Proportional Differentiated Services: Loss Differentiation and Packet Dropping

Constantinos Dovrolis
University of Delaware
dovrolis@cis.udel.edu

Parameswaran Ramanathan
University of Wisconsin
parmesh@ece.wisc.edu

ABSTRACT

The proportional differentiation model provides the network operator with the 'tuning knobs' for adjusting the per-hop QoS spacing between classes, independent of the class loads. This paper applies the proportional model in the differentiation of class loss rates, and investigates appropriate buffer management and packet dropping mechanisms. The first part of this work [13] considered the corresponding delay differentiation and packet scheduling problem.

According to the Proportional Loss Differentiation (PLD) model, the loss rate ratios between classes are fixed, based on certain parameters that the network operator specifies. We derive the loss rate in each class under the PLD constraints, and state the dynamics and feasibility conditions of the model. We then consider two Proportional Loss Rate (PLR) dropping algorithms. The two droppers, *PLR(∞)* and *PLR(M)*, differ in the interval over which the loss rates are measured and proportionally adjusted. PLR(∞) can achieve the PLD constraints when the model is feasible. PLR(M), on the other hand, is not as accurate as PLR(∞) in meeting the PLD model, but it should be preferred when the class load distribution varies with time. We derive a lower bound on the 'history length' of the PLR(M) dropper, under stationary conditions for the aggregate loss rate and class load distribution. Finally, we examine the coupled effect of delay and loss rate proportional differentiation on the throughput of bulk-transfer TCP connections.

Parts of this paper were first published at the IEEE/IFIP International Workshop on Quality of Service (IWQoS) 2000. The work of C.Dovrolis and P.Ramanathan reported in this paper was supported in part by the National Science Foundation under Grant No. MIP-9526761, and by Cisco Systems Inc.

1 Introduction

Providing some kind of service differentiation in the Internet has been a problem of enormous commercial and research importance in at least the last ten years (see [16] for some recent studies). The Differentiated Services (DiffServ) architecture [1] was proposed as a more scalable solution to this problem, compared to previous approaches such as the Integrated Services (IntServ) architecture [36]. The DiffServ vision is that stateless priority mechanisms at the network core can be combined with stateful mechanisms at the network edges, in order to construct versatile end-to-end services. Among the first DiffServ proposals was the Virtual Leased Line (VLL) service [19] and the Assured service [8]. Recent findings, however, have raised concerns about the effect of aggregation on the VLL service [7], and about the provisioning complexity of the Assured service [30, 29]. Several other models for scalable differentiated services have been proposed. To name a few, the *SCORE* architecture can provide per-flow service guarantees without per-flow state in the core routers [31, 32]. [5] proposes a core-stateless scheme in which resource management and admission control are performed only at egress routers. The Asymmetric Best Effort [4] provides two service classes: one for delay-sensitive applications and another for throughput-sensitive applications.

A fundamentally different approach in the DS framework is the *relative differentiated services*. 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* [10]. The eight *Class Selector PHBs*, standardized by the IETF [25], follow the relative service differentiation model. In this context, applications and users do not get an absolute service level assurance, such as an end-to-end delay bound or throughput, since there is no admission control and resource reservations. Instead, the network assures that higher classes will offer better QoS 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 [11]. From this point of view, the relative service differentiation model follows the architectural principle of *end-system adaptation*, since it provides the class selection as an additional dimension in the end-system adaptation space.

There are several ways in which a network can provide relative differentiated services. For example, the differentiation can be based on appropriate pricing (i.e., higher classes are more expensive) [26], or on capacity provisioning (i.e., higher classes get more bandwidth relative to their expected loads) [17]. Such mechanisms, however, cannot always provide consistent class differentiation, because the relative QoS differentiation between classes varies with the class loads. Other mechanisms, such as strict class prioritization, provide consistent differentiation that does not depend on the load variations, but they do not allow the network operator to adjust the QoS spacing between classes.

Such 'tuning knobs' are necessary in a practical setting, since the network operators must be able to adjust the class spacing based on pricing and policy objectives. Our basic premise, starting from these limitations of other relative differentiation models, is that a relative differentiated services architecture has to be:

- *Predictable*, in the sense that the differentiation should be consistent (i.e., higher classes are better, or at least no worse) independent of the class load variations, and

- *Controllable*, meaning that the network operators should be able to adjust the QoS spacing between classes based on their criteria.

As a target for predictable and controllable relative differentiation, we consider the *proportional differentiation model* [10]. According to this model, the basic performance measures for packet forwarding locally at each hop are ratioed proportionally to certain *class differentiation parameters* that the network operator chooses. Even though there is no wide consensus on the most appropriate performance measures for packet forwarding, it is generally agreed that a better network service means lower queueing delays and lower likelihood of packet losses. The first part of this work [13] focused on delay differentiation and packet scheduling. In this paper, we extend the same proportional model in the case of loss rate differentiation, considering a joint model of proportional delay and loss rate differentiation.

In the first part of the paper, we propose and study the Proportional Loss Differentiation (PLD) model. According to PLD, the loss rate ratios between classes are fixed, based on certain parameters that the network operator specifies. The PLD model provides a *differentiation invariant* that does not depend on the aggregate load or on the class load distribution. The PLD model is similar in several aspects with the Proportional Delay Differentiation (PDD) model of [13]. One similarity is that the PLD model cannot be feasible in all operating conditions either. The feasibility of the model can be determined from the loss rates that result with the Strict Priorities dropper.

In the second part of the paper, we focus on the design of two Proportional Loss Rate (PLR) droppers for the PLD model. The two droppers, *PLR(∞)* and *PLR(M)*, differ in the interval over which the loss rates are measured and proportionally adjusted. PLR(∞) can achieve the PLD constraints when the model is feasible. In addition, it approximates the PLD model quite well in short timescales, when the loss rates are measured in every 100,000 packets or more. PLR(M), on the other hand, is not as accurate as PLR(∞) in meeting the PLD model, but it also approximates the PLD model in short timescales when the loss rates are measured in every 100,000 packets or more (given that M is also in the same range). The PLR(∞) dropper is simpler to implement

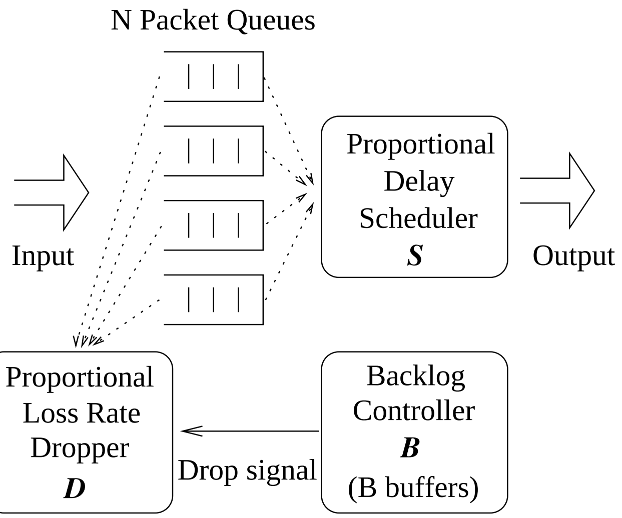


Figure 1: The model of a lossy packet multiplexer that we consider in this paper.

than PLR(M). PLR(M) should be preferred over PLR(∞), however, when the class load distribution varies with time. In such conditions, PLR(∞) can deviate for large time intervals from the PLD constraints. PLR(M), instead, adapts quickly to the new load conditions. We derive a lower bound on the 'history length' M of the PLR(M) dropper, under stationary conditions for the aggregate loss rate and class load distribution. 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. In Section 2, we study the proportional loss rate differentiation model, stating its dynamic properties and feasibility conditions. Section 3 presents the two PLR droppers, while Section 4 evaluates and compares them. Section 5 examines the differentiation that results in the throughput of bulk TCP connections, with coupled proportional delay and loss rate differentiation. Section 6 reviews previous buffer management and dropping mechanisms, and discusses recent contributions in the area of proportional loss rate differentiation. We conclude in Section 7.

2 Proportional Loss Differentiation (PLD) model

We consider the following queueing model of a packet multiplexer (or 'link') (see Figure 1). A non-preemptive link with capacity C (bytes per second) services N First-Come First-Served (FCFS) queues, one for each traffic class[1]. The N queues share the same pool of B packet buffers. A *scheduler* $\mathcal{S}$ determines the order in which packets are selected for transmission, providing a certain

[1]We use the terms 'class' and 'queue' interchangeably.

delay differentiation between classes. The backlog controller $\mathcal{B}$ determines the time instants that a packet should be dropped, either because there is no available buffer, or because the link uses an active buffer management mechanism [24]. When a packet has to be dropped, the backlog controller signals the dropper to select the backlogged class that the packet should be removed from. Consequently, $\mathcal{B}$ determines the aggregate amount of losses, as well as the time instants that losses occur, while $\mathcal{D}$ determines the distribution of losses between classes, i.e., the loss differentiation.

The simplest type of backlog controller is called *Drop-Tail* and it signals a packet drop when there is no available buffer to store an arriving packet. More sophisticated algorithms, such as RED [15] and its variations [14, 27], are examples of other backlog controllers. An example of a simple dropper is the *Last-In First-Out (LIFO)* algorithm, which removes the last packet that entered the link, independent of its class. LIFO corresponds to the FCFS scheduler, since both schemes do not discriminate between classes. An important dropper is the *Strict Priority (SP)* algorithm, which removes the packet from the tail of the lowest priority backlogged class. The SP dropper corresponds to the SP scheduler, since both schemes create the strongest possible discrimination in favor of higher classes. As shown later in this section, the SP dropper is directly related to the feasibility of the PLD model.

