A Simple FIFO-Based Scheme for Differentiated Loss Guarantees

Yaqing Huang Roch Guérin
Dept. of Electrical and Systems Engineering, University of Pennsylvania
{yaqing@seas, guerin@ee}.upenn.edu

Abstract— **Today's Internet carries an ever broadening range of application traffic with different requirements. This has stressed its original, one-class, best-effort model, and has been one of the main drivers behind the many efforts aimed at introducing QoS. Those efforts have, however, experienced only limited success because their added complexity often conflict with the scalability requirements of the Internet. This has motivated many proposals that try to offer service differentiation while keeping complexity low. This paper shares similar goals and proposes a simple scheme, BoundedRandomDrop (BRD), that supports multiple service classes. BRD focuses on loss differentiation, as although both losses and delay are important performance parameters, the steadily rising speed of Internet links is progressively limiting the impact of delay differentiation. BRD offers strong loss differentiation capabilities with minimal added cost. BRD does not require traffic profiles or admission controls. It guarantees each class losses that, when feasible, are no worse than a specified bound, and enforces differentiation only when required to meet those bounds. In addition, BRD is implemented using a single FIFO queue and a simple random dropping mechanism. The performance of BRD is investigated for a broad range of traffic mixes and shown to consistently achieve its design goals.**

I. Introduction

Today's common communication infrastructure is largely based on IP networks, and the traffic they carry has, therefore, evolved from a relatively homogeneous mix of basic data sources to a diverse set of applications with varying requirements and importance. This widening range of requirements has been behind the many efforts aimed at introducing service differentiation in the Internet. Unfortunately, the success to date of those efforts has been relatively limited. This has been attributed by many to the intrinsic conflict that exists between the added complexity associated with service differentiation, and the scalability requirements of the continuously growing Internet. As a result, there have been a number of proposals aimed at offering some form of service differentiation without incurring too much added cost. The Proportional Differentiated Services model [1] [2], is one example of such efforts.

This paper has a similar target, namely, providing different levels of service in IP networks while introducing minimum additional complexity. We expand later on the various aspects of complexity when implementing service differentiation, but it broadly consists of implementation, deployment and management complexity. Our goal is to develop a solution that while effective at enforcing different levels of service, introduces minimal added complexity along all above three dimensions and can be deployed incrementally in the network. Specifically, we are targeting a solution that, from an implementation complexity perspective, requires little more than a simple FIFO queue. As we shall see, the only addition we consider is in the form of a random drop decision logic through which the different levels of service are enforced. This random drop logic calls for the *a priori* configuration of a single parameter for each offered service class, so that deployment complexity is also kept to a minimum. Finally, the system automatically adapts to the level of traffic in the different service classes, without the need for interactions between users and the network besides the a priori identification of the service class to which a user belongs. In other words, there is no need for active management of resources.

The mechanism we propose, called BoundedRandomDrop (BRD), focuses on loss differentiation. There are two major sources of impairment in IP networks, packet loss and queuing delay. Both are caused by network congestion that arises when the incoming traffic exceeds the network resources, i.e., link speed and buffer space. However, over the last few years the speed of network links, including access links, has been steadily rising at a pace that exceeds that of the growth in buffer size. As a result, the relative contribution of queueing delays to the end-to-end delay has been regularly decreasing. In contrast, losses are unaffected by the higher speed as they remain a function of the network load. This does not mean that delay has become an irrelevant performance measure and that only losses matter, but clearly points to losses as the dominant parameter, and increasingly so as link speeds keep increasing. This is the main motivation behind our focus on losses. Specifically, the paper investigates the possibility of providing per-hop differentiated loss guarantees without upstream policing, knowledge of traffic profiles, or exchange of signalling messages. The choice of per-hop guarantees, as opposed to end-to-end guarantees, is again motivated by our goal of minimum complexity and by the fact that most flows typically encounter only a few bottlenecks on their path. As a result, BRD per hop guarantees on bottleneck links should offer a reasonable approximation of end-to-end guarantees.

There have been a number of previous works that share similar goals as ours. Several of these works originated from the proportional differentiated services model proposed in [1], [2] and [3], and therefore share similar limitations in both performance and implementation complexity. More specifically, they focus primarily on long-term average loss performances and typically require more complicated implementations than

0-7803-8277-3/04/$20.00 ©2004 IEEE.

what we consider in this paper. We discuss these related schemes and illustrate the differences that exist between them and our scheme later in the paper.

The main contributions of this work are in proposing a *simple* FIFO-based scheme, BRD, that is effective at enforcing loss differentiation, and can be deployed relatively easily. BRD will gradually improve the overall loss performances if it is incrementally deployed across the network. The rest of the paper is devoted to describing, characterizing and evaluating BRD, as well as highlighting its differences when compared to previous proposals. Section II articulates more precisely the goals and requirements of BRD. Section III reviews a number of other works that share to different degrees some of our goals, and we highlight key differences. Section IV is devoted to a more formal description of the algorithm on which BRD relies, while Section V evaluates BRD's performance by simulations involving a broad range of scenarios.

II. Problem Description

We assume that the network traffic can be categorized into N traffic classes, with the traffic intensity of each class unknown ahead of time. At a given hop, each class specifies its loss bound LB_i. We refer to these loss bounds as absolute loss requirements. In addition, since some applications are more sensitive to losses or are deemed more important because their users are willing to pay more for better performances, it is natural to also require relative loss requirements among the N classes, i.e., *Class i always has better (or at least no worse) packet loss performance than Class j, for $\forall i < j$, regardless of traffic load variations*. As a result, we can assume without loss of generality that $LB_i \leq LB_j$, for $\forall i < j$, since if there exists $i < j$ such that $LB_i > LB_j$, we can always replace the loss bound of Class i with LB_j. Therefore, we say that Class i has higher priority than Class j, for $\forall j > i$.

