Inc.

differentiation model as a versatile way to provide relative differentiated services. The proportional differentiation model aims to provide the network operator with the *'tuning knobs'* for adjusting the quality spacing between classes, *independent of the class loads*; this cannot be achieved with other relative differentiation models, such as strict prioritization, capacity differentiation, or price differentiation. With the proportional differentiation model, the relative service differentiation is:

- *Controllable*, meaning that the network operators are able to adjust the quality spacing between classes based on their selected criteria, and
- *Predictable*, in the sense that the differentiation is consistent (i.e., higher classes are better, or at least no worse) independent of the variations of the class loads.

In this paper we extend the model of proportional differentiation to include loss rate differentiation. The likelihood of packet losses is a very important performance measure for most of the Internet traffic, and it is especially important for all TCP-based applications. Several previous buffer management and packet dropping mechanisms, such as complete buffer partitioning, partial buffer sharing, or multiclass RED have received extensive attention in the past. These schemes, however, are not appropriate in the context of relative differentiated services, because they do not dynamically adjust the buffer distribution between classes so that higher classes are better than lower classes. We propose and evaluate two packet dropping algorithms that decide which class to drop a packet from, based on the measured class loss rates. These droppers provide controllable and predictable relative differentiation, as they closely approximate the proportional loss rate differentiation model. The two droppers, $PLR(\infty)$ and $PLR(M)$, differ in the time interval over which the loss rates are proportionally adjusted. This difference leads to trade-offs in their implementation complexity, their accuracy in meeting the proportional loss rate constraints, and their ability to deal with nonstationary traffic loads. We also re-evaluate the delay differentiation that the Waiting Time Priorities (WTP) scheduler of [1] can achieve, in the presence of finite buffers and packet losses. This study extends the results of [1], providing further insight for the WTP behavior in heavy load conditions. Finally, we examine the coupled effect of delay and loss rate proportional differentiation on the throughput of bulk-transfer TCP connections.

The structure of the paper is as follows. Section 2 describes several previous buffer management and dropping mechanisms, explaining why they are not suitable for relative differentiated services. In Section 3, we present the proportional differentiation model in terms of both delay and loss rate differentiation. Section 4 presents the two PLR droppers, while Section 5 evaluates and compares them. Section 6 is an extension of [1], as it re-evaluates the WTP scheduler in the context of finite buffers and lossy conditions. Section 7 examines the differentiation that results in the throughput of bulk TCP connections, with per-hop proportional delay and loss rate differentiation. The paper concludes in Section 8.

2 Previous Work on Packet Droppers

Prioritized buffer management mechanisms have received significant research attention, especially in the ATM context; for a review of the early works in this field see [18, 19]. Several of those efforts assumed that the *pushout* technique, i.e., to drop a packet that is already buffered in the forwarding engine (as opposed to drop the packet that just arrived), is hard to implement. In fact, buffer management schemes such as Complete Buffer Partitioning (CBP) [18], Partial Buffer Sharing (PBS) [19], or Dynamic Thresholds [20], were proposed exactly because they avoid pushout. Recently, however, it has been shown in actual implementations of high-performance switches that pushout is not difficult to implement [21, 22]. In this paper, we assume that a pushout scheme can be implemented without significant complexity.

In the following, we review some major buffer management schemes, and examine them from the perspective of relative differentiated services.

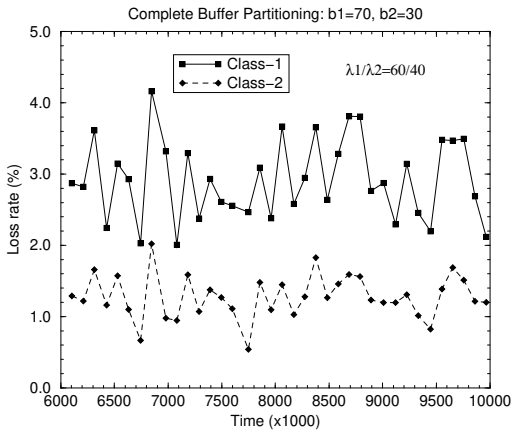
Pushout, or Lowest Priority First (LPF): The most straightforward pushout scheme is to remove a packet from the lowest backlogged class when a packet needs to be dropped [18, 19]. The corresponding scheme, in the delay differentiation context, is the Strict Priority scheduler, which serves the highest backlogged class first. Although such mechanisms provide consistent relative differentiation ('higher classes are always better'), they do not provide any means for adjusting the quality spacing between classes, and additionally, they can cause starvation in the lower classes.

Complete Buffer Partitioning (CBP): Another simple scheme is to statically partition the buffer space between classes (i.e., class i has b_i available buffers). From Little's law, the ratio of aggregate backlogs between two classes i and j is $\bar{q}_i/\bar{q}_j = \bar{d}_i/\bar{d}_j\ \lambda_i/\lambda_j$, where λ_i is the input (accepted) rate in class i and $\bar{d}_i$ is the average queueing delay of the accepted packets in class i. This relationship shows that the buffer partitions must be selected based on the load distribution between the classes, and the delay differentiation provided by the scheduler. CBP increases the overall loss rate because packets can be dropped even when there are available buffers. Another problem is that the selection of the buffer partitions in order to achieve a certain loss rate differentiation is quite sensitive to variations in the load distribution. This is shown with two sample paths in Figure 1. The buffer partitions in Figure 1-a were selected so that the loss rate of the lowest class is about 3%, while the loss rate of the highest class is about 1%, in the long run, when the load distribution is Class-1: 60%, Class-2: 40%. A slight change in the load distribution (Class-1: 55%, Class-2: 45%), however, makes the selected buffer partitions totally inappropriate, since they result in a larger loss rate for the higher class than for the lower class (Figure 1-b).