We assume that the scheduler $\mathcal{S}$ is work-conserving, and that the packet size distribution is the same in all classes, with an average packet size $\bar{L}$=1. In addition, we make the following assumptions for the backlog controller and dropper. First, *a backlogged packet requires one buffer independent of the packet size.* This assumption simplifies the queueing model, because an arriving packet can cause only one packet drop. Second, we assume that *the dropper can perform pushout*, i.e., to drop a packet that is already backlogged, as opposed to drop arriving packets only. In the past, it was often assumed that pushout is hard to implement in high-speed links [20]. Recently, however, it has been shown with actual implementations of high-performance routers that pushout can be efficient [6, 33], when packets are removed from the head or the tail of the class queues. Third, we consider droppers that *remove packets from the tail of the class queues.* Even though removing packets from the head of a queue leads to lower average queueing delays [38], this optimization would make the delay differentiation between classes dependent not only on the scheduler $\mathcal{S}$, but also on the dropping algorithm $\mathcal{D}$; we chose to avoid this kind of coupling between delay and loss differentiation. Fourth, we limit our attention to a *Drop-Tail backlog controller.* The Drop-Tail backlog controller forces the drop of a backlogged packet when a packet arrives, if the number of backlogged packets is equal to the number of buffers B. This is a necessary assumption for the loss rate conservation law (Equation 5), and for the dynamics and feasibility of the PLD model that follow next. It is not a requirement though, for the operation of the droppers in §3.

The loss-related performance metric that the differentiation is based on is the average loss rate, or simply the *loss rate*. The loss rate $\bar{l}_i$ in class i is defined as *the fraction of class i packets that have been dropped*. So, if D_i packets have been dropped out of A_i arrivals in class i, the loss rate $\bar{l}_i$ is D_i/A_i. Similarly, the *aggregate loss rate* $\bar{l}_{ag}$ in the link is defined as the fraction of arrived packets that have been dropped, independent of their class, i.e., $\bar{l}_{ag} = D/A = \frac{\sum_i D_i}{\sum_i A_i}$. Another loss-related performance metric is the *fraction of dropped bytes*, instead of packets, but this metric is not as commonly used in practice.

Let λ_i be the *average rate* in class i (packets per second), and $\lambda{=}\sum_{i=1}^{N} \lambda_i$ the average rate in the link. In the presence of packet drops, we have to distinguish between the *offered rate* λ_i^o and the *accepted rate* λ_i^a. The later is the fraction of the offered rate that corresponds to *serviced (i.e., non-dropped)* packets,

$$\lambda_i^a = (1 - \bar{l}_i)\, \lambda_i^o \qquad i = 1 \ldots N \tag{1}$$

The offered rate can be larger than the capacity C, while the utilization of the server, defined as $u{=}\bar{L}\lambda_i^a/C$, is always $u < 1$. In the following, when we refer to a class rate λ_i or to the aggregate link rate λ, we mean the corresponding accepted rate. Also, the average queueing delay $\bar{d}_i$ of packets in class i refers to serviced (i.e., non-dropped) packets.

The *Proportional Loss Differentiation (PLD)* model states that the loss differentiation between classes has to follow certain proportional constraints, independent of the aggregate load or the class load distribution. These proportional constraints are controlled with parameters that the network operator specifies. Formally, the PLD model requires that the ratio of loss rates between two classes i and j is set to the ratio of the corresponding *Loss Differentiation Parameters (LDPs)*,

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

where $\sigma_1 > \sigma_2 > \ldots > \sigma_N > 0$ are the LDPs. We choose Class-1 to be the 'reference class' and set $\sigma_1{=}1$. Then, the PLD model requires that the loss rate in each class i is a certain fraction σ_i of the loss rate in Class-1, i.e.,

$$\bar{l}_i = \sigma_i\, \bar{l}_1 \qquad i = 2 \ldots N \tag{3}$$

independent of the aggregate load, or the class load distribution.

In this paper, we consider a joint delay and loss rate differentiation model. Consequently, in addition to the previous PLD constraints, we also impose the following Proportional Delay Differentiation

(PDD) constraints

$$\frac{\bar{d}_i}{\bar{d}_j} = \frac{\delta_i}{\delta_j} \qquad 1 \leq i, j \leq N \tag{4}$$

where $\delta_1 = 1 > \delta_2 > \ldots > \delta_N > 0$ are the Delay Differentiation Parameters (DDPs). The PDD constraints, as well as scheduling mechanisms that can achieve them when they are feasible, are studied in detail in [13].

2.1 Per-class loss rates in the PLD model

Given the assumptions of the previous paragraph for the scheduler $\mathcal{S}$, the backlog controller $\mathcal{B}$, and the dropper $\mathcal{D}$, we can now state the following conservation law. Specifically, the *conservation law for loss rates* states that a dropper $\mathcal{D}$ must satisfy the following equation

$$\sum_{i=1}^{N} \lambda_i \bar{l}_i = \lambda \bar{l}_{ag} \tag{5}$$

where $\bar{l}_i$ is the loss rate in class i with $\mathcal{D}$, and $\bar{l}_{ag}$ is the aggregate loss rate with a LIFO dropper [22]. The conservation law implies that even though a dropper $\mathcal{D}$ can affect the relative magnitude of the class loss rates, making the loss rate in one class lower than the loss rate in another class, this is a 'zero-sum' game because the weighted sum of (5) has to be equal to the loss rate $\bar{l}_{ag}$ of the aggregate traffic stream. Note that $\bar{l}_{ag}$ is independent of the class that packets belong to, and thus, of the scheduler $\mathcal{S}$ or the dropper $\mathcal{D}$. The aggregate loss rate $\bar{l}_{ag}$ depends only on the arriving traffic stream, on the aggregate backlog controller $\mathcal{B}$ (and thus, on the number of buffers B), and on the link capacity C.

From the PLD model of (2) and the loss rate conservation law (5), it is easy to show that the loss rate in each class i is

$$\bar{l}_i = \frac{\sigma_i \bar{l}_{ag}}{\sum_{n=1}^{N} \sigma_n \lambda_n} \qquad i = 1 \ldots N \tag{6}$$

Equation (6) shows that even though the PLD model consists of N-1 *relative* constraints on the class loss rates, the loss rate in each class is uniquely determined when the PLD model is applied to a traffic stream with a certain load distribution $\{\lambda_i\}$ and a certain aggregate loss rate $\bar{l}_{ag}$.

2.2 Loss rate dynamics in the PLD model

Based on Equation (6), we can state the following properties for the *loss rate dynamics in the PLD model*. The proofs of these properties follow similar derivations with the proofs of the delay dynamics in the PDD model (§2.2 of [13]), and they are not included here.

Property 1: Increasing the input rate of a class, increases (in the wide sense) the loss rate of all classes.

Property 2: Increasing the rate of a higher class causes a larger increase in the class loss rates than increasing the rate of a lower class.

Property 3: Decreasing the loss differentiation parameter of a class decreases (in the wide sense) the loss rate of that class, and increases (in the wide sense) the loss rate of all other classes.

The following two properties are important in the Dynamic Class Selection (DCS) framework [11], in which users with absolute end-to-end performance constraint search for the minimum acceptable class. The first property states that when one ore more users move to a higher class, the loss rate in all classes increases. The loss rates decrease, on the other hand, when one or more users move to a lower class. The second property shows that when a user switches from one class to another, the user still observes a consistent class ordering, i.e., the higher class provides a lower loss rate.

Specifically, suppose that the class load distribution changes from $\{\lambda_n\}$ to $\{\lambda'_n\}$, with $\lambda'_i = \lambda_i - \epsilon$, $\lambda'_j = \lambda_j + \epsilon$, and $\lambda'_k = \lambda_k$ for all $k \neq i, j$ $(\epsilon > 0)$. Let $\bar{l}'_n$ be the loss rate in class n when the class load distribution is $\{\lambda'_n\}$.

Property 4: If $i > j$ then $\bar{l}'_n \leq \bar{l}_n$ for all $n = 1 \ldots N$. Similarly, if $i < j$ then $\bar{l}'_n \geq \bar{l}_n$.

Property 5: If $i > j$ then $\bar{l}'_j \geq \bar{l}_i$. Similarly, if $i < j$ then $\bar{l}'_j \leq \bar{l}_i$.

2.3 Feasibility of the PLD model

The PLD model may not be always feasible. Given the load distribution and the aggregate loss rate, the PLD model does not only specify N-1 loss rate ratios, but it also specifies the N loss rates of Equation (6). It may not be possible, though, to enforce a certain loss rate in each class. To see why, suppose that the dropper attempts to provide a very low loss rate to a class k. It can happen, though, that the only backlogged class k at the drop time instants is class k, and so the dropper will be forced to drop only class k packets. This scenario shows that the loss rate in class k cannot be arbitrarily manipulated.

Formally, given a traffic stream with class rates $\{\lambda_i\}$, a certain number of buffers B in a Drop-Tail backlog controller, an aggregate loss rate $\bar{l}_{ag}$, and a particular scheduler $\mathcal{S}$, we say that a set of LDPs $\{\sigma_i,\, i = 2 \ldots N\}$ is feasible if there exists a dropper $\mathcal{D}$ that can set the loss rate in each class as in (6). So, the set of LDPs $\{\sigma_i,\, i = 2 \ldots N\}$ is feasible if and only if the set of loss rates $\{\bar{l}_i = \sigma_i \bar{l}_{ag}/(\sum_{n=1}^{N} \lambda_n \sigma_n),\, i = 1 \ldots N\}$ is feasible. Note that the feasibility of the LDPs depends on the scheduler $\mathcal{S}$, since $\mathcal{S}$ affects the service that each class receives, and so the likelihood that a class is backlogged at the time instant of a packet drop.

To the best of our knowledge, there are no results in the queueing theory literature for the feasibility of a set of class loss rates, similar to the feasibility conditions by Coffman and Mitrani [9] or Regnier [28] for the feasibility of a set of class average delays (also see §2.3 of [13]). The basic intuition behind those delay-related feasibility conditions, however, also holds in the case of loss rates. Specifically, the loss rate of a class (or of any set of classes) has a lower bound due to the inherent load in that class (or set of classes), given a certain scheduler $\mathcal{S}$ and a Drop-Tail backlog controller with B buffers. This lower loss rate bound would result in practice if that class (or set of classes) was given strictly higher priority (i.e., minimum losses) over the rest of the traffic.

In the following, we state a set of feasibility conditions for the N loss rates $\{\bar{l}_i\}$, similar to the Regnier feasibility conditions for the case of class average queueing delays [28]. Our conjecture is that the set of N class loss rates $\{\bar{l}_i\}$ is feasible if and only if the following N-1 conditions are true,

$$\sum_{i=k}^{N} \lambda_i \, \bar{l}_i \geq \bar{l}_{k,N}^{SP} \sum_{i=k}^{N} \lambda_i \qquad k = 2 \ldots N \tag{7}$$

where $\bar{l}_{k,N}^{SP}$ is the loss rate that would be encountered by classes $\{k, \ldots, N\}$ with an SP dropper, if these classes were given strictly higher priority over the rest of the traffic in terms of packet drops. These conditions impose a lower bound on the number of losses in the N-1 subsets of the k highest classes ($k = 1 \ldots N-1$).