Without knowledge of the input traffic, improving the service quality of higher priority classes typically means reducing the resources available to lower priority classes. A simple scheme such as Priority Queue is an extreme example of giving better protection to higher priority classes. However, such an extreme scheme may not be desirable for several reasons. First, it leaves no control over the actual level of quality received by each class; second, it may provide higher priority classes with unnecessarily good service quality at the cost of degrading the service quality of lower priority classes. Therefore, we want to be able to control the actual quality level of each class and avoid unnecessary quality degradation to lower priority classes whenever possible. Specifically, a higher priority class will experience the same loss performance as lower priority classes as long as its absolute loss bound is not violated. In doing so, we avoid unnecessary loss performance degradation to lower priority classes. A higher priority class will receive preferential loss treatment only when it is required to avoid violating it own loss bound. In such instances, the higher priority class will experience a loss rate equal to its stated bound. In addition, when it is not feasible to satisfy the loss bounds of all traffic classes simultaneously, the loss bounds of lower priority classes are relaxed first. More specifically, the loss rate of Class i will exceed its bound only after all packets of classes with lower priority have been discarded. Finally, our definition of loss rate extends to both short-term and long-term loss performance. According to previous studies [4], [5], most traffic flows in the current Internet, web traffic in particular, are of short duration. Enforcing only long-term loss guarantees may, therefore, not be of much benefit to many applications. It may also be desirable to impose a rate limit on each class (except the lowest priority class) so that the amount of traffic receiving preferential treatment is upper-bounded. This can be incorporated through a simple extension of BRD [6].

Finally, we want to achieve the above goals using the simplest possible mechanisms. Providing any form of service guarantees typically involves added complexity when compared to the single FIFO queue implementation required for best-effort service. We review next, the main issues we face when designing a scheme that achieves our goals.

A. Implementation Complexity

Scheduling and buffer management are the two main mechanisms involved in differentiating between packets of different classes. The simplest scheduler transmits packets in FIFO order from a single queue. Introducing multiple queues, e.g., one for each class, adds complexity along multiple dimensions.

First, a scheme that divides the available memory into multiple queues requires a mechanism to enforce memory allocation across the different queues (see [7] for a description of basic memory partition schemes). In general, the greater the desired flexibility in memory allocation, the higher the complexity, and even the simplest multi-queue scheme introduces a significant step in terms of complexity when compared to a single queue system. In addition, the presence of multiple queues also calls for the introduction of a scheduler to arbitrate transmissions across the different queues. The complexity of the scheduler increases with the level of sophistication it uses when deciding which packet to send next (see [8] for a recent survey). In general, the main benefit afforded by schedulers is in terms of the delay guarantees they can provide. Given that packet losses are our primary focus, this is only of limited benefit.

Because multi-queue systems introduce many sources of added complexity without clear benefits given the goals we have set, we focus on a single queue system with a simple FIFO scheduler. In that context, the main remaining control knob for enforcing service differentiation is through differentiated packet dropping, i.e., the decisions of which packets to drop and when to drop them. The simplest schemes make dropping decisions only when a packet arrives and preclude the subsequent removal of packets once they are stored in memory. This allows for a simple queue structure, as there is no need to track the identity and location of individual packets in the queue. Dropping decisions are typically made based on global state variables such as packet counts in each class that can be easily updated at transmission and arrival times.

Given our goal of a simple system, we concentrate on buffer management schemes that only drop packet on arrivals. Our

ability to enforce service differentiation relies then only on the decision process used to determine whether or not to accept an arriving packet. Clearly, packets need to be dropped whenever the buffer is full, but limiting dropping decisions to such cases is unlikely to offer much service differentiation ability, given that the identity of the arriving packet is not under our control. As a result, dropping decisions need to be made for each arriving packet based on additional information such as the class of the packet, the buffer state, and some estimates of the current performance and traffic characteristics of each class. RED [9] and CHoKe [10] are two examples of such mechanisms. We adopt a conceptually similar approach even if we differ in the details of how we reach dropping decisions. In BRD, an arriving packet is randomly dropped with a probability that depends on its traffic class and is computed based on the loss requirements and input traffic intensities of all traffic classes. The added complexity of BRD, when compared to the simple FIFO of a single class system is, therefore, small. It consists of only the initial packet classification, and the dropping function logic that, as we will see, can be implemented relatively easily.

It is worth mentioning that, schemes within the RED family, such as WRED [11] or MRED [12], also rely on random dropping. However, the use of dropping decisions based on (average) queue size, makes it difficult, if not impossible, to enforce accurate loss bounds. This is because the rate of change in queue size depends on both the arrival rates and loss probabilities, so that it is difficult to control loss rates without estimating arrival rates, as BRD does. Similarly, adding ingress policing to limit the traffic entering the network is also not adequate, as even if can help guarantee loss bounds, it will do so by unnecessarily penalizing traffic when the overall level of congestion on the bottleneck link is low.

III. Related work

As mentioned earlier, many of our goals have been shared to different extent by earlier works. We briefly review the most relevant ones and identify what we consider to be key differences between our contributions and theirs.

The original proportional differentiated services model [1], [2], [3] targeted fixed proportions between the QoS levels of the different classes rather than absolute bounds. This often resulted in significant variations in the actual level of performance seen by a given class, in particular, across periods of high and low loads. However, the initial framework provided a starting point for many extensions, several of which incorporated support for absolute QoS bounds. Some of them, [13], [14], [15], [16], rely on admission control or adaptive class selection. Others, such as JoBS, proposed in [17], [18], [19], and the scheme of [20], [21] (denoted as PractQoS in the rest of the paper, because of its original characterization as a "practical solution for proportional QoS"), focus on per-hop performances and control the actual level of service directly. We focus next on JoBS and PractQoS as they are more relevant to our work.