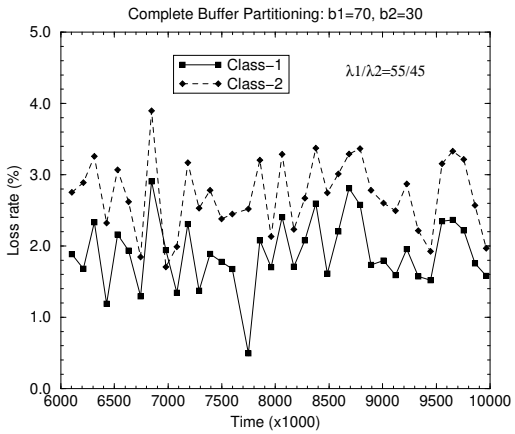
Partial Buffer Sharing (PBS): In the PBS scheme [19], a packet from class i is accepted in the buffer space only if the *aggregate backlog* is less than a threshold b_i, with $b_i < b_j$ for $i < j$. Consequently, higher classes are given more buffer space than lower classes. The PBS scheme does not require pushout, and offers the operator with a set of adjusting parameters. However, the required thresholds b_i are also sensitive to the load distribution, when attempting to adjust the class loss rates. Another problem that we observed experimenting with PBS, is that the buffer thresholds cannot 'fine tune' the loss rate differentiation between classes, in the sense that a slight change in the thresholds b_i can lead to a dramatic change in the obtained loss rates. More generally, the problem with both CBP and PBS is that they are *static resource provisioning mechanisms*, and so they do not dynamically readjust the partitioning of buffer resources among classes based on the actual class loads. A different approach to the buffer management problem, that is pursued in this paper, is to measure the loss rates at run-time, and to select the drop-target class so that the loss rate differentiation is explicitly controlled.

Multiclass RED and RIO: The multiclass RED [15] and RIO [4] mechanisms are priority-dropping extensions of the RED buffer management scheme [23]. Both mechanisms are essentially instances of the PBS scheme, since each class is given a buffer threshold b_i, and arriving packets from that class are dropped when the aggregate backlog is more than b_i [1]. Even though these mechanisms

[1] In RIO, an arriving IN packet is dropped when the backlog of IN packets only is



(a) Well provisioned buffer partitions



(b) Poorly provisioned buffer partitions

Figure 1: Sample paths of the loss rates obtained with CBP. A loss rate point is measured in every T=100,000 packet arrivals. A WTP scheduler [1] with $\delta_1/\delta_2 = 2$ provides the delay differentiation. The offered load is 95%. More simulation details are given in §5.

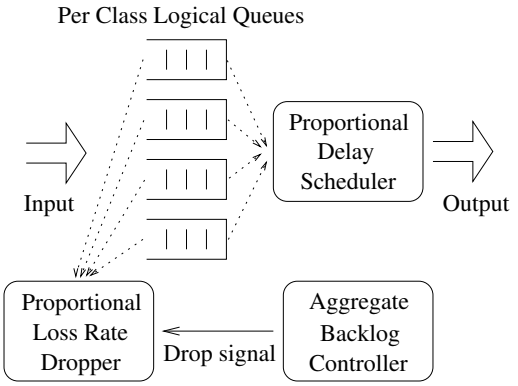


Figure 2: The model of a packet forwarding engine in the proportional differentiation model.

inherit several features from RED, such as probabilistic dropping, gradual increase of the loss probability as the backlog increases, and backlog monitoring through run-time estimators, they share the drawbacks of PBS in controlling the loss rate differentiation between classes. In this work, we attempt to decouple the RED buffer management principles, which refer to *when should a drop occur*, from the characteristics of the dropper, which refer to *which class should a packet be dropped from*.

3 The Proportional Differentiation Model

We consider the following model (see Figure 2) for a packet forwarding engine that offers N classes of service, one for each Class Selector PHB for instance. A logical packet queue is associated with each class. An arriving packet is queued based on its class marking in one of these queues. The *packet scheduler* determines which class should be serviced next, so as to achieve the necessary delay differentiation between classes. Another module, that we refer to as *backlog controller*, monitors the aggregate backlog of waiting packets in the forwarding engine, and decides whether a packet should be dropped. The simplest type of backlog controller is a Drop-Tail buffer manager, in which a packet is dropped when there is no available buffer space. More sophisticated backlog controllers, such as RED [23], can be used instead. When a packet has to be dropped, the *packet dropper* module selects a backlogged queue that the packet will be dropped from. In other words, it is the backlog controller that controls the aggregate backlog in the forwarding engine, while the dropper controls the loss rate differentiation between classes. The dropper removes a packet from the tail of the corresponding queue. Although dropping packets from the front of a queue can lead to lower average queueing delays [24], we have not investigated this optimization here.

The proportional differentiation model states that certain class performance metrics, related to the per-hop queueing delays and packet drops, should be proportional to the corresponding differentiation parameters that the network operator chooses. Through these differentiation parameters the network operator can control the relative quality spacing (ratio) of the offered classes, based on pricing or policy requirements. Specifically, let $\bar{d}_i$ be the average queueing delay of the departed (non-dropped) packets. The proportional delay differentiation model requires that the class average delays are spaced as

$$\frac{\bar{d}_i}{\bar{d}_j} = \frac{\delta_i}{\delta_j} \qquad 1 \le i, j \le N \tag{1}$$

more than a threshold; we think that multiclass RED improves over RIO on this point.

The parameters δ_i are the Delay Differentiation Parameters (DDPs), and they are ordered so that higher classes provide lower delays, i.e., $\delta_1 > \delta_2 > \ldots > \delta_N > 0$. Schedulers that can approximate the proportional delay differentiation model have been studied in [1, 25, 26]; in this paper we use the Waiting Time Priorities (WTP) scheduler of [1]. Similarly, let $\overline{l}_i$ be the average loss rate for class i, i.e., the long-term fraction of class i packets that have been dropped[2]. The proportional differentiation model, in the case of packet drops, requires that the class loss rates are spaced as

$$\frac{\overline{l}_i}{\overline{l}_j} = \frac{\sigma_i}{\sigma_j} \qquad 1 \leq i, j \leq N \tag{2}$$

The parameters σ_i are the Loss rate Differentiation Parameters (LDPs), and they are also ordered as $\sigma_1 > \sigma_2 > \ldots > \sigma_N > 0$. The focus of this paper is the design and evaluation of droppers that can achieve Eq-2. Note that, in this model, higher classes have better performance in terms of both delays and loss rates, i.e., we do not consider two-dimensional services in which a class can have higher delays but lower loss rates, for instance.