Let $\bar{l}_i^{SP}$ be the loss rate in Class-i with an SP dropper that drops the lowest backlogged class first. With this notation, we have that $\bar{l}_{k,N}^{SP} \sum_{i=k}^{N} \lambda_i = \sum_{i=k}^{N} \lambda_i \bar{l}_i^{SP}$, and so the feasibility inequalities become

$$\sum_{i=k}^{N} \lambda_i \, \bar{l}_i \geq \sum_{i=k}^{N} \lambda_i \bar{l}_i^{SP} \qquad k = 2, \ldots, N \tag{8}$$

The conditions of (8) show that the feasibility of the PLD model depends on the loss rates that would occur with the same scheduler and number of buffers, if the dropper $\mathcal{D}$ was replaced by the Strict Priority (SP) dropper. In the case of N=2 classes, a certain LDP σ_2 ($\sigma_2 < \sigma_1 = 1$), is feasible if and only if

$$\sigma_2 \geq \frac{\bar{l}_2^{SP}}{\bar{l}_1^{SP}} \quad \text{or} \quad \frac{1}{\sigma_2} \leq \frac{\bar{l}_1^{SP}}{\bar{l}_2^{SP}} \tag{9}$$

So, *a given loss rate ratio $\sigma_2 < 1$ between Class-2 and Class-1 is feasible if and only if the corresponding loss rate ratio in the SP dropper is not higher than σ_2.* When $N > 2$, the feasibility conditions for the PLD model become the following N-1 inequalities

$$\sum_{i=k}^{N} \lambda_i \sigma_i \geq \frac{S}{\bar{l}_{ag}} \sum_{i=k}^{N} \lambda_i \bar{l}_i^{SP} \qquad k = 2 \ldots N \tag{10}$$

where $S = \sum_{i=1}^{N} \lambda_i \sigma_i$.

In order to examine the feasibility of the PLD model we need to know the loss rates $\bar{l}_k^{SP}$ that would result with the SP dropper. In practice, the network operator can measure these terms by either deploying SP on the link for short time intervals and measuring the loss rates $\bar{l}_k^{SP}$, or by emulating the SP dropper with the traffic workload of the link.

3 Proportional Loss Rate (PLR) droppers

In this section, we describe two dropping algorithms, called *Proportional Loss Rate (PLR)* droppers, that aim to meet the PLD model of §2 when the specified LDPs are feasible. Another way to state the PLD model is that the *normalized loss rates*, defined as $\tilde{l}_i = \bar{l}_i / \sigma_i$, must be equal in all classes, i.e.,

$$\tilde{l}_i = \frac{\bar{l}_i}{\sigma_i} = \frac{\bar{l}_j}{\sigma_j} = \tilde{l}_j \qquad 1 \leq i, j \leq N \tag{11}$$

The PLR droppers maintain a running estimate $\tilde{l}_i(t)$ of the normalized loss rate in each class i. When the backlog controller requires a packet to be dropped, the PLR droppers select *the backlogged class j with the minimum normalized loss ratio.* Dropping a packet from class j increases $\bar{l}_j$, and hence it reduces the difference of $\bar{l}_j / \sigma_j$ from the normalized loss rates of the other classes. The objective is to equalize the normalized loss rates in the long run, and thus to meet the proportional loss differentiation constraints.

$B(t)$: Set of backlogged classes at time t
A_i: Count of packet arrivals in class i up to time t
D_i: Count of packet drops from class i up to time t

Packet arrival in class i at time t:
$A_i = A_i + 1$;

Packet drop from class j at time t:
$j = \arg\min_{i \in B(t)} \frac{D_i}{A_i \sigma_i}$;
$D_j = D_j + 1$;

Figure 2: Description of the PLR(∞) dropper.

The first PLR dropper, called PLR(∞), measures the class loss rates based on all previous packet arrivals, aiming to achieve proportional loss rates in the long run ('long-term PLD model'). The second PLR dropper, called PLR(M), measures the class loss rates based on the last M packet arrivals, aiming to meet the PLD model in a shorter timescale, with a duration determined by a 'history length' M. As will be shown in §4, this difference between the two PLR droppers reflects on their accuracy in meeting the PLD model, on their implementation complexity, and on their adaptability to varying class load distributions.

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

In this dropper, the loss rate estimate $\bar{l}_i(t)$ is measured as the fraction of dropped packets in class i up to time t. $\bar{l}_i(t)$ can be measured using counters for the arrivals and drops in each class. Since the loss rate is estimated from the history of all previous arrivals and drops, we say that PLR(∞) has 'infinite' memory. The complete algorithm for this dropper is shown in Figure 2.

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 §4, however, it may be beneficial under certain traffic conditions to reset the counters over shorter intervals, or to reduce the length of the counters so that the dropper is more adaptive to varying class load distributions.

In terms of implementation complexity, 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 50 cycles, the calculation of the normalized loss rates for 8 classes would require 400 cycles, and with a 500MHz processor the selection of the drop-target class would take about 0.8μs. This overhead would not be a major issue for network interfaces of up to 1Gbps, that transmit a (minimum-size) 40-byte packet in about 0.32μs, since a packet drop would need less than three packet transmission times.

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

In the PLR(M) dropper, the loss rate estimate $\bar{l}_i(t)$ is measured as the fraction of dropped packets from class i in the last M arrivals of the aggregate traffic stream. So, if there are $A_i(M)$ class i packets in the last M arrivals, and $D_i(M)$ of those packets were dropped, the loss rate estimate for class i is $D_i(M)/A_i(M)$. If $A_i(M)$=0, the loss rate in class i is undefined.

The implementation of PLR(M) is slightly more complex than the implementation of PLR(∞). PLR(M) requires a cyclic queue with M entries, called *Loss History Table (LHT)*. The LHT records the class index (*LHT[i].class* $\in \{1, \ldots, N\}$) and the drop status (*LHT[i].loss*: 1 if packet is dropped; 0 otherwise) for 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 3.

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, together with the overhead of maintaining the LHT, are the main additional implementation complexity of PLR(M) compared to PLR(∞). It is noted though, that such temporary tags that are attached to a packet while the packet is stored in the router are not uncommon in modern switch and router designs.

Note that the PLR(M) dropper is different from a PLR(∞)-type of dropper in which the counters are reset to zero after every M arrivals. The PLR(M) dropper attempts to meet the proportional loss rate differentiation model in *every* time window of M arrivals, while the latter dropper attempts to meet the proportional loss rate differentiation in *successive* time windows of M arrivals.

The major question in PLR(M) is how large should M be. A larger value of M leads to a closer approximation of the PLD constraints, but as will be shown in the next section, the dropper becomes less adaptive to a varying class load distribution. One constraint is that M should be large enough so that a dropped packet is always one of the last M arrived packets; otherwise, the drop will not

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

$A_i(M)$: Count of packet arrivals in class i in last M arrivals

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

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

LHT[j].class: Class index of packet in the j'th LHT entry

LHT[j].loss: Drop status of packet in the j'th LHT entry

I: Tail index of LHT

Packet arrival in class i at time t:

// Replace the packet at the LHT tail with the new packet
k=LHT[I].class;
$D_k(M) = D_k(M) -$ *LHT[I].loss*;
$A_k(M) = A_k(M) - 1$;
LHT[I].class=i;
LHT[I].loss =0;
$A_i(M) = A_i(M) + 1$;
Tag g of arrived packet = I;
$I = (I + 1) \bmod M$;

Packet drop from class j at time t:

$j = \arg\min_{i \in B(t) \wedge A_i(M) > 0} \frac{D_i(M)}{A_i(M)\sigma_i}$;
g = tag of dropped packet;
LHT[g].loss=1;
$D_j(M) = D_j(M) + 1$;

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

be recorded in the LHT. This constraint is met in practice, even for small values of M, because the dropped packets are removed from the queue tails.

Another constraint on M is that it should be large enough so that it is feasible to achieve the specified LDPs in a time window of M packet arrivals. Intuitively, if M is not large enough, the PLR(M) dropper does not 'remember' enough previous arrivals and drops in order to adjust the loss rates in that time window based on the proportional constraints of (2). The following result gives a lower bound on the required LHT size M, under stationary conditions for the aggregate loss rate and class load distribution.

Proposition 1: If the aggregate loss rate is $\bar{l}_{ag}$ and the load distribution is $\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$, it is necessary that the LHT size is more than M_{min} entries in order for PLR(M) to meet the LDPs $\{\sigma_1 = 1, \sigma_2, \sigma_3, \ldots, \sigma_N\}$, where

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

and $m = \arg\min_{1 \leq i \leq N} \{\lambda_i \sigma_i\}$.

The proof of this result is given in Appendix I.

4 Evaluation of PLR droppers

This section presents simulation results for the two proposed droppers PLR(∞) and PLR(M). The simulated model is described in detail in Appendix II. We briefly mention here that the backlog controller is Drop-Tail, while the scheduler is HPD with g=0.875 [13]. The number of buffers B and the utilization u are adjusted so that the long-term aggregate loss rate does not exceed about 5%; this is a common operating regime in today's Internet. Other parameters, such as the load distribution $\{\lambda_i\}$, or the LDPs and DDPs, are given for each simulation in the corresponding graph or text. A summary of the simulation results follows.

Focusing first on long-term loss differentiation, it is shown that PLR(∞) is able to meet the specified PLD constraints over a wide range of operating conditions and differentiation parameters. When this is not the case, the simulations show that the Strict Priorities (SP) dropper cannot meet the specified PLD constraints either, meaning that the PLD model is infeasible in those operating conditions. PLR(M) is less accurate in meeting the PLD constraints than PLR(∞). The effect of the LHT size M is shown; the deviations of PLR(M) from the PLD model are less than 10% when M is more than 50-100 thousand packets.