JoBS and PractQoS extend the proportional service model by providing both absolute loss and delay guarantees and proportional differentiations. If we specialize them to only support absolute and proportional loss performance, as is the case in this paper, JoBS and PractQoS are similar, and can be viewed as direct extensions of the original proportional differentiated loss services model [2]. In both schemes, the proportional constraints are relaxed to satisfy the absolute constraints when the two set of constraints cannot be simultaneously satisfied.

By further specializing JoBS and PractQoS to operate with a proportional ratio of 1 across traffic classes, these two schemes can be seen as targeting the same performance goals as the ones described earlier for this paper. It is, therefore, important to identify how they differ from the approach we take to achieve those goals. The main difference has its root in the fact that both schemes were originally designed to meet a more complex set of requirements, namely, enforcing absolute and proportional guarantees for both loss and delay. In contrast, we only target absolute loss bounds with an implicit priority between classes that is based on the relative value of their bounds. This enables us to achieve several advantages within this more confined set of goals, which we briefly review next.

The benefits of the approach we propose can be classified along two dimensions: (1) Functional benefits; and (2) Implementation benefits.

From a functional standpoint, the main advantage of BRD is that it is able to provide "tighter" loss guarantees over both long and short time scales. This is because loss decisions are made based on estimates of the current *rate* of traffic in each class, rather than by relying on past loss *counts*, as is the case with schemes such as JoBS and PractQoS. The main disadvantage of relying on the loss process is that it responds relatively slowly to variations in traffic patterns. As a result, decisions based on the loss process itself often lag behind the changes triggered by traffic fluctuations. In contrast, decisions made based on directly estimating the arrival process are usually more responsive in the presence of traffic fluctuations. This affords better control of loss guarantees, and we illustrate this advantage further in Section V through simulations.

In addition, relying on the loss process to make dropping decisions, typically involves counters to track the number of packets received and lost in each class. Those values are then used to compute the loss rates and make dropping decisions accordingly. This reliance on counters introduces problems that further affect a scheme's ability to tightly control performance over both short and long time scales. One generic problem whenever counters are used, is the need to clear the counters because of wrap-around. Even in the absence of such problems, e.g., through the use of very long counters, counters still need to be reset every so often, as large count values limit the ability to react quickly to traffic changes. Conversely, resetting counters too frequently can limit a scheme's ability to enforce long term guarantees. In general, selecting the right counter resetting strategy is a difficult task that involves multiple trade-offs, even if some adaptive approaches have been proposed, e.g., see [20] for an active counter resetting process.

However, as we illustrate in Section V, the incorporation of loss bounds often conflicts with the active counter resetting process. As a result, both JoBS and PractQoS are likely to exhibit substantial deviations from the desired loss targets over short-term scales. In contrast, because BRD directly measures the traffic rate of each class using a simple exponential filter, it mostly avoids those issues.

The other major advantage of BRD is its implementation simplicity. BRD is implemented using a single FIFO queue and a logic that enforces random dropping decisions only on arriving packets. In contrast, JoBS and PractQoS drop packets only when the buffer overflows. As discussed earlier, this means that they need the ability to remove packets belonging to a specific class and already present in the queue. Implementing this capability with a single queue can be complicated as it calls for the removal of packets that are possibly in the middle of the queue. As a result, both of them use a multi-queue structure, in which dropping a packet from a specific class can be done relatively easily by dropping the last packet of the associated queue. Similarly, when it comes to scheduling, both JoBS and PractQoS rely on complex schedulers. This is in part because of their concern for both delay and loss guarantees, and simpler schedulers, e.g., round-robin, could be used if only loss guarantees were desired, but even those remain more complex than the FIFO scheduler used by BRD.

In summary, the less ambitious goals of BRD translate into several benefits in terms of its ability to offer tight loss guarantees (see Section V for details), and most importantly, in the cost of offering those guarantees. BRD does not rely on signalling or traffic profiles, but is capable of offering meaningful service differentiation using little more than a standard FIFO together with a simple random dropping decision logic.

IV. Algorithm Description

We assume there are N traffic classes with smaller class numbers indicating higher priorities. Our goal is to avoid penalizing lower priority classes, so that their loss rates are increased only when required to enforce the loss bounds of higher priority classes. Our approach, therefore, seeks to minimize the loss rate differences between traffic classes subject to the absolute loss constraints and the relative loss constraints described earlier in Section II. These requirements can be formulated as the following optimization problem:

$$\min \sum_{i=1}^{N-1} p_{i+1} - p_i = \min\ p_N - p_1 \quad (1)$$

$$\text{s.t.} \quad \sum_{i=1}^{N} r_i(1-p_i) \leq C \quad (2)$$

$$p_i \leq p_j \quad \forall i < j \quad (3)$$

$$\sum_{i=1}^{N} p_i > 0 \Rightarrow \sum_{i=1}^{N} r_i(1-p_i) = C \quad (4)$$

$$\forall j \in [1,N],\ p_j > LB_j \Rightarrow p_i = LB_i \text{ and } p_k = 1, \text{ for } \forall i \in [1,j) \text{ and } \forall k \in (j,N] \quad (5)$$

$$0 \leq p_i \leq 1 \quad \forall i \in [1,N] \quad (6)$$

in which p_i, LB_i and r_i are the targeted loss probability, loss bound and input rate of Class i, respectively, and C is the total output bandwidth. Eq. (5) specifies the condition under which the loss rate of a traffic class can exceed its bound. Specifically, *Class i will exceed its bound only after dropping all packets from lower priority classes, and ensuring that higher priority classes experience loss rates that match their bounds*. This guarantees that the loss rate increase in lower priority classes is avoided as long as possible and, when necessary, the absolute constraints are relaxed in order of class priorities.