4 Proportional Loss Rate (PLR) Droppers

Another way to interpret the proportional differentiation model of Eq-2 is that the *normalized loss rates* $\overline{l}_i/\sigma_i$ should be equal for all classes. The basic idea in the PLR droppers is to keep a running estimate l_i of the loss rate in each class, and when a packet needs to be dropped, to select the *backlogged* class with the minimum l_i/σ_i ratio, i.e., to drop a packet from a backlogged class j with $j = \arg\min_i \frac{l_i}{\sigma_i}$. Dropping a packet from class j reduces the difference of l_j/σ_j from the normalized loss rates of the other classes, tending to equalize them.

Two important questions arise. First, how should the estimate l_i be defined and measured? Second, what happens when the class with the minimum l_i/σ_i ratio is not backlogged when a packet is to be dropped? The first question is the main difference between the two droppers described next. The second question refers to the *feasibility* of the proportional differentiation model. Although this issue will be discussed in Section 5, we note here that under certain conditions the proportional loss rate differentiation model cannot be met, depending on the specified LDPs, the delay differentiation and the corresponding DDPs, the aggregate backlog, and the class load distribution. All these factors can affect the likelihood that a certain class is idle when its loss rate should be increased, and thus, they can affect the feasibility of the proportional loss rate differentiation model.

4.1 PLR(∞): Proportional Loss Rate dropper with infinite memory

In this dropper, the loss rate estimate l_i is the long-term fraction of packets from class i that have been dropped, and it can be measured using simple counters for the arrivals and drops in each class. Since the loss rate is estimated from the entire history of previous arrivals and drops, we say that this dropper has 'infinite' memory. The complete algorithm for PLR(∞) is shown in Figure 3. Note that the selection of the drop-target class is limited to the backlogged classes.

An important issue about the PLR(∞) dropper is the length of the counters A_i and D_i, and how to deal with counter overflows. One approach is to reset all counters, and enter a 'cold-start' phase, when any of the arrival counters overflow. With 32-bit counters, one of the A_i's will overflow after at least four billion packet arrivals. As will be shown in the next section, however, it may be beneficial under certain traffic conditions to reset the counters over shorter intervals, so that the dropper is more adaptive to changing class load distributions.

[2] Another approach would be to define the loss rate based on the fraction of dropped bytes. This metric is not commonly used in practice though.

$B(t)$: Set of backlogged classes at time t

A_i: Counter of packet arrivals in class i

D_i: Counter of packet drops from class i

Packet arrival in class i:
A_i++;
Packet drop from class j at time t:
$j = \arg\min_{i \in B(t)} \frac{D_i}{A_i \sigma_i}$;
D_j++;

Figure 3: Description of the PLR(∞) dropper.

For N classes, the PLR(∞) dropper requires N multiplications (for the $A_i\sigma_i$ terms) and N divisions (for the $\frac{D_i}{A_i\sigma_i}$ terms) every time a packet needs to be dropped; these operations would be, in general, in floating-point arithmetic. An optimization that avoids the multiplications would be to increase A_i by σ_i every time a packet arrives in class i. If a single-precision division takes 40 cycles, the calculation of the normalized loss rates for 8 classes would take 320 cycles, and with a 700MHz CPU the selection of the drop-target class would require about 0.5μs.

Similar droppers have also been studied in [27, 21]. [21] designed a *self-calibrating pushout* dropper, and proposed an implementation that does not require floating-point operations when the arrival rate of each class is policed and known. [27] studied the PLR(∞) dropper in combination with a FCFS scheduler, and proved that this dropper is optimal, in terms of the bandwidth required for providing a certain loss rate to each class.

4.2 PLR(M): Proportional Loss Rate dropper with memory M

The fact that PLR(∞) estimates the class loss rates based on a quite long history of packet arrivals makes it less adaptive to changing class load distributions (see §5). Instead, in the PLR(M) dropper, the loss rate of class i is estimated as the fraction of dropped packets from class i in the last M arrivals. A cyclic queue with M entries, called *Loss History Table (LHT)*, records the class index (LHT[i].class $\in \{1, \ldots, N\}$) and the drop status (LHT[i].drop $\in \{0, 1\}$) of each packet in the last M arrivals. Based on the information stored in the LHT, the dropper knows the number of arrivals $A_i(M)$ and the number of drops $D_i(M)$ from class i in the last M arrived packets. The complete algorithm is shown in Figure 4. Note that the PLR(M) dropper is different from a PLR(∞)-type of dropper in which the counters are reset after every M arrivals, since the PLR(M) dropper attempts to achieve the proportional loss rate differentiation model in *every* time window of M arrivals, while the latter dropper would only attempt to achieve the proportional loss rate differentiation in *successive* time windows of M arrivals.

The PLR(M) dropper requires a packet tag, that is only used internally in the router, recording the LHT entry of the packet. This index is necessary so that the corresponding LHT entry is updated when that packet is dropped. The LHT tags are the main additional implementation complexity of PLR(M) as compared to PLR(∞). It has to be noted, however, that such temporary packet tags, attached to the packet in the router while the packet is being forwarded, are not uncommon in switch and router designs.

LHT: Loss History Table (cyclical queue with M entries)

I: Tail index of LHT

$A_i(M)$: Counter of packets from class i in last M arrivals

$D_i(M)$: Counter of drops from class i in last M arrivals

Packet arrival in class i:

/* k: class number of replaced LHT entry */
k=LHT[I].class; $D_k(M)$−=LHT[I].loss; $A_k(M)--$;
LHT[I].class=i; LHT[I].loss=0; $A_i(M)$++;
Set tag g of arrived packet = I;
I=(I+1) mod M;

Packet drop from class j at time t:

$j = \arg\min_{i \in B(t)} \frac{D_i(M)}{A_i(M)\sigma_i}$;
g = tag of dropped packet;
LHT[g].loss=1;
$D_j(M)$++;

Figure 4: Description of the PLR(M) dropper.