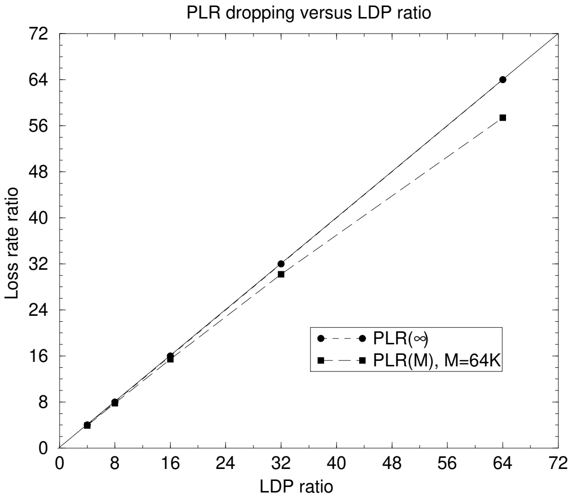
We then focus on loss differentiation in shorter timescales, namely in every K packet arrivals of the aggregate traffic stream. When the class load distribution is stationary and K is in the order of 100,000 packets or more, PLR(∞) approximates quite closely the specified LDPs in almost every interval of K packet arrivals. PLR(M) performs similar with PLR(∞), when K and M are in the same range (100,000 packets or more). When the class load distribution is nonstationary (varying), however, PLR(∞) deviates significantly from the specified LDPs, since it cannot adapt quickly to the changing arrival rates. PLR(M), on the other hand, approximates closely the specified LDPs even with a nonstationary load distribution, as long as the timescales in which the class loads vary are larger than M. Finally, we briefly examine a 'two-dimensional' differentiation framework in which some classes encounter lower delays but higher loss rates than other classes, and illustrate some feasibility problems that appear in that framework.

Feasible proportional loss rate differentiation: Figure 4 shows simulation results for the loss rate ratio between two classes with PLR(∞) and PLR(M), in diverse operating conditions. The loss rate ratio shown is $\bar{l}_1/\bar{l}_2$. The aggregate loss rate is about 1.3% in the first three graphs, and it varies between 0.6%-2.7% in the fourth graph. The parameters that we vary in these simulations are the specified LDP ratio σ_1/σ_2, the DDP ratio δ_1/δ_2, the load distribution (determined by λ_1/λ), and the number of buffers B. The LDP ratio is the desired loss rate between the two classes. The DDP ratio affects the service that each class receives, and thus the likelihood that a class is idle at the drop time instants. A higher DDP ratio makes Class-2 more likely to be empty. Similarly, the load distribution and the number of buffers affect the likelihood that a certain class is empty at the time of a packet drop, and thus they can affect the feasibility of the PLR model.

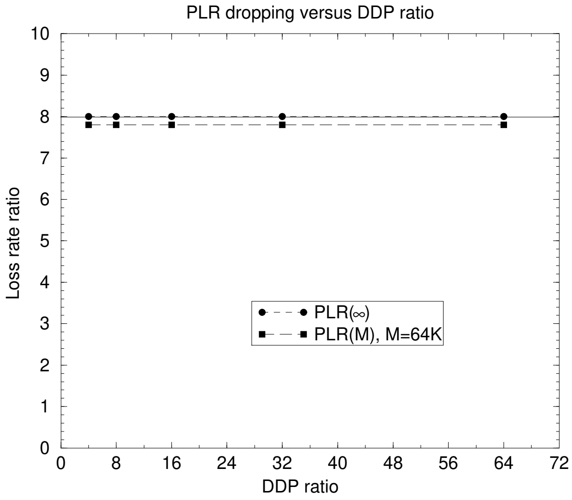
The major observation in these graphs is that PLR(∞) achieves almost exactly the specified LDPs in all the diverse operating conditions shown. In the next paragraph, we show some simulation results in which PLR(∞) deviates from the PLD model, but as it turns out, the model is infeasible in those operating conditions. PLR(M), on the other hand, with an LHT size of M=64K entries, approximates the PLD model with some deviations; these deviations are quantified later in this section.

Infeasible proportional loss rate differentiation: Recall from §2 that the LDP ratio σ_1/σ_2 is feasible if and only if the corresponding loss rate ratio $\bar{l}_1/\bar{l}_2$ with the SP dropper is higher than σ_1/σ_2. Figure 5 shows simulation results for the loss rate ratio $\bar{l}_1/\bar{l}_2$ between two classes with PLR(∞) and SP, in operating conditions that reside at the onset of infeasibility.

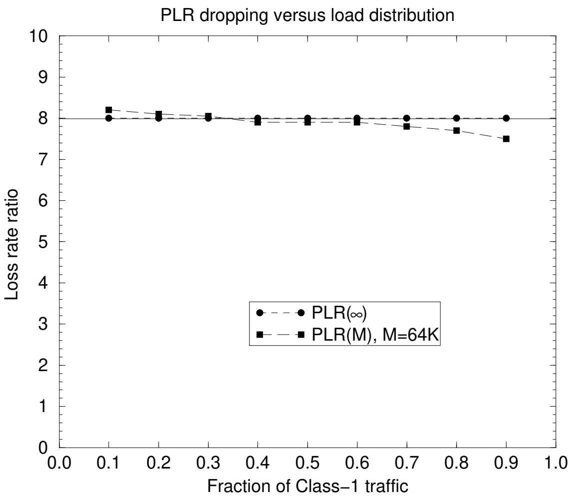
In Figure 5-a, both droppers create a loss rate ratio that is lower than σ_1/σ_2=128. The infeasibil-



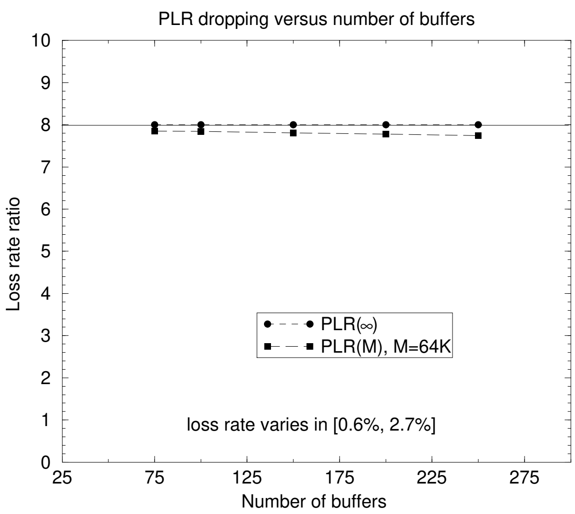
(a) B=150, (λ_1, λ_2)=(70,30), δ_1/δ_2=8



(b) B=150, (λ_1, λ_2)=(70,30), σ_1/σ_2=8

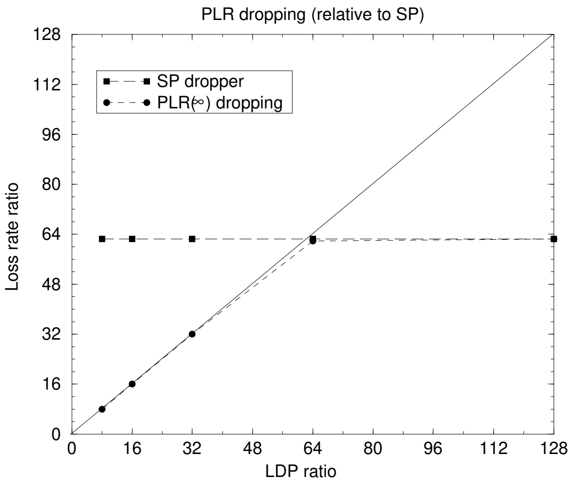


(c) B=150, σ_1/σ_2=8, δ_1/δ_2=8

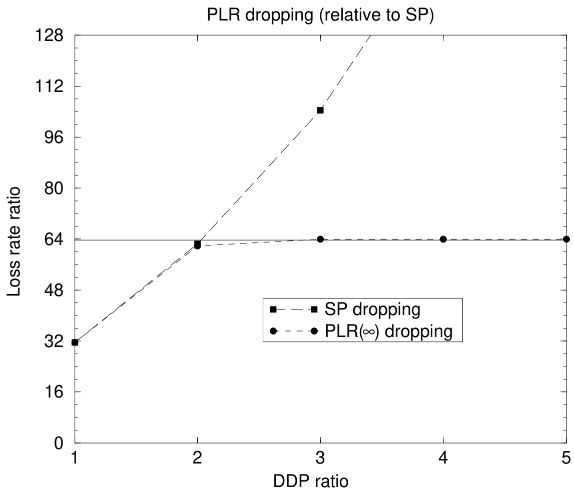


(d) (λ_1, λ_2)=(70,30), σ_1/σ_2=8, δ_1/δ_2=8

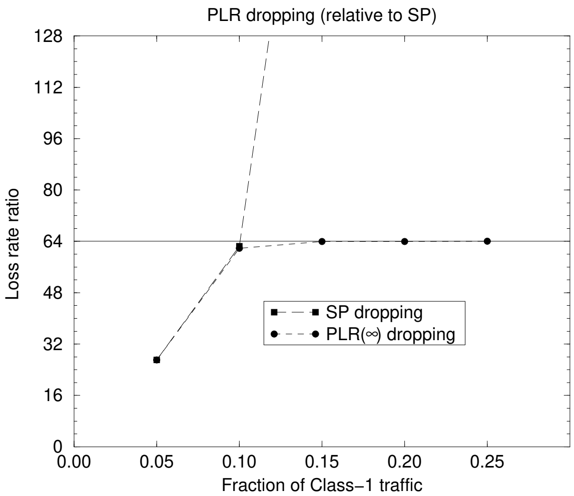
Figure 4: Loss rate differentiation with PLR(∞) and PLR(M) (feasible LDPs).