We can derive a unique closed-form optimal solution for this problem (see [6] for the optimal solution in general form). Eq. (7) shows the optimal solution when $N = 3$. Notice that we can assume without loss of generality that $LB_N = 1$, i. e., we do not need to specify a loss bound on the lowest priority class. As mentioned before, our model guarantees the best possible loss treatment to the lowest priority class.

The BRD algorithm is described as follows:

1) Input traffic rates are estimated using an exponentially weighted moving average with parameter α. For each class i, we use a counter A_i to track the amount of input traffic during each Δt sampling period. At the end of each period, the input rate estimates are updated by $r_i = (1-\alpha)r_i + \alpha A_i/\Delta t$, for $i = 1, \ldots, N$. Then the target loss probabilities, $p_i, i = 1, \ldots, N$, are computed based on the r_i's and all counters are reset.
   Note that different choices of α and Δt embody different trade-offs. A larger weight α and a smaller sampling period Δt result in faster detection times for traffic load variations but less stable estimates. The sensitivity of BRD to choices of α and Δt is a topic we are investigating further. In our simulations, $\alpha = 0.125$ and $\Delta t = 1ms$ were used and performed well across a broad range traffic scenarios, as reported in Section V.
2) Upon the arrival of a packet pkt belonging to Class k, we increase A_k by the size of pkt. Then pkt is randomly dropped with probability p_k, otherwise it enters the buffer. Because dropping packets when the buffer occupancy is relatively low may be overly conservative, probabilistic packet dropping is enabled only when the buffer occupancy exceeds a certain threshold. In our simulations, a threshold of 50 % was found to be a reasonable compromise across a broad range of traffic patterns. Note that because of the probabilistic nature of the early dropping decision, it is still possible, even if rare, to lose packets because of buffer overflows. In all our experiments with a 50 % threshold, we encountered only a few instances of such forced losses.

The BRD algorithm involves few operations. Specifically, upon each arrival, we need one addition and generate one random number; and after each sampling period, we need one addition and three multiplications plus the operations needed to compute the target loss probabilities. Those operations are of a similar overall complexity, especially when considering a small number of classes as will typically be the case.

$$\bar{p} = \begin{cases} (0,0,0), & \text{if } r_1+r_2+r_3 \le C; \\ \left(1-\frac{C}{r_1+r_2+r_3}, 1-\frac{C}{r_1+r_2+r_3}, 1-\frac{C}{r_1+r_2+r_3}\right), & \text{if } 1-\frac{C}{r_1+r_2+r_3} \le LB_1 \text{ and } r_1+r_2+r_3 > C; \\ \left(LB_1, 1-\frac{C-r_1(1-LB_1)}{r_2+r_3}, 1-\frac{C-r_1(1-LB_1)}{r_2+r_3}\right), & \text{if } 1-\frac{C}{r_1+r_2+r_3} > LB_1 \text{ and } 1-\frac{C-r_1(1-LB_1)}{r_2+r_3} \le LB_2; \\ \left(LB_1, LB_2, 1-\frac{C-r_1(1-LB_1)-r_2(1-LB_2)}{r_3}\right), & \text{if } 1-\frac{C-r_1(1-LB_1)}{r_2+r_3} > LB_2 \text{ and } 1-\frac{C-\sum_{i=1}^{2} r_i(1-LB_i)}{r_3} \le 1; \\ \left(LB_1, 1-\frac{C-r_1(1-LB_1)}{r_2}, 1\right), & \text{if } LB_2 < 1-\frac{C-r_1(1-LB_1)}{r_2} \le 1; \\ \left(1-\frac{C}{r_1}, 1, 1\right), & \text{if } LB_1 < 1-\frac{C}{r_1}. \end{cases} \quad (7)$$

V. Simulation results

In this section, we first compare and contrast the performance of BRD and that of JoBS and PractQoS, which when configured properly share similar goals as BRD. Our first scenario is specially designed to illustrate when and why BRD is better capable of achieving the specific set of goals we selected than JoBS and PractQoS that were originally designed for more complex requirements. We then proceed to investigate the performance of BRD across a wide range of traffic mixes, including UDP video traffic, short-term web TCP traffic and long-term FTP TCP traffic. In all of our simulations, we assume that there are no more than 3 traffic classes, which we believe represents a meaningful first step when introducing service differentiation. The target loss probabilities are computed using Eq. (7) with $\alpha = 0.125$ and $\Delta t = 1$ms. Actual loss rates are monitored and computed every 1ms.

A. Performance comparison with JoBS and PractQoS

Our first scenario is specifically designed to highlight the performance differences between BRD and the two most related schemes, namely, JoBS and PractQoS. We used the JoBS module implemented in ns-2.26 [22], and we implemented a module for PractQoS following the specifications put forth in [20]. The main question we wanted to answer in this initial investigation was whether BRD's reliance on traffic estimates rather than loss counts, would indeed allow it to enforce better short-term loss guarantees in the presence of traffic fluctuations. For this purpose, we used the configuration shown in Fig. 1 that consists of three classes each fed by ON-OFF UDP CBR sources. Link $(n1, n0)$ is where service differentiation is enforced using alternatively BRD, JoBS and PractQoS. The loss bounds assigned to each class are set to 10% for Class 1, 20% for Class 2, and none for Class 3.