A major question is how large should M be. A first constraint is that M should be large enough so that a dropped packet is always one of the last M arrived packets; otherwise, the LHT-based loss rates may deviate from the actual loss rates. This constraint is met in practice, even for small values of M, when the dropped packets are removed from the tail of the queues. A more tight constraint on M is that it should be large enough so that the specified LDPs are feasible, given the current load distribution and the aggregate loss rate. Intuitively, if M is not large enough, the PLR(M) dropper cannot 'remember' enough previous arrivals and drops in order to adjust the loss rates based on the proportional constraints of Eq-2. In the appendix, we prove that, under stationary conditions, it is necessary that the memory length is at least M_{min} in order for the PLR(M) dropper to meet the proportional loss rate differentiation constraints, where

$$M_{min} = \frac{\sum_{i=1}^{N} \frac{\lambda_i}{\lambda_m} \frac{\sigma_i}{\sigma_m}}{\bar{l}} \quad (3)$$

λ_i is the arrival rate in class i, $\bar{l}$ is the average aggregate loss rate, and $m = \arg\min_{1 \le i \le N}\{\lambda_i \sigma_i\}$. In practice, as will be shown in the next section, the memory length should be much larger than this lower bound in order for the PLR(M) dropper to closely approximate Eq-2.

5 Evaluation of PLR Droppers

The model that we simulate is as in Figure 2. A WTP scheduler serves N queues, one for each class. The backlog controller is Drop-Tail, and the total number of buffers is B. The dropper, PLR(∞) or PLR(M), provides the loss rate differentiation. The DDPs and LDPs, as well as the class load distribution, are given in each graph. The buffer size B is selected so that the average loss rate is in the range 0.1% to 1.5%, which is a common operating regime in the Internet. The traffic is generated from Pareto sources, one for each class, with a scale parameter $\alpha = 1.5$, and so the interarrivals have the 'infinite variance' property. All packets have the same length, and the packet transmission time is normalized to one time unit. The aggregate offered load, which is the inverse of the average interarrival, is denoted by u, and it is given in each graph.

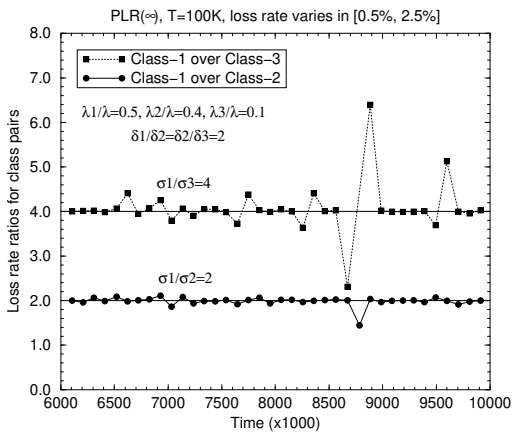
It is noted that these assumptions are not crucial; we have experimented with variable-length packets, RED instead of Drop-Tail, or different source models, and the results follow the same trends. In the measurement of short-term loss rates, a loss rate point is measured for each class in every T packet arrivals; in Figures 5 and 6, T is set to 100,000 packets.

Stationary versus nonstationary load distribution: Figure 5 shows a sample path of the ratios of loss rates for PLR(∞) and PLR(M), in the case of three classes. In these experiments, the average arrival rate in each class remains constant (stationary load distribution). Both droppers closely approximate the specified proportional differentiation constraints σ_i/σ_{i+1}=2 in most time intervals, although a thorough inspection of the sample paths shows that PLR(∞) meets the proportionalities more accurately; this is a general observation as will be shown later.

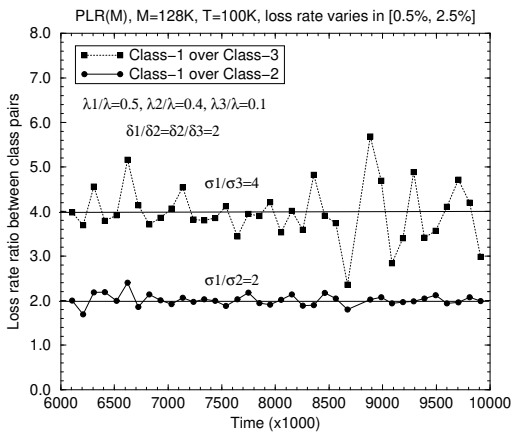
Figure 6 also shows a sample path of the ratios of loss rates for PLR(∞) and PLR(M), but in this case with a nonstationary load distribution. Specifically, at time 7,000,000, the average class load distribution is modified as shown in the graphs. The PLR(M) dropper is unaffected by this load distribution change, since it only 'remembers' the arrivals from each class in the last M=128K packets. The PLR(∞) dropper, on the other hand, is negatively affected by the load distribution change, and the loss rate ratios, especially between classes 1 and 3, diverge from the specified constraint. The reason for this erratic behavior is as follows. Even though, after the load change, all classes have about the same rate, the arrival counters are ratioed based on the long history of previous arrivals, in which Class-1 had five times higher rate than Class-3. The duration of the erratic behavior depends on the size of the arrival counters and the time until the next overflow. Consequently, when the class load distribution is strongly nonstationary, the preferred dropper should be PLR(M); otherwise, the PLR(∞) counters should be set to zero in the same timescales in which the load distribution changes.