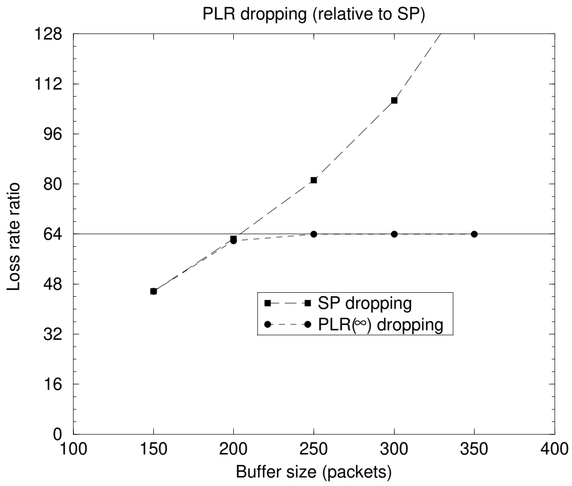
(a) B=200, (λ_1, λ_2)=(10,90), δ_1/δ_2=2



(b) B=200, (λ_1, λ_2)=(10,90), σ_1/σ_2=64

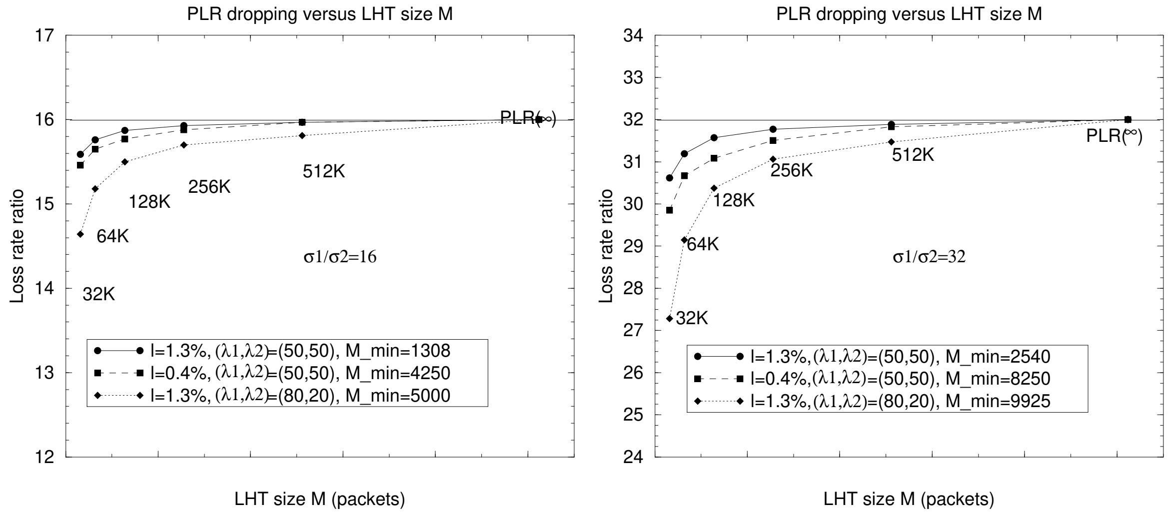


(c) B=200, σ_1/σ_2=64, δ_1/δ_2=2



(d) (λ_1, λ_2)=(10,90), σ_1/σ_2=64, δ_1/δ_2=2

Figure 5: Proportional loss rate differentiation at the onset of infeasibility.



(a) σ_1/σ_2=16, δ_1/δ_2=8

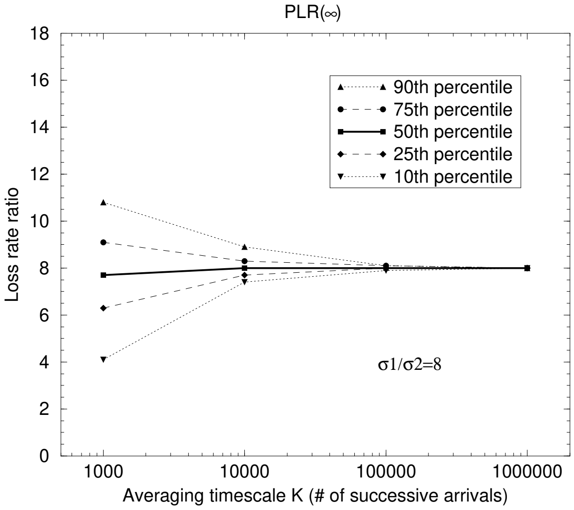
(b) σ_1/σ_2=32, δ_1/δ_2=8

Figure 6: The effect of the LHT size in PLR(M).

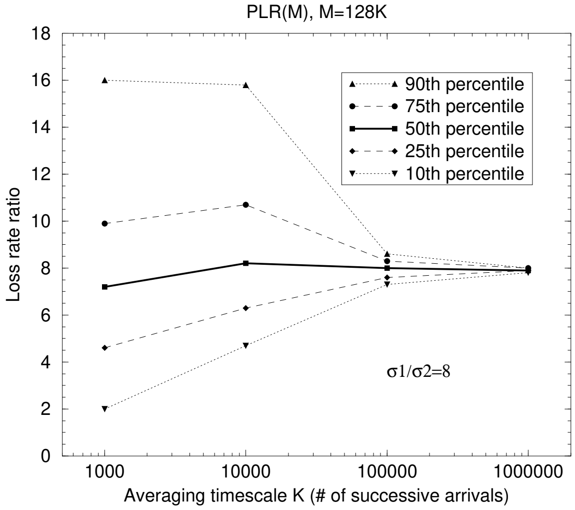
ity cause is that with this high LDP ratio and weak Class-1 load (λ_1/λ=0.10), there are not enough Class-1 packets to drop for the specified loss rate ratio. In Figure 5-b, both droppers create a loss rate ratio that is lower than σ_1/σ_2=64 when there is no delay differentiation between the two classes (δ_1/δ_2=1). The cause of infeasibility is that when the DDP ratio is too low, Class-2 is not discriminated in favor of Class-1, and so Class-1 does not accumulate enough backlog for the specified loss rate ratio. In Figure 5-c, both droppers create a loss rate ratio that is lower than σ_1/σ_2=64 when the load distribution is λ_1/λ=0.05. The infeasibility cause is that Class-1 is too weakly loaded, and so there are not enough Class-1 packets to drop for the specified loss rate ratio. In Figure 5-d, both droppers create a loss rate ratio that is lower than σ_1/σ_2=64 when B=150. The cause of infeasibility is that the low number of buffers leads to a low average backlog in Class-1, and so Class-1 is often idle when it should encounter a packet drop.

In all cases when the PLD model is infeasible, the PLR(∞) and SP droppers lead to the same loss rate ratio. This leads us to the conjecture that *PLR(∞) can meet the specified LDPs, when the PLR model is feasible.*

The effect of the LHT size M in PLR(M): Figure 6 shows the loss rate ratio between two classes for different values of the LHT size M. The resulting loss rate ratio with PLR(∞) is also



(a) PLR(∞)



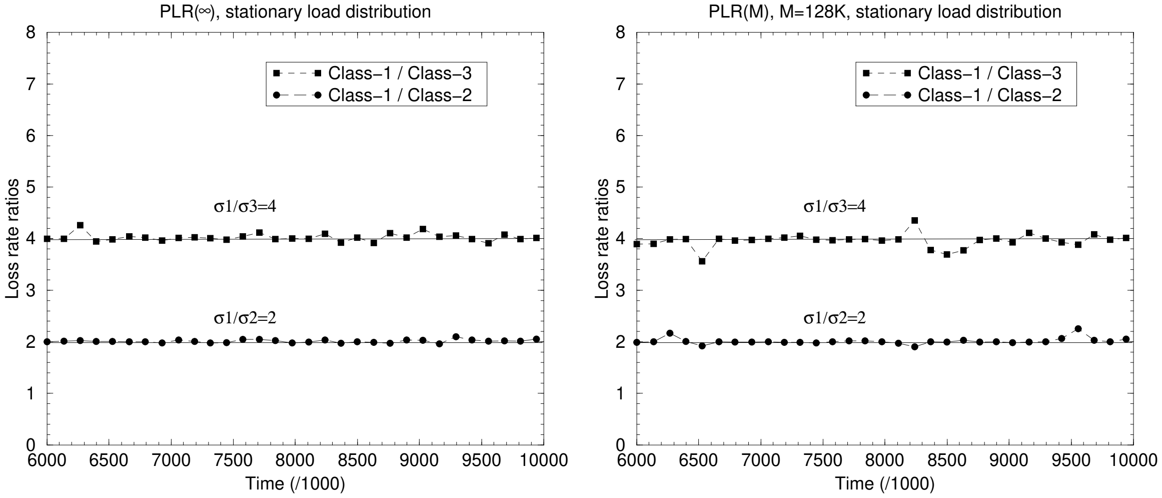
(b) PLR(M), M=128000

Figure 7: Percentiles of the loss rate ratios with PLR(∞) and PLR(M) as a function of the averaging timescale K.

shown, as the asymptotic case $M \to \infty$, in order to compare the two droppers.

The first observation is that as we increase M, the deviations of PLR(M) from the PLD model decrease, because PLR(M) maintains more accurate loss rate estimates. For the operating conditions in these graphs, if the relative error from the PLD constraints is to be less than 10%, the LHT should have at least 64K entries. The second observation is that the required LHT size for meeting the PLD model with a certain accuracy increases as the LDP ratio σ_1/σ_2 increases, as the aggregate loss rate $\bar{l}_{ag}$ decreases, or as the load ratio λ_1/λ_2 increases. These dependencies of the required LHT size M are also predicted from (12). Third, the lower bound on M given in (12) is quite smaller than what is needed in practice for a close approximation of the PLD model. The reason is that the proof of Proposition 1 assumes that the class load distribution and the aggregate loss rate *in any time window* of M packet arrivals follow their long-term average values. In practice, though, the class load distribution and the aggregate loss rate vary significantly around their average values, due to the burstiness in the arrival and packet loss random processes.

Proportional loss rate differentiation in short timescales: Figure 7 examines the behavior of PLR(∞) and PLR(M) in short timescales. Specifically, the graphs in Figure 7 show *five percentiles of the distribution of loss rate ratios between two classes, when the loss rates are measured in time*



(a) PLR(∞),

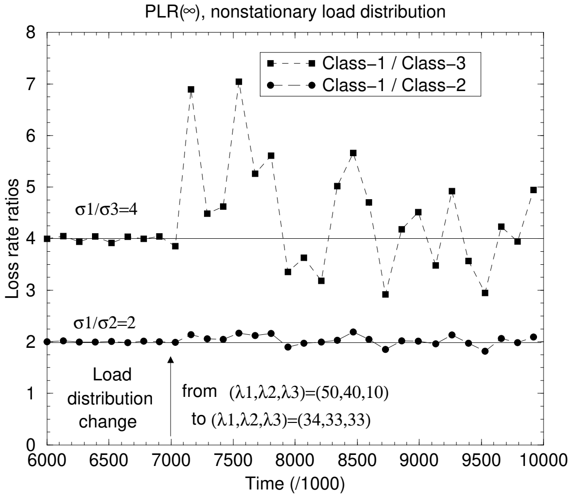
(b) PLR(M), M=128K

Figure 8: Loss rate differentiation with PLR(∞) and PLR(M) in a stationary class load distribution.