The input rates of the three classes are shown in Fig. 2. Before time 200s, the total input is 10.47 Mbps, which will allow all three classes to have a 4.5% loss rate without any service differentiation. However, at time 200s, the input of Class 3 increases so that the total input reaches 11.77 Mbps. As a result and as shown in Fig. 3(a), the loss rates of both Class 2 and 3 will be forced to increase to 16.8% so that the loss rate of Class 1 can remain bounded at 10%. At time 400s, the rate increase of Class 2 makes it impossible to satisfy the loss bounds of both Class 1 and 2 simultaneously even after dropping all Class 3 packets[1]. Therefore, Class 2's loss bound will be relaxed beyond 20%. At time 600s, the input of Class 2 drops back down to 6 Mbps, and the target loss rates of the three classes shall return to the values they had during $[200, 400]$s. To summarize, the target loss rates of the three classes are shown in Fig. 3(a).

We proceed next to evaluate the performance of the three schemes for the above scenario. From Fig. 3(b), we see that BRD performs as intended and closely tracks the desired loss target of each class. The small fluctuations in the actual packet loss rates are due to the probabilistic nature of the packet dropping decisions. When JoBS or PractQoS are used, a different behavior is observed as shown in Figs. 3 (c) and (d), which illustrate that the short-term loss rates of the three classes exhibit substantial deviations from their intended targets. As discussed earlier, we believe that those deviations are caused by the reliance of both JoBS and PractQoS on loss counts, as well as interferences between the enforcement of absolute loss bounds and the resetting of the loss counters.

Specifically, JoBS makes packet dropping decisions based on the packet loss counts, and drops a packet from the class that has the *minimum normalized loss rate history*. When the absolute loss bounds are in conflict with the proportional loss requirements, JoBS drops a packet from the class that *has a loss rate history that currently least exceeds its absolute bound*. The impact of this rule is well illustrated in the time window $[200, 400]$s of Fig. 3(c), during which Class 1 violates its loss target; and during $[600, 800]$s, during which Class 2 is over protected. During $[200, 400]$s, Class 1 does not receive the appropriate preferential loss treatments. This is caused by its low loss rate during the initial 200 seconds when the total input is only slightly over the link capacity. Class 1 "catches up" with its target loss rate at time 400s, so differentiation finally kicks in and its loss rate drops down to 10%. The impact of such delayed response of packet loss history is also felt past time 600s when Class 2 drops its rate back down to 6 Mbps. The fact that Class 2 receives essentially lossless performance is because the high loss rate it suffered during $[400, 600]$s allows it to have a loss rate history that is larger than that of Class 1 but smaller than that of Class 3 during the following time period of $[600, 800]$s. Therefore, during $[600, 800]$s, when JoBS makes its dropping decision based on the proportional constraints, it will choose Class 1, since Class 1 has the smallest loss rate history; while if the absolute constraints are at odds with the proportional constraints, JoBS chooses Class 3 because the difference between the Class 3 loss bound (which is essentially 100%) and the Class 3 loss rate history is the

[1]Dropping all Class 3 packets caused by the rate increase in Class 2 can be avoided by introducing a rate limit on Class 2 as mentioned in Section II.

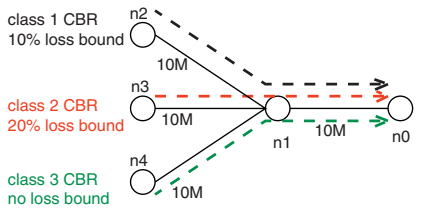

Fig. 1. Scenario 1 configuration.

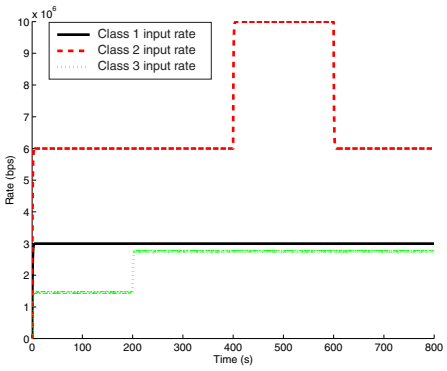

Fig. 2. Scenario 1: Input rate of the three classes.

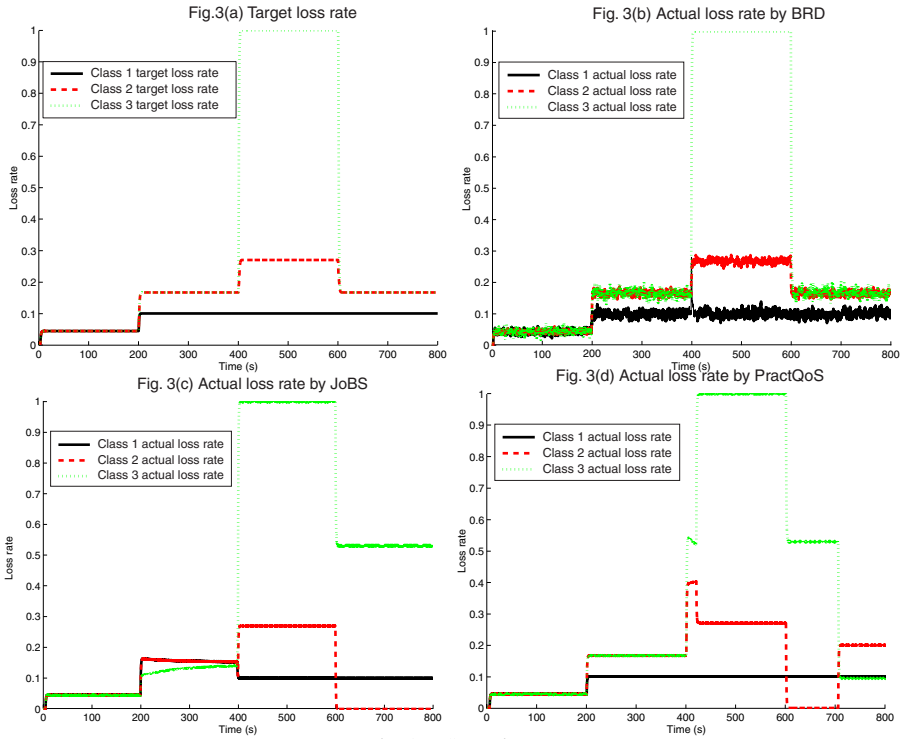

Fig. 3. Scenario 1.