Short-term deviations from the proportional differentiation model: In both Figures 5 and 6, the short-term loss rate ratios are not always equal to the specified ratios, and there are occasional 'noise spikes' of significant magnitude. There are two reasons for these deviations. The first reason is related to the operation of the droppers, while the second is related to the feasibility of the proportional model. Over short-term intervals, such as the periodic measurements every T=100,000 packets shown, the arrival rates can deviate significantly from their long-term averages due to traffic burstiness. A dropper that keeps a longer history than T packet arrivals, such as PLR(∞) or PLR(M) with $M > T$, drops packets based on the measured loss rates over a longer history than the last T packets. Consequently, the short-term loss rates, which are measured every T arrivals, can deviate from the proportional differentiation constraints. Regarding the feasibility issue, the PLR droppers cannot drop a packet from a class that is idle, and such a drop may be necessary to achieve the specified proportional loss rate constraints. If most of the drops in an interval of T arrivals occur during a burst in which class i is idle, it will not be possible to equalize the normalized loss rate of class i with that of other classes. Another aspect of the feasibility issue is that if the loss rate in an interval of T arrivals is too small, it may not be possible to drop enough packets from each class so that the loss rates are proportionally adjusted. In either case, the deviations from the proportional loss rate constraints decrease as we increase the 'monitoring timescale' T.

The effect of the memory length in PLR(M): Figure 7 shows the ratio of long-term loss rates, for the case of two classes, as we increase the memory length M of the PLR(M) dropper. The result-

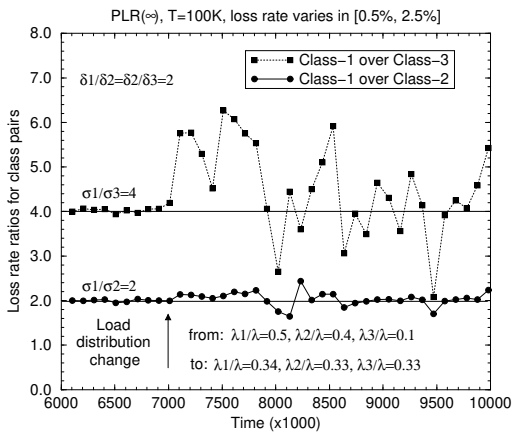


(a) PLR(∞)

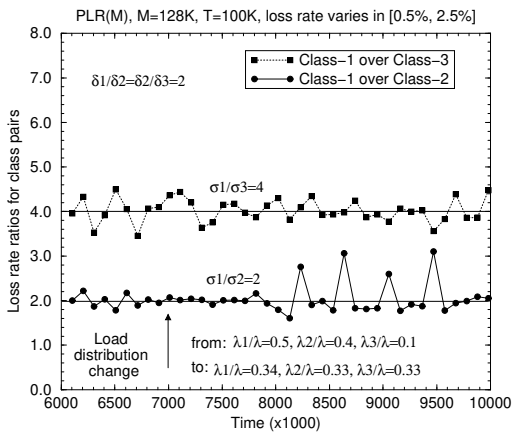


(b) PLR(M), M=128000

Figure 5: A sample path of the loss rate ratios obtained with PLR(∞) and PLR(M) in stationary conditions.

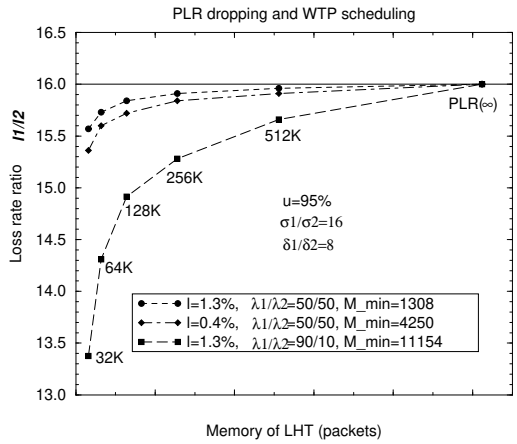


(a) PLR(∞)

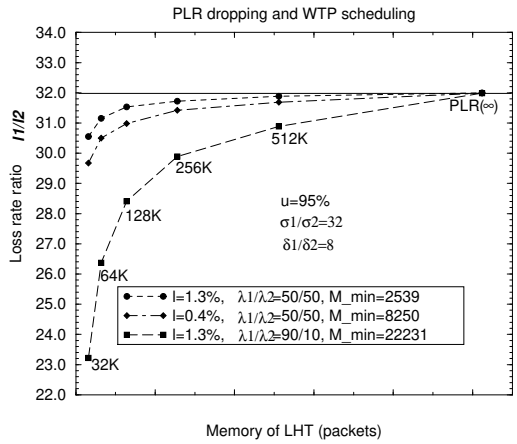


(b) PLR(M), M=128000

Figure 6: A sample path of the loss rate ratios obtained with PLR(∞) and PLR(M) in nonstationary conditions.



(a) $\sigma_1/\sigma_2 = 16$



(b) $\sigma_1/\sigma_2 = 32$

Figure 7: The effect of the memory length M in PLR(M).

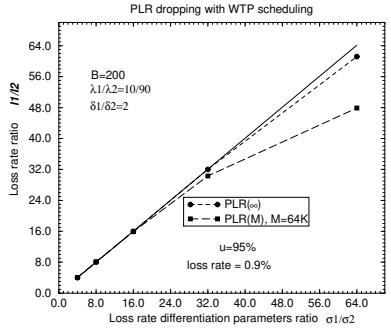
ing ratio with the PLR(∞) dropper is also shown, as the asymptotic case $M = \infty$, providing a comparison between the two droppers. The first observation is that increasing the memory length causes smaller deviations from the proportional differentiation model. The reason is that the loss rate estimates are more robust when they are based on longer sequences of packets arrivals, since the burstiness in the arrival rates and drop times tends to smooth out. A second observation is that the required memory length for meeting the proportional loss rates constraints within a certain error tolerance increases:
a) as the ratio of LDPs increases,
b) as the aggregate loss rate decreases, or
c) as the load ratio between classes increases.
This dependency is also expected from the derivation of the minimum history length M_{min} of Eq-3. A final observation is that the lower bound M_{min} is quite smaller than what is needed in practice for a close approximation of the proportional differentiation model.