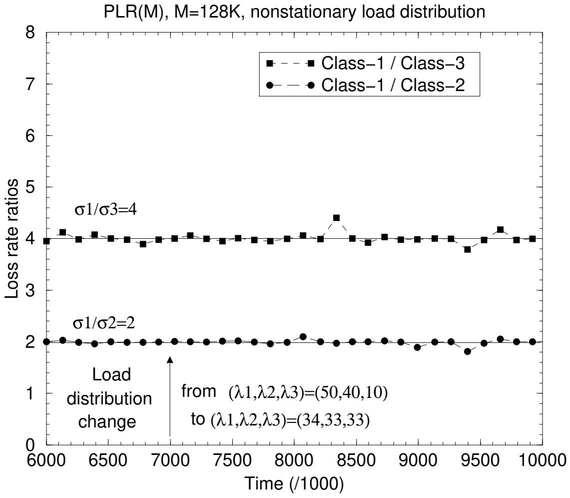
windows of K successive packet arrivals. The number of buffers in these simulations is B=80, and the aggregate loss rate is $\bar{l}$=3.7%. The DDP ratio is δ_1/δ_2=8, the LDP ratio is σ_1/σ_2=8, and the load distribution is (λ_1, λ_2)=(70,30).

As shown, the resulting distributions of loss rate ratios with PLR(∞) are quite narrow and centered around the specified LDP, when K is 100,000 packets or more. In these timescales, the desired loss rate ratio is achieved in almost all intervals of K packet arrivals. In short timescales, say every K=1,000 or 10,000 packets, the resulting distribution of loss rate ratios becomes significantly more spread, but the loss differentiation is still predictable (i.e., the higher class has a lower loss rate). The corresponding distributions in PLR(M) have a wider range than in PLR(∞), especially when $K < M$. For $K > M$, PLR(M) also provides narrow loss rate ratio distributions, centered around the specified LDPs. The PLR(∞) dropper, though, provides more accurate and narrow distributions than PLR(M) in all timescales. As will be shown in the next paragraph, the major advantage of PLR(M) over PLR(∞) is when the load distribution between classes is nonstationary.

Stationary and nonstationary class load distributions: Figure 8 shows a sample path of the loss rate ratios between three classes with PLR(∞) and PLR(M). The loss rates are measured in every K=100,000 packet arrivals of the aggregate traffic stream. In these graphs, the average arrival rate in each class remains constant, i.e., *the class load distribution is stationary.* The major observation



(a) PLR(∞)



(b) PLR(M), M=128K

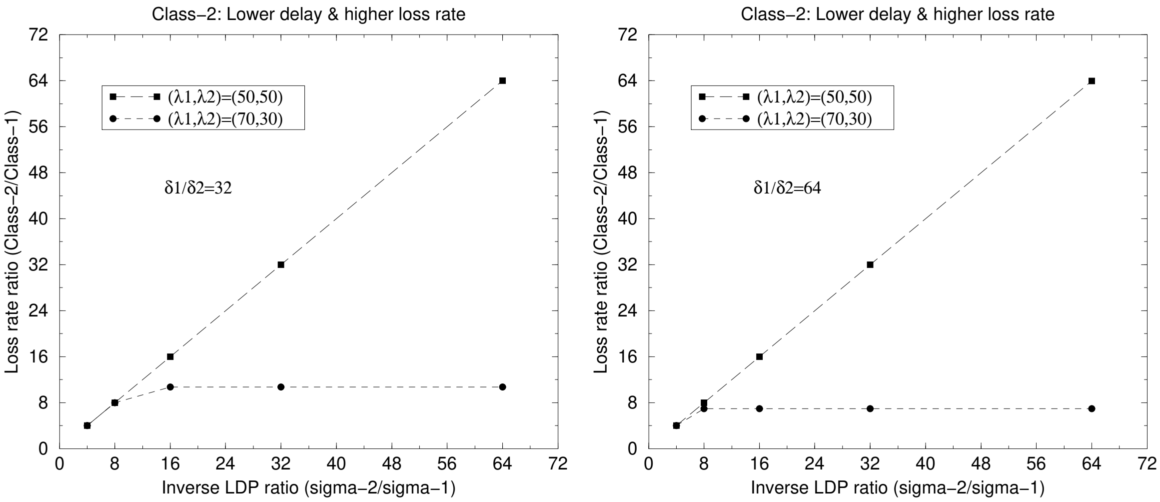
Figure 9: Loss rate differentiation with PLR(∞) and PLR(M) in a nonstationary class load distribution.

is that both droppers approximate quite closely the specified LDPs (σ_i/σ_{i+1}=2) in almost all time intervals of K packets. A more thorough inspection of these, as well as other similar sample paths, shows that PLR(∞) meets the PLD constraints more accurately than PLR(M); this observation also agrees with the results shown in Figure 7.

Figure 9 also shows a sample path of the loss rate ratios between three classes, but this time the class load distribution is nonstationary. Specifically, at the time instant 7000, the average class load distribution is modified from $(\lambda_1, \lambda_2, \lambda_3)$=(50,40,10) to $(\lambda_1, \lambda_2, \lambda_3)$=(34,33,33). Such a change can occur in practice if some users switch between classes.

The PLR(M) dropper is unaffected by the load distribution change, since it only 'remembers' the arrivals from each class in the last M=128,000 packets. So, quickly after the load distribution change, it adapts to the new class arrival rates and delivers the desired loss rate ratios. PLR(M) can deal effectively with continuous load distribution changes, as long as the timescales in which they occur are larger than M.

The PLR(∞) dropper, on the other hand, is affected negatively by the load distribution change, and the loss rate ratio, especially between classes 1 and 3, diverges significantly from the specified LDPs. The reason is that PLR(∞) attempts to provide a certain loss rate ratio over the entire previous history of packet arrivals, which does not reflect the new arrival rates in each class. So,



(a) δ_1/δ_2=32 (b) δ_1/δ_2=64

Figure 10: Example of two-dimensional differentiation.

even though it manages to achieve the specified LDPs over the long run, it fails to provide the desired loss rate ratio in short timescales after the load distribution change. The duration of the erratic behavior depends on the size of the arrival counters and the time until their next counter overflow.

This experiment shows that when the class load distribution is strongly nonstationary, perhaps due to users that switch between classes in order to dynamically select an acceptable class, the preferred dropper should be PLR(M). Even though PLR(M) is not as accurate as PLR(∞) in meeting the PLD constraints under stationary conditions, it is adaptive to varying class load distributions while PLR(∞) is not. If the PLR(M) dropper is considered too hard to implement, the PLR(∞) can be used instead, as long as the arrival and drop counters are reset to zero in relatively short timescales (e.g., every few millions of packets).

Two-dimensional delay and loss rate differentiation: The context of this work is a 'one-dimensional' differentiation framework (such as the IETF Class Selectors) in which *higher classes offer better performance in terms of both queueing delays and packet losses.* The proposed PDD and PLD models, as well as the corresponding schedulers and droppers, however, can also be applied in a 'two-dimensional' differentiation framework in which some classes offer lower delays but higher loss rates than other classes. Even though such a two-dimensional framework is out of the scope of this paper, we show here an example of this approach, illustrating also a feasibility issue that emerges.

In the case of two classes, suppose that Class-2 offers a lower average delay but a higher loss rate than Class-1. The PDD model requires that the average delay ratio is $\bar{d}_2/\bar{d}_1 = \delta_2 < 1$, while the PLD model requires that the loss rate ratio is $\bar{l}_2/\bar{l}_1 = \sigma_2 > 1$. Figure 10 shows the resulting loss rate ratio for two load distributions and two target delay ratios. The scheduler is HPD with g=0.875, and the dropper is PLR(∞). When the load distribution is uniform, (λ_1, λ_2)=(50,50), the specified loss rate ratio is achieved even when we require that Class-2 has a 64 times lower average delay and a 64 times higher loss rate than Class-1. When the load in Class-2 is slightly less though, namely (λ_1, λ_2)=(70,30), the loss rate ratio is $\bar{l}_2/\bar{l}_1 < 11$ when δ_1/δ_2=32, and $\bar{l}_2/\bar{l}_1 < 7$ when δ_1/δ_2=64. The reason that the loss differentiation is not as specified is that the delay differentiation in favor of Class-2 increases the likelihood that Class-2 is idle. Consequently, it becomes hard to introduce a high loss rate in Class-2, relative to the loss rate in Class-1. This effect becomes stronger as the Class-2 load decreases.

In the 'one-dimensional' differentiation framework, on the other hand, the fact that classes with higher delays also have a higher loss rate reduces the space of infeasible operating conditions. The reason is that classes with higher delays are more likely to be backlogged, and so, it is simpler to also provide them with a higher loss rate. In summary, even though a two-dimensional differentiation framework appears to be more flexible than the one-dimensional framework that we consider, it faces some harder feasibility problems that require further investigation.

5 TCP throughput with proportional delay and loss differentiation

Most of the Internet traffic today uses TCP as the underlying transport protocol [35]. Consequently, 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-order investigation of this issue for the case of *bulk-transfer TCP connections*, which spend most of their lifetime in the congestion avoidance phase of the TCP algorithm [23]. During that phase, TCP increases the sending window by a fixed amount of bytes every time it receives an acknowledgement [18]. When a packet loss is detected, the sending window is halved in order to reduce the congestion in the network.