largest among the three classes. As a result, Class 2 receives lossless performance during $[600, 800]$s, which is achieved at the cost of a much higher loss rate in Class 3. This also violates the relative loss requirements by allowing Class 2 to enjoy better loss performance than Class 1 does during that time period.

The problems caused by the delayed reactions of JoBS are to some extent alleviated in PractQoS because of its use of an active counter resetting process. As seen in Fig. 3(d), PractQoS quickly adapts to the increase of Class 3 traffic at time 200s due to its effective counter resetting process during the initial 200 seconds. However, counters can only be reset when the loss ratio among classes are close to their targeted ratio [20], which in our case is $1:1:1$. Therefore, starting from time 200s when the traffic intensity increases, the counter resetting process cannot be performed any more since the absolute loss bound of Class 1 forces the loss ratio to leave the $1:1:1$ proportion. As a result, PractQoS exhibits difficulties in adapting to changes in the input traffic after 200s. The exact nature of those difficulties depends on how PractQoS resolves the conflict between the proportional and absolute loss requirements, and this aspect is not fully specified in [20]. If the same method as JoBS is used, PractQoS will exhibit a similar behavior. In Fig. 3(d), we assume that PractQoS resolves the conflict in the same way as BRD does. Therefore, during $[400, 600]$s, the loss bound of Class 2 is relaxed when dropping all packets from Class 3 still can't satisfy the loss bounds of Class 1 and 2. This again causes the subsequent over-protection of Class 2 during time $[600, 700]$s at the cost of Class 3. Once the loss rate history of Class 2 falls back to its bound at about time 700s, the high loss rate history of Class 3 causes another violation of the relative loss requirements by allowing Class 3 to experience a smaller short-term loss rate than that of Class 2 during the time $[700, 800]$s.

In summary, through this scenario, we have shown BRD's ability to quickly respond to changes in input traffic and to deliver the desired loss guarantees even over relatively short time scales. We also illustrated the limitations exhibited by both JoBS and PractQoS in responding to traffic fluctuations because of their reliance on loss counts as the main parameter to enforce loss differentiation. Even the counter resetting process used by PractQoS failed when absolute requirements interfered with the scheme's target loss proportions. The overall structure and mechanisms used by both JoBS and PractQoS may be justifiable in the context of the more complex goals they were initially designed for, but as we have just shown they present a number of disadvantages for the simpler and narrower design goals set forth for BRD. Our simulations should hopefully establish the benefits of BRD in a setting of extended busy periods with significant traffic fluctuations, the very setting where QoS is truly needed. We proceed next with further investigations of BRD's performance in a number of scenarios that capture different traffic mixes where the service differentiation capabilities of BRD can be of benefit.

B. Additional Performance Investigations

In this section, we further investigate the performance of BRD across a wide range of traffic mixes consisting of both short-lived and long-lived TCP traffic, as well as UDP video traffic. UDP CBR traffic is used as background traffic when necessary. Our investigation is carried out using "realistic" traffic sources. For the UDP video traffic, we use the MPEG-4 video traces of the movie "Jurassic Park I" [23]. Long-lived TCP flows are generated using ns simulated FTP connections, while short-lived TCP flows are generated using both ns

simulated Web connections and short-lived exponential on-off TCP connections that emulate average Internet flows. We first structure the investigation to cover multiple scenarios that attempt to provide a reasonably comprehensive coverage of the different possible assignments of traffic types to service classes. For each, we assess BRD's ability to enforce the desired loss behavior. Next, we focus on what we consider to be realistic configurations for which we evaluate the benefits offered by BRD in terms of not only losses, but also performance measure such as TCP throughput, HTTP response time, and FTP file transfer times. We compare and contrast those with what is achievable when service differentiation is offered using a simple priority queue scheme.

The three types of traffic mentioned above typically have different loss requirements. For example, TCP traffic is sensitive to losses because TCP reduces its rate in the presence of cumulative losses. Short-lived TCP traffic generated by interactive applications that require low delay is even more sensitive because losses are more likely to cause TCP time-outs that can significantly increase its response time. UDP streaming video traffic is also sensitive to packet losses as they degrade the intrinsic quality of the video signal. However, since UDP traffic does not reduces its rate when detecting packet losses and thanks to various loss concealment techniques, this traffic type may not require loss bounds as strict as, say, short-lived TCP traffic. Finally, the level of loss guarantee required by a traffic class can also depend on the importance of that traffic to the service subscriber, or the importance of the subscriber to the service provider, e.g., a higher paying subscriber. Therefore, in our simulations we vary the relative importance of different traffic types to reflect this possibility.

In scenario 2, we study the effectiveness of BRD separately for each of the above three types of traffic. We consider only 2 traffic classes with UDP CBR traffic as the lower priority background traffic. The high priority class is either UDP video or one of the two types of TCP traffic. In our simulations, the video traffic requires 10% loss bound; while both types of TCP traffic require 1% loss bounds.