Instances of infeasible proportional loss rate differentiation: A PLR dropper deviates from the proportional loss rate differentiation model when the class with the minimum normalized loss rate is not backlogged at the times of packet drops. These deviations can become the common case under several conditions, depending on the specified LDPs, DDPs, average backlog, and class load distribution. To illustrate some of these conditions, as well as the impact of the previous parameters, Figure 8 shows the loss rate ratio for PLR(∞) and PLR(M), in the case of two classes, for several instances of 'hard to achieve' loss rate differentiation. In Figure 8-a, a high ratio of LDPs ($\sigma_1/\sigma_2 = 64$), in combination with a weaker load in Class-1 ($\lambda_1/\lambda_2 = 10/90$), makes the loss rate differentiation infeasible, because there are not enough Class-1 packets to drop in order to make the loss rate in Class-1 64 times larger than the loss rate in Class-2. In Figure 8-b, the scheduler does not introduce sufficiently large delays at the Class-1 packets when the DDP ratio is low, and so Class-1 does not accumulate adequate backlog for the specified loss rate differentiation. A similar scenario can occur if the ratio of DDPs is too high: the highest class will not have adequate backlog, because it is serviced too promptly, and the loss rate differentiation will be higher than what is specified. In Figure 8-c, Class-1 does not have enough arrivals compared to Class-2 when $\lambda_1/\lambda_2 < 0.15$, and so it cannot maintain the required backlog for the given loss rate differentiation. In Figure 8-d, as the aggregate backlog decreases, the likelihood that any class is idle increases. This has a larger impact on the feasibility of the loss rate differentiation model when a class has a much weaker load, as is Class-1 in this case. In general, it is hard to know a priori if the proportional loss rate constraints are feasible, due to the many factors that affect the resulting loss rates. The examples of this paragraph are intended to provide a further intuition on the feasibility issue, rather than to provide quantitative guidelines for the selection of feasible LDPs.

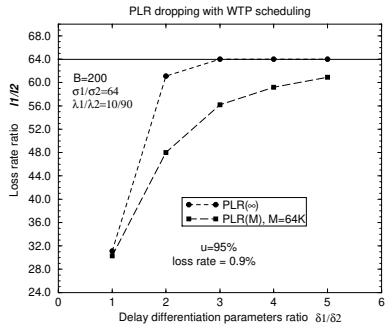
6 Evaluation of WTP Scheduling with Finite Buffers

The Waiting Time Priorities (WTP) scheduler was studied in the context of proportional differentiated services in [1]; the main finding was that WTP approximates the proportional delay differentiation model as the utilization tends to 100% (heavy load conditions). Since that paper did not consider loss differentiation, the evaluation of WTP was based on a lossless model with unlimited buffers. In this section, we extend the results of [1], investigating the delay differentiation that WTP can achieve in the more realistic scenario of finite buffers.

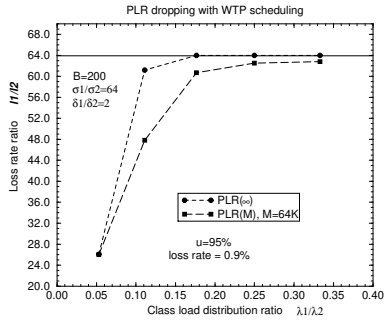
In Figure 9, we show the delay ratios between successive classes i and $i+1$, for increasing values of the average aggregate



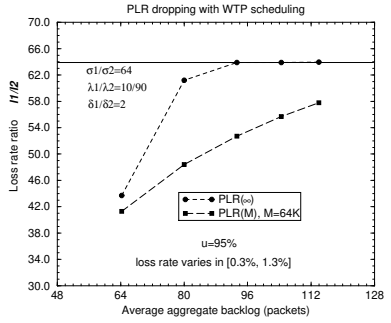
(a) The ratio of LDPs is too high



(b) The ratio of DDPs is too low

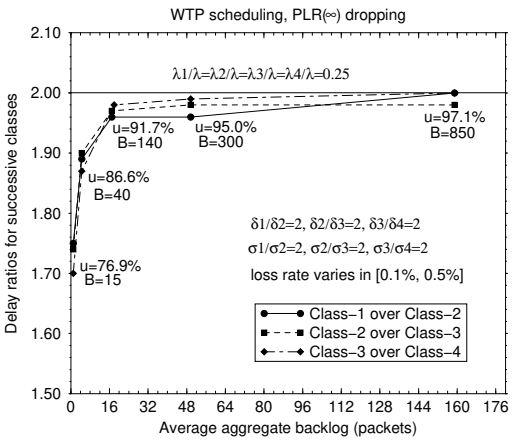


(c) Class-1 is weakly loaded

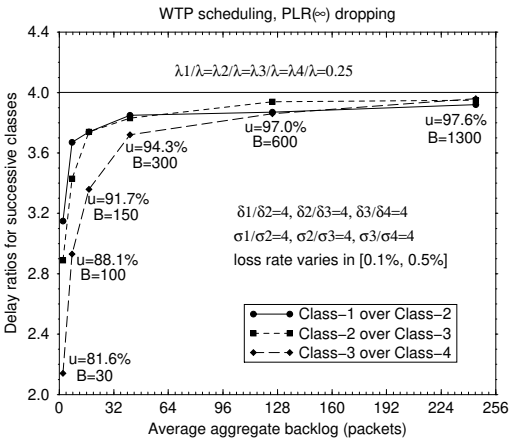


(d) The aggregate backlog is too low

Figure 8: The behavior of PLR(∞) and PLR(M) when the proportional loss rate differentiation model is at the onset of infeasibility.