The throughput of a bulk-transfer TCP connection in the congestion avoidance phase can be approximated by

$$R = \frac{0.93\,M}{T\,\sqrt{l}} \tag{13}$$

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 [23]. We assume that the connection encounters the average delay and loss rate of the aggregate traffic in each link along its 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 of the connection's path, 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 TCP connection would get in class i over class j is

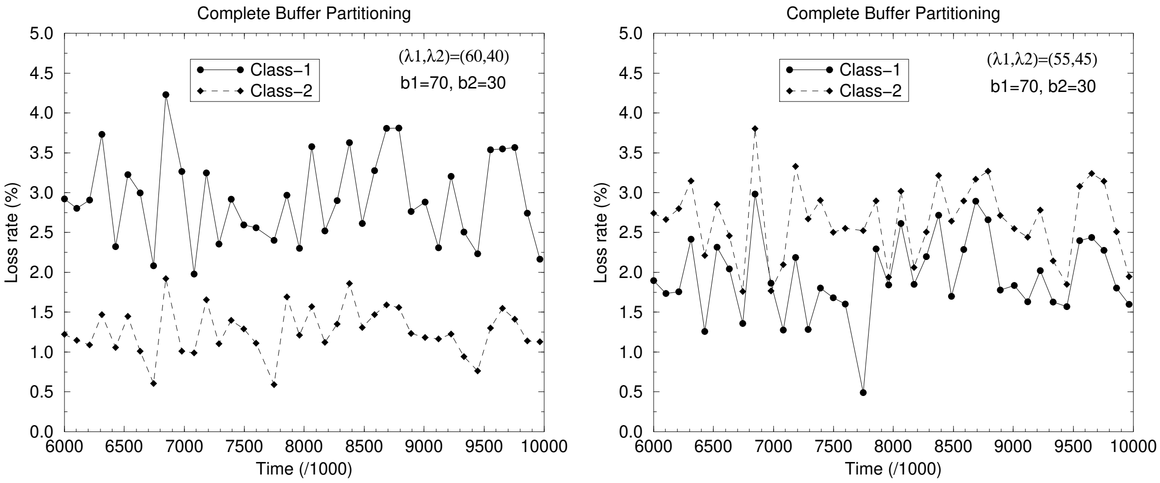
$$\frac{R_i}{R_j} = \sqrt{\sigma}\, \frac{\delta + \tau/\bar{d}_i}{1 + \tau/\bar{d}_i} \tag{14}$$

Note that the loss rate differentiation affects the TCP throughput ratio between classes in a square-root law. The effect of the delay differentiation, 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 linearly. When the connection's propagation delay is large compared to the queueing delays, the impact of the delay differentiation can be minor.

As an example, suppose that δ=σ=2, and that the the propagation delay is equal to the class i average queueing delay ($\tau \approx \bar{d}_i$). The TCP throughput in class i will be approximately 2.1 times higher than in class j, and the contribution of the loss rate differentiation on the throughput ratio will be about the same with the contribution of the delay differentiation. If the propagation delay is three times higher than the class i queueing delay though (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%.

6 Related work on loss differentiation

Prioritized buffer management and dropping algorithms have received significant attention in the literature, especially in the ATM context. Detailed reviews of the early works in this field can be found in [22, 20]. In the following, we first review some of the early buffer management and dropping schemes, illustrating their limitations in the context of relative loss differentiation. Then, we examine



(a) Well provisioned buffer partitions

(b) Poorly provisioned buffer partitions

Figure 11: Loss rates with a two-class CBP dropper.

RIO and multiclass-RED, which have received significant attention recently, especially because of the active buffer management features of RED [15]. Finally, we discuss a few other droppers in the context of proportional loss differentiation.

6.1 Early buffer management schemes

Pushout or Strict Priority (SP): In this simple dropper, the packet removed is always from the lowest backlogged class [22, 20]. We call it *Strict Priority (SP) dropper* to show that it corresponds to the SP scheduler in the delay differentiation context. As shown in §2, the SP dropper is directly related to the feasibility of the PLD model. Although SP provides predictable relative differentiation, it does not provide any means of controlling the loss rates between classes, i.e., it is not a controllable differentiation mechanism. Additionally, SP can cause starvation in the lower classes in the form of excessive packet drops.

Complete Buffer Partitioning (CBP): Another simple buffer management scheme is to statically partition the buffer space between classes, i.e., to allocate b_i packet buffers to each class i (with $\sum_i b_i = B$). A problem with CBP is that the selection of the buffer partitions is quite sensitive to variations in the load distribution. This is shown with two sample paths in Figure 11. The loss rates are measured in every K=100,000 packet arrivals. The HPD scheduler with g=0.875 and

δ_1/δ_2=2 provides proportional delay differentiation between the two classes. The buffer partitions in Figure 11-a are selected so that, when the load distribution is (λ_1, λ_2)=(60,40), the loss rate is about 3% in Class-1 and about 1% in Class-2. A slight change in the load distribution, however, from (λ_1, λ_2)=(60,40) to (λ_1, λ_2)=(55,45), makes the selected buffer partitions totally inappropriate, since they cause a larger loss rate in Class-2 than in Class-1 (Figure 11-b). This is an instance of *unpredictable* loss differentiation, which is common for static resource allocation mechanisms such as CBP.

Partial Buffer Sharing (PBS): In the PBS scheme [20], a packet from class i is accepted in the corresponding queue if the *aggregate backlog* is less than a threshold b_i (with $b_i < b_j$ for $i < j$). So, 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. A problem that we observed experimenting with PBS, is that the buffer thresholds cannot adjust the loss rate differentiation between classes with a fine granularity. In other words, a slight change in the thresholds b_i can lead to a dramatic change in the obtained loss rates.

6.2 Multiclass Random Early Detection (RED) schemes

The mRED [29] and RIO [8] mechanisms are priority extensions of the RED active buffer manager [15]. RIO was originally proposed for two classes, while mRED (or *Weighted RED*) is proposed for N classes. Both RIO and mRED are similar to PBS, since each class is given a buffer threshold b_i, and arriving packets in that class are dropped when the aggregate backlog is more than b_i. The two algorithms differ from PBS in certain buffer management features that they borrow from RED, such as probabilistic dropping, gradual increase of the loss probability as the backlog increases, and backlog estimation through run-time averages. These features are not related to the loss differentiation aspect though, that is the same as in PBS. Consequently, both RIO and mRED have the same drawbacks as PBS in achieving a controllable and predictable relative loss differentiation. These drawbacks, together with some other problems of mRED, are studied in [3].

6.3 Other proportional loss rate droppers

A dropping algorithm that is essentially the same with PLR(∞) has been also discussed in [37, 6], but from a different perspective than ours. Specifically, [6] designed a *self-calibrating pushout* ATM dropper in which 'the number of dropped cells is proportional to the loss weight of different loss priorities', which is similar to the PLD model. [6] proposed a VLSI implementation that does not require floating-point operations for this dropper, *when the arrival rate in each class is policed and* *known.* We cannot use that optimization, because the class loads are not known a priori in the context of relative differentiation.

[37] studied a dropper that is essentially the same as PLR(∞), in a more abstract manner. Specifically, [37] showed that the PLR(∞) dropper, jointly with a FCFS scheduler, is *optimal* because it requires *the minimum capacity for a certain loss rate in each class.* This result is related to the *class provisioning* problem, in which the capacity of a link is adjusted based on the expected class loads, so that each class provides a given *absolute* performance level.

After our original work on the PLR droppers [12], other researchers have proposed variations and improvements. The *Joint Buffer management and Scheduling (JoBS)* algorithm [21] can offer proportional loss rate differentiation between classes. An interesting feature of JoBS is that it unifies the scheduling and dropping decisions in a single step. The loss rate metric that [21] uses is the fraction of lost traffic since the beginning of the current busy period. JoBS solves an optimization problem, taking into account both absolute and relative differentiation constraints, in order to decide which packet to transmit or drop next.

Recently, [2] proposed a variation of PLR(M) called *Average Drop Distance (ADD)* dropper. The main advantage of ADD over PLR(M) is that it avoids the division that is required for calculating the fraction of dropped packets (i.e., the D_i/A_i terms). ADD is based on a run-time estimator for the average drop distance, which is the average number of arrived packets between two successive drops in a class. The estimation of this measure does not require a division, and it can be efficiently implemented in software or hardware. The authors of [2] implemented ADD in the FreeBSD kernel of a Pentium II system, and measured that selecting the drop-target class takes 113 cycles and updating of the loss rate estimates takes 240 cycles for 8 classes.

7 Conclusions

The objective in this paper was to apply the general model of proportional differentiation in the context of loss rates. In the first part of the paper, we proposed and studied the Proportional Loss Differentiation (PLD) model. According to PLD, the loss rate ratios between classes are fixed, based on certain parameters (LDPs) that the network operator specifies. The PLD model provides a *differentiation invariant* that does not depend on the aggregate load or on the class load distribution. The PLD model is similar in several aspects with the PDD model of [13]. One similarity is that the PLD model cannot be feasible in all operating conditions either. The feasibility model of the model can be determined from the loss rates that result with the Strict Priorities dropper.

In the second part of the paper, we considered two Proportional Loss Rate (PLR) droppers for the PLD model. PLR(∞) can achieve the PLD constraints when the model is feasible. In addition, it approximates the PLD model quite well in short timescales when the loss rates are measured in every 100,000 packets or more. PLR(M), on the other hand, is not as accurate as PLR(∞) in meeting the PLD model, but it also approximates the PLD model closely in short timescales when the loss rates are measured in every 100,000 packets or more, given that M is also in the same range. The PLR(∞) dropper is simpler to implement than PLR(M), because the latter requires a cyclical queue and a tag for each packet. PLR(M) should be preferred over PLR(∞), however, when the class load distribution varies with time. In such conditions, PLR(∞) can deviate for large time intervals from the PLD constraints. PLR(M), instead, adapts quickly to the new load conditions.

A number of interesting issues that deserve further investigation follow. First, the feasibility conditions for the PLD model (8) were presented here as a conjecture, based on the corresponding Regnier conditions for the feasibility of the PDD model. Even though our simulation results provide evidence that the conditions of (8) are correct, a mathematical treatment of the feasibility problem in the context of loss rates is still needed. Second, in most of this paper, including the evaluation of the two PLR droppers (§4), we considered only the Drop-Tail backlog controller. It is interesting to examine the PLD model, as well as the PLR droppers, in the presence of RED-like backlog control. Note that active buffer management schemes such as RED drop packets randomly, even when there are available packet buffers. For such backlog controllers the conservation law does not hold, because they are not 'work-conserving' in terms of buffer utilization. Third, our conjecture that PLR(∞) can meet the PLD model when the LDPs are feasible, has been validated with simulations. It is important to examine the validity of this conjecture for arbitrary traffic patterns.