First, we consider the high priority class to be long-lived FTP connections with a configuration illustrated in Fig. 4. Differentiated loss guarantees is enforced on the bottleneck link $(n1, n0)$. We have two sets of TCP sources, *src*1 and *src*2, each consisting of 50 FTP connections. The FTP connections from *src1* are active throughout the simulations; while the connections from *src2* are active only during $[500, 700]$s. If only one of the two sets of sources is active, BRD should be able to provide 1% loss bound to Class 1 traffic regardless of the intensity of Class 2 traffic. However, if both sets of sources are active, we expect the packet loss rate of Class 1 to exceed 1% and all Class 2 packets to be dropped.

As we can see from the TCP and CBR input rates in Fig. 5(a) and TCP throughput in Fig. 5(b), when TCP is protected by a 1% loss bound, TCP is able to achieve its maximum throughput and remains mostly insensitive to the increases in the CBR traffic.

When both sets of sources are active during $[500, 700]$s, TCP should and did experience more than 1% losses, as shown in Fig. 5(c). The combined input from the *src1* and *src2* makes it impossible to satisfy the loss bound of Class 1 even after all Class 2 packets are dropped. Furthermore, we notice that the actual TCP loss rate is about 3%. This is smaller than what would have been experienced had the TCP sources used their full total input link bandwidth of 12 Mbps, because of TCP's rate adaptation to losses. We also observe from Fig. 5(c) that when *src2* is not active after 700s, the TCP loss rate returns to 1% as expected. BRD can also successfully enforce the different loss behaviors it was designed for when the long-lived FTP traffic is replaced with short-lived TCP traffic (see [6] for those results).

We investigate next the performance of BRD when the long-lived TCP traffic is replaced with UDP Video traffic. We use the same setup as in Fig. 4, but now each of the two sets of sources consists of 35 video sessions. As shown in Fig. 6(a)-(d), the loss rate of the video sources can also be effectively controlled by BRD, as the video loss rate only exceeds its bound when the total video input exceeds 11 Mbps. Overall, BRD shows consistent performances whether long-lived or short-lived TCP or UDP video traffic is used.

In scenario 3, we investigate the performance of BRD in what might be considered a more realistic setting. As shown in Fig. 7, our setup involves all three previous traffic types that are now mapped onto 3 classes. The short-lived TCP traffic consists of 50 exponentially on-off connections with an average off period of 1s and an average on period of 15s and an average rate of 50 Kbps during the on periods to simulate average web transactions. Web transactions are most important to service subscribers, and are most sensitive to losses as was mentioned before. A 1% loss bound is, therefore, required for this Class 1 traffic. The UDP video traffic, consisting of two groups of users from $n7a$ and $n7b$ requesting MPEG-4 streaming videos from two sites, namely $n2a$ and $n2b$, is also important. A 10% loss bound is required for this Class 2 traffic. The long-lived TCP traffic consists of 50 FTP connections representing normal file transfers or average Internet traffic. It is of least importance in this setting. Thus, it is our Class 3 traffic and no loss bound is required.

As shown in Fig. 7, we have configured the setup so that link $(n4, n5)$ is the bandwidth bottleneck and its delay dominates the total RTT time in our simulation, which is approximately 150 ms representing the typical RTT time of a cross continental path. We implement BRD on $n4$ to provide differentiated loss guarantees on the large volume of traffic originated from node $n1$, $n2a$, $n2b$ and $n3$ and headed to node $n6$, $n7a$, $n7b$ and $n8$, respectively. Throughout the simulation, all connections originating from node $n1$, $n2a$ and $n3$ are always active, while connections from node $n2b$ are only active in the $[200, 600]$s interval, which results in a Class 2 traffic input increase from about 3.5 Mbps to 7 Mbps.

In terms of performance, we are particularly interested in the impact of BRD on the throughput of the two TCP classes. We are also interested in the difference in user perceived performances such as the average HTTP response

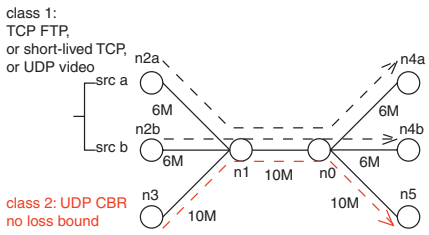

Fig. 4. Scenario 2 configuration.

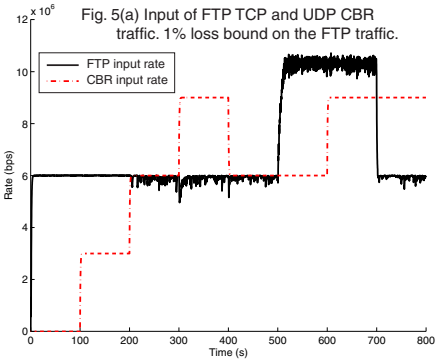

Fig. 5. (a) Scenario 2*a*.

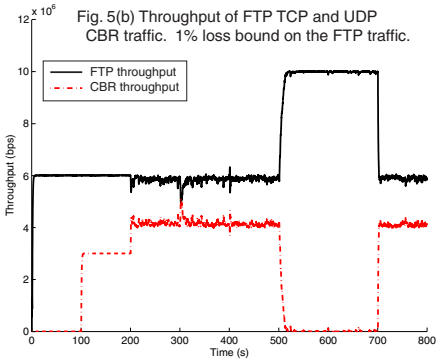

Fig. 5. (b) Scenario 2*a*.