(a) $\delta_1/\delta_2 = \delta_2/\delta_3 = \delta_3/\delta_4 = 2$



(b) $\delta_1/\delta_2 = \delta_2/\delta_3 = \delta_3/\delta_4 = 4$

Figure 9: Ratios of average delays with WTP over increasing aggregate backlog.

backlog. The buffer space B and the offered load u that lead to the corresponding backlog values are also shown. B was selected in each case so that the loss rate is less than 0.5%. Due to space constraints we only show results for the case of uniform load distribution between classes. Note that these figures are similar to Figure 1 of [1], with the difference that the x-axis here shows the average aggregate backlog in the multiplexer, rather than the utilization. As the aggregate backlog increases, the WTP scheduler tends to the specified proportional delay differentiation, even with finite buffers and lossy conditions. In order for WTP to offer proportional delay differentiation, the aggregate backlog should be sufficiently large compared to the specified delay differentiation. In the example of four classes with DDPs $\delta_i/\delta_{i+1} = 2$, the maximum delay ratio between the highest and the lowest class is 8; in this case, we observe in Figure 9-a that the delay ratios reach their target value within 5% when the aggregate backlog is around 8 packets. When a wider delay spacing between classes is requested, $\delta_i/\delta_{i+1} = 4$, the maximum delay ratio between the highest and lowest class is 64, and the required aggregate backlog for a 5% deviation from the target

delay ratio is around 64 packets (Figure 9-b). It has to be finally noted that an average aggregate backlog in the order of tens or hundreds of packets is not uncommon in overloaded Internet links, or heavily loaded links with bursty traffic, which is also where service differentiation is mostly needed.

7 Differentiation of TCP throughput

Most of the Internet traffic uses TCP as the underlying transport protocol. It is interesting to examine the impact of per-hop proportional delay and loss rate differentiation on the TCP throughput. In this section we attempt a first investigation of this issue for the case of bulk-transfer TCP connections in the congestion avoidance phase. The throughput of such a TCP connection can be approximated by

$$R = \frac{0.93\,M}{T\sqrt{l}} \tag{4}$$

where M is the connection's Maximum Segment Size, l is the average loss rate that the connection encounters, and T is the round-trip delay in the connection's path, *including* queueing delays [28]. We assume that the connection encounters the average delay and loss rate of the path. The round-trip delay, when the connection is in class i, consists of a fixed propagation delay τ and an average queueing delay $\bar{d}_i$, i.e., $T_i = \tau + \bar{d}_i$. Also, let $\bar{l}_i$ be the end-to-end loss rate in class i. If $\bar{d}_i^k$ and $\bar{l}_i^k$ are the average queueing delay and the loss rate, respectively, in class i at the k'th hop we have that $\bar{d}_i = \sum_k \bar{d}_i^k$, and $\bar{l}_i = 1 - \prod_k (1 - \bar{l}_i^k)$. If $\sum_k \bar{l}_i^k$ is less than 1% or so, then $\bar{l}_i \approx \sum_k \bar{l}_i^k$.

Suppose now that every hop along the connection's path offers proportional delay and loss rate differentiation with the same DDPs and LDPs. Then, $\bar{d}_j^k = \delta \bar{d}_i^k$, and $\bar{l}_j^k = \sigma\, \bar{l}_i^k$ where $\delta = \delta_j/\delta_i$ and $\sigma = \sigma_j/\sigma_i$; if $i > j$, $\delta > 1$ and $\sigma > 1$. So, the ratio of the throughput that the connection would get in class i over class j is

$$\frac{R_i}{R_j} = \sqrt{\sigma}\,\frac{\delta + \tau/\bar{d}_i}{1 + \tau/\bar{d}_i} \tag{5}$$

This shows that the proportional loss rate differentiation parameter σ affects the TCP throughput ratio between classes in a square-root law. The effect of the delay differentiation parameter, on the other hand, depends on the relative magnitude of the queueing delays compared to the propagation delay of the path. If the propagation delay is small compared to the queueing delay ($\frac{\tau}{\bar{d}_i} \ll 1 < \delta$), the delay differentiation affects the TCP throughput ratio between classes linearly. When the connection's propagation delay is large compared to the queueing delays, the impact of the delay differentiation can be much smaller. As an example, suppose that $\delta = \sigma = 2$, and that the ratio of the propagation delay versus the class i queueing delay is one. The TCP throughput in class i will be approximately 2.1 times higher than in class j, and the impact of the loss rate differentiation on the throughput ratio will be about the same with the impact of the delay differentiation. If the ratio of the propagation delay versus the class i queueing delay is three (say in an cross-continental connection), the TCP throughput in class i will be about 1.8 times higher than in class j, and the contribution of the loss rate differentiation on the TCP throughput ratio will be about 80%.

8 Conclusions

In this paper, we extended the proportional differentiation model in the direction of loss rate differentiation. The main contribution was the design and evaluation of two droppers that can approximate this model. The PLR(∞) dropper meets the proportional loss rate constraints more accurately and it is simpler to implement. The PLR(M) dropper, on the other hand, requires that each packet is tagged with its LHT index and is less accurate than PLR(∞), but it is more adaptive to nonstationary traffic loads. The final choice between the two droppers should be based on these considerations. An interesting direction for further research would be to analytically derive the class loss rates in a forwarding engine that combines a dynamic priority scheduler, such as WTP, and a measurement-based dropper, such as PLR(∞) or PLR(M). Such a result, even for a simplistic traffic model, would show more explicitly the impact of the different parameters on the final loss rate differentiation.

Appendix

Suppose that the proportional loss rate constraints of Eq-2 are met. If A_i and D_i are the number of arrivals and drops, respectively, from class i, the ratio of packet drops for any pair of classes i and j is

$$\frac{D_i}{D_j} = \frac{\sigma_i}{\sigma_j}\frac{A_i}{A_j} \tag{6}$$

Under stationary conditions, the ratio of arrivals is equal to the ratio of the corresponding arrival rates, and so,

$$\frac{D_i}{D_j} = \frac{\sigma_i}{\sigma_j}\frac{\lambda_i}{\lambda_j} \tag{7}$$