References

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

[2] U. Bodin, A. Jonsson, and O. Schelen, "On Creating Proportional Loss Differentiation: Predictability and Performance," Technical report, Department of Computer Science and Electrical Engineering, Lulea University of Technology, Sweden, July 2000.

[3] U. Bodin, O. Schelen, and S. Pink, "Load-Tolerant Differentiation with Active Queue Management," *ACM Computer Communication Review (CCR)*, July 2000.

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

[5] C. Cetinkaya and E.W.Knightly, "Egress Admission Control," In *Proceedings INFOCOM*, March 2000.

[6] H. 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, no. 5, pp. 867–879, June 1997.

[7] A. Charny and J. L. Boudec, "Delay Bounds in a Network with Aggregate Scheduling," In *Proceedings QOFIS*, October 2000.

[8] D. D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362–373, August 1998.

[9] E. Coffman and I.Mitrani, "A Characterization of Waiting Time Performance Realizable by Single-Server Queues," *Operations Research*, vol. 28, no. 3, pp. 810–821, May 1980.

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

[11] C. Dovrolis and P.Ramanathan, "Dynamic Class Selection: From Relative Differentiation to Absolute QoS," Technical report, University of Wisconsin-Madison, Department of ECE, October 2000.

[12] C. Dovrolis and P.Ramanathan, "Proportional Differentiated Services, Part II: Loss Rate Differentiation and Packet Dropping," In *IEEE/IFIP International Workshop on Quality of Service (IWQoS)*, June 2000.

[13] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," Technical report, University of Delaware, April 2001. Submitted for publication to the IEEE/ACM Transactions on Networking.

[14] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "A Self-Configuring RED Gateway," In *Proceedings IEEE INFOCOM*, April 1999.

[15] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

[16] R. Guerin, J. Roberts, D. Saha, and H. Zhang, "Quality of Service (QoS) in the Internet (special issue)," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2485–2489, December 2000.

[17] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. *Assured Forwarding PHB Group*, June 1999. RFC 2597.

[18] V. Jacobson, "Congestion Avoidance and Control," In *Proceedings ACM SIGCOMM*, pp. 314–329, September 1988.

[19] V. Jacobson, K. Nichols, and K.Poduri. *An Expedited Forwarding PHB*, June 1999. RFC 2598.

[20] H. Kroner, G.Hebuterne, P.Boyer, and A.Gravey, "Priority Management in ATM Switching Nodes," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 3, pp. 418–427, April 1991.

[21] J. Liebeherr and N.Christin, "Buffer Management and Scheduling for Enhanced Differentiated Services," Technical Report CS-2000-24, University of Virginia, August 2000.

[22] A. Y.-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, no. 9, pp. 1524–1536, December 1991.

[23] M. Mathis, J. Semke, and J. Madhavi, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *ACM Computer Communications Review*, vol. 27, no. 3, pp. 67–82, July 1997.

[24] V. Misra, W. B. Gong, and D. Towsley, "Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED," In *Proceedings ACM SIGCOMM*, September 2000.

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

[26] A. M. Odlyzko, "Paris Metro Pricing: The Minimalist Differentiated Services Solution," In *Proceedings IEEE/IFIP International Workshop on Quality of Service*, June 1999.

[27] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," In *Proceedings IEEE INFOCOM*, April 1999.

[28] J. Regnier, "Priority Assignment in Integrated Services Networks," Technical Report LIDS-TH-1565, LIDS-MIT, December 1986.

[29] 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.

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

[31] I. Stoika and H.Zhang, "Providing Guaranteed Services Without Per Flow Management," In *Proceedings ACM SIGCOMM*, September 1999.

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

[33] 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*, September 1999.

[34] M. S. Taqqu, W.Willinger, and R.Sherman, "Proof of a Fundamental Result in Self-Similar Traffic Modeling," *ACM Computer Communications Review*, pp. 5–23, April 1997.

[35] K. Thompson, G. J. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics," *IEEE Network*, pp. 10–23, November 1997.

[36] P. P. White, "RSVP and Integrated Services in the Internet: A Tutorial," *IEEE Communications Magazine*, pp. 100–106, May 1997.

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

[38] N. Yin and M.G.Hluchyj, "Implication of Dropping Packets from the Front of a Queue," *IEEE Transactions on Communications*, vol. 41, no. 6, pp. 846–851, June 1993.

Appendix I: Lower bound of M in PLR(M) dropper

Suppose that the PLD constraints of (2) are met. If A_i and D_i are the number of arrivals and drops, respectively, in class i, the ratio of packet drops between any pair of classes i and j is

$$\frac{D_i}{D_j} = \frac{\sigma_i}{\sigma_j}\frac{A_i}{A_j} \tag{15}$$

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

$$\frac{D_i}{D_j} = \frac{\sigma_i}{\sigma_j}\frac{\lambda_i}{\lambda_j} \tag{16}$$

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

$$m = \arg \min_{1 \leq i \leq N} \{\lambda_i \sigma_i\} \tag{17}$$

Let us now derive the minimum required number of arrivals, given a class load distribution $\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$ and an aggregate average loss rate $\bar{l}_{ag}$, so that the $N-1$ PLD constraints of (2) are met. The minimum number of arrivals causes 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 get

$$D_i = D_m \frac{\sigma_i}{\sigma_m}\frac{A_i}{A_m} = \frac{\sigma_i}{\sigma_m}\frac{\lambda_i}{\lambda_m} \tag{18}$$

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

$$D = \sum_{i=1}^{N} D_i = \sum_{i=1}^{N} \frac{\sigma_i}{\sigma_m}\frac{\lambda_i}{\lambda_m} \tag{19}$$

Since the aggregate loss rate is $\bar{l}_{ag}$, the required number of arrivals is

$$A = \frac{\sum_{i=1}^{N} \frac{\sigma_i}{\sigma_m}\frac{\lambda_i}{\lambda_m}}{\bar{l}_{ag}} \tag{20}$$

This is the minimum number of arrivals that the PLR(M) should remember, in order for the $N-1$ proportional loss rate constraints of (2) to hold. So, this is also the minimum requirement on the LHT size M_{min},

$$M \geq M_{min} = \frac{\sum_{i=1}^{N} \frac{\sigma_i}{\sigma_m}\frac{\lambda_i}{\lambda_m}}{\bar{l}_{ag}} \tag{21}$$

We emphasize that M_{min} is a lower bound on the required LHT size M. The reason is that the previous proof assumes that the class load distribution and the aggregate loss rate are given by

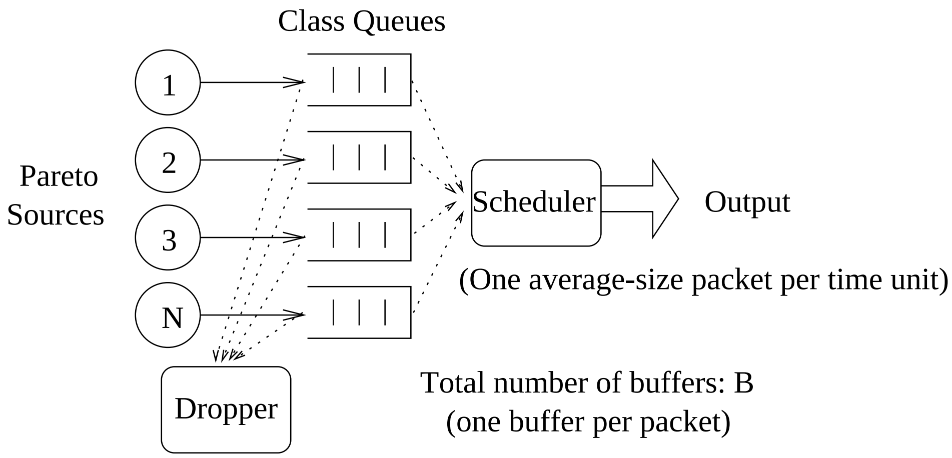


Figure 12: The simulation model for a packet multiplexer.

their average values, namely $\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$ and $\bar{l}_{ag}$, *in any time window of M packet arrivals.* In practice though, the class load distribution and the aggregate loss rate vary significantly around their average values, even when M is many thousands of packets, due to the burstiness of the traffic arrivals and of the packet loss process.

Appendix II: Simulation model

Our simulations are based on the following model (see Figure 12). A scheduler services N *logical* queues, one for each class. The capacity (service rate) of the scheduler is normalized to C=1 average-size packet per time unit. The utilization of the scheduler is $u = \lambda/C$, where λ is the average rate of packet arrivals. The class load distribution is specified in terms of packets. For example, when we say that Class-1 is 70% of the aggregate traffic, we mean that 70% of the packets belong to Class-1.

All queues share the same pool of buffers. In the lossy simulations, the number of buffers is limited to B, and the backlog controller is Drop-Tail. A packet requires one packet buffer, independent of the packet size. When the number of buffers is exceeded, *a packet is dropped from the tail of one of the backlogged queues*; that queue is determined by the packet dropper.

The traffic in each class is generated from a random source. The interarrivals between packets of the same class follow a Pareto distribution with shape parameter $\alpha = 1.5$. The cumulative distribution function of this distribution is $F(x) = 1 - x^{-\alpha}$, the mean is $\alpha/(\alpha-1)$ (for $\alpha > 1$), and the variance is $\alpha/[(\alpha-1)^2(\alpha-2)]$ (for $\alpha > 2$). The distribution has infinite variance when $\alpha < 2$, leading to very bursty interarrivals. The aggregation of many Pareto sources with $\alpha < 2$ has been shown to produce Long Range Dependent (LRD) traffic [34].

We have experimented with several distributions for the packet sizes, including the exponential distribution, multimodal distributions around certain common values (e.g., 40, 550, and 1500 bytes), and also with fixed-size packets. In the case of variable-size packets, all classes have the same packet size distribution. In this paper, we only include simulation results for fixed-size packets; we did not observe any qualitative differences in the results when simulating variable-size packets.

The time unit in all graphs is the time it takes to transmit an average-size packet. The reported delays in all graphs are in these time units. In the case of fixed-size packets, when we say that the queueing delay of a packet is 10 units, we mean that the packet waited in the queue for the transmission of ten other packets.