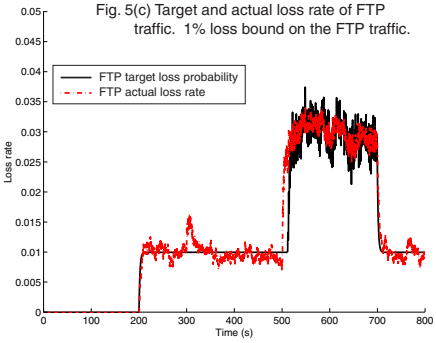

Fig. 5. (c) Scenario 2*a*.

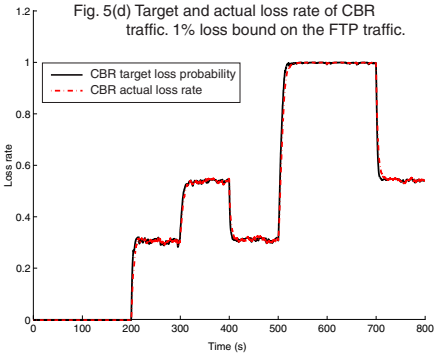

Fig. 5. (d) Scenario 2*a*.

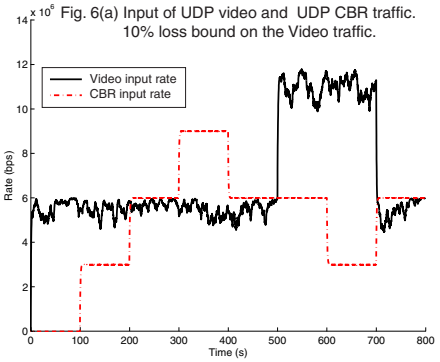

Fig. 6. (a) Scenario 2*b*.

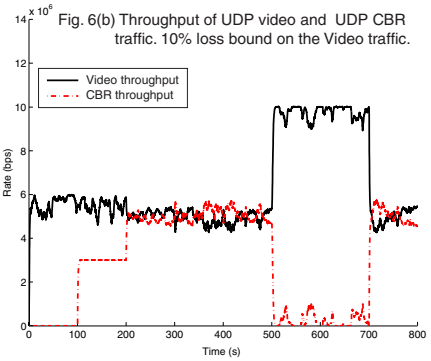

Fig. 6. (b) Scenario 2*b*.

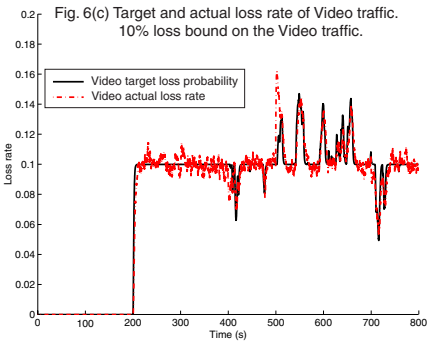

Fig. 6. (c) Scenario 2*b*.

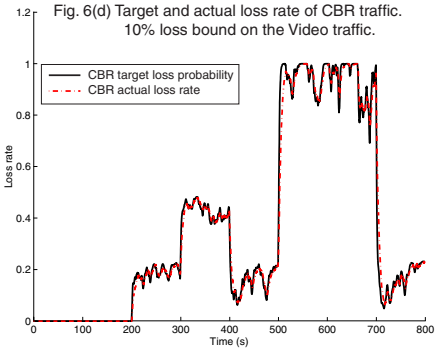

Fig. 6. (d) Scenario 2*b*.

times and FTP file transfer times when these connections are assigned to different traffic classes. The HTTP response time is measured by adding to both Class 1 and 3 two HTTP1.0 [22] connections. Both of the HTTP connections generate HTTP requests with exponential distributed inter-arrival time between consecutive requests. The mean inter-arrival time of the Class 1 requests is 10 seconds and the mean inter-arrival time for Class 3 requests is 50 seconds, so that the number of concurrent connections within each class is small yet the total number of connections finished during the simulation is large enough to obtain a meaningful average. For more realistic results, the requested webpages used in the simulations have the same characteristics as typical web pages sampled from popular web sites such as `CNN.com` and `Amazon.com`.

The performance of BRD is then compared to that of a simple 3-class priority scheme that is commonly used to provide service differentiation. In the priority scheme, short-lived TCP traffic is granted the highest priority and long-lived TCP traffic is given the lowest priority. Video traffic is again assigned to the middle priority. The priority scheme is implemented with three equal-sized FIFO queues, each dedicated to one traffic class, and served according to a strict priority schedule. Arriving packets are dropped when the associated FIFO queue is full.

The input rate of the three classes using BRD is shown in Fig. 8. The increase in the Class 2 input during the time $[200, 600]$s only affects Class 3 traffic, as we can see in both Fig. 8 and 9. Although Class 1 is also protected from the lower priority classes when a priority scheme is used, the throughput of Class 3 is much lower in that case. As we can see in Fig. 10, the throughput of Class 3 is actually close to zero during time $[200, 600]$s. This illustrates a key deficiency of a priority scheme when compared to BRD in that it "over-penalizes" lower classes by giving unnecessarily good performance to higher priority classes.

Next, we further quantify this difference by looking at HTTP response times and FTP file transfer times. The average HTTP response time is an important performance measure

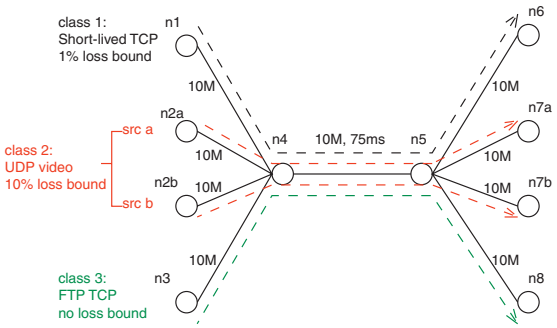

Fig. 7. Scenario 