Consequently, the class with the minimum number of drops is class m, where

$$m = \arg \min_{1 \le i \le N} \{\lambda_i \sigma_i\} \tag{8}$$

Let us now derive the minimum number of arrivals, given an aggregate average loss rate $\bar{l}$, so that all the $N-1$ proportional loss rate constraints of Eq-2 are satisfied. The minimum number of required arrivals leads to the minimum number of drops. If class m, which has the minimum number of drops, gets only one packet drop ($D_m = 1$), class i will have

$$D_i = D_m \frac{\sigma_i}{\sigma_m}\frac{A_i}{A_m} = \frac{\sigma_i}{\sigma_m}\frac{\lambda_i}{\lambda_m} \tag{9}$$

drops, and so the total number of drops from all classes will be

$$D = \sum_{i=1}^{N} D_i = \sum_{i=1}^{N} \frac{\sigma_i}{\sigma_m}\frac{\lambda_i}{\lambda_m} \tag{10}$$

and since the aggregate loss rate is $\bar{l}$, the required number of arrivals will be

$$A = \frac{\sum_{i=1}^{N} \frac{\sigma_i}{\sigma_m}\frac{\lambda_i}{\lambda_m}}{\bar{l}} \tag{11}$$

This is the minimum number of arrivals that the PLR(M) should remember, in order for the $N-1$ proportional loss rate constraints of Eq-2 to hold, and so, this is the minimum memory length M_{min},

$$M \ge M_{min} = \frac{\sum_{i=1}^{N} \frac{\sigma_i}{\sigma_m}\frac{\lambda_i}{\lambda_m}}{\bar{l}} \tag{12}$$

In practice, due to the burstiness of the packet arrivals and packet drops, the memory length should be much larger than M_{min} in order for PLR(M) to closely approximate the constraints of Eq-2.

References

[1] C.Dovrolis, D.Stiliadis, and P.Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," in *ACM SIGCOMM*, Sept. 1999.

[2] S.Blake, D.Black, M.Carlson, E.Davies, Z.Wang, and W.Weiss, *An Architecture for Differentiated Services*, Dec. 1998. IETF RFC 2475.

[3] K.Nichols, V.Jacobson, and L.Zhang, *A Two-bit Differentiated Services Architecture for the Internet*, July 1999. IETF RFC 2638.

[4] D. D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 362–373, Aug. 1998.

[5] M.May, J.C.Bolot, C.Diot, and A.Jean-Marie, "Simple Performance Models of Differentiated Services Schemes for the Internet," in *Proceedings IEEE INFOCOM*, Mar. 1999.

[6] J. S.Sahu, D.Towsley, "A Quantitative Study of Differentiated Services for the Internet," in *Proceedings Global Internet Symposium*, Dec. 1999.

[7] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "Adaptive Packet Marking for Maintaining End-to-End Throughput in a Differentiated-Services Internet," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 685–697, Oct. 1999.

[8] I.Stoika, S.Shenker, and H.Zhang, "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," in *Proceedings ACM SIGCOMM*, Sept. 1998.

[9] I.Stoika and H.Zhang, "Providing Guaranteed Services Without Per Flow Management," in *Proceedings ACM SIGCOMM*, Sept. 1999.

[10] C.Cetinkaya and E.W.Knightly, "Egress Admission Control," in *Proceedings INFOCOM*, Mar. 2000.

[11] J. L. Boudec, M. Hamdi, L. Blazevic, and P.Thiran, "Asymmetric Best Effort Service for Packet Networks," in *Proceedings Global Internet Symposium*, Dec. 1999.

[12] A.Charny, "Delay Bounds in a Network with Aggregate Scheduling," Tech. Rep. Draft version 3, Cisco Systems, Feb. 2000.

[13] I.Stoika and H.Zhang, "LIRA: An Approach for Service Differentiation in the Internet," in *Proceedings NOSSDAV*, 1998.

[14] I.Yeom and A. N. Reddy, "Modeling TCP Behavior in a Differentiated Services Network," tech. rep., Texas A&M University, Feb. 2000.

[15] S.Sahu, P.Nain, D.Towsley, C.Diot, and V.Firoiu, "On Achievable Service Differentiation with Token Bucket Marking for TCP," in *Proceedings ACM SIGMETRICS*, June 2000.

[16] C.Dovrolis and P.Ramanathan, "A Case for Relative Differentiated Services and the Proportional Differentiation Model," *IEEE Network*, Oct. 1999.

[17] K.Nichols, S.Blake, F. Baker, and D.L.Black, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, Dec. 1998. IETF RFC 2474.

[18] A.-M. Lin and J.A.Silvester, "Priority Queueing Strategies and Buffer Allocation Protocols for Traffic Control at an ATM Integrated Broadband Switching System," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1524–1536, Dec. 1991.

[19] H.Kroner, G.Hebuterne, P.Boyer, and A.Gravey, "Priority Management in ATM Switching Nodes," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 418–427, Apr. 1991.

[20] A.K.Choudhury and E.L.Hahne, "Dynamic queue length thresholds for multipriority traffic," in *15th International Teletraffic Congress*, June 1997.

[21] H.J.Chao, H.Cheng, Y-R.Jeng, and D.Jeong, "Design of a generalized priority queue manager for ATM switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 867–879, June 1997.

[22] B.Suter, T.V.Lakshman, D.Stiliadis, and A.K.Choudhury, "Buffer Management Schemes for Supporting TCP in Gigabit Routers with Per-Flow Queueuing," *IEEE Journal on Selected Areas of Communications*, Sept. 1999.

[23] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, Aug. 1993.

[24] N.Yin and M.G.Hluchyj, "Implication of Dropping Packets from the Front of a Queue," *IEEE Transactions on Communications*, vol. 41, pp. 846–851, June 1993.

[25] T.Nandagopal, N.Venkitaraman, R.Sivakumar, and V.Bharghavan, "Delay Differentiation and Adaptation in Core Stateless Networks," in *Proceedings IEEE INFOCOM*, Mar. 2000.

[26] Y.Moret and S.Fdida, "A Proportional Queue Control Mechanism to Provide Differentiated Services," in *International Symposium on Computer and Information Systems (ISCIS)*, Oct. 1998.

[27] T.Yang and J.Pan, "A Measurement-Based Loss Scheduling Scheme," in *Proceedings INFOCOM*, 1996.

[28] M. Mathis, J. Semke, and J. Madhavi, "The macroscopic behavior of the TCP congestion avoidance algorithm," *ACM Computer Communications Review*, vol. 27, pp. 67–82, July 1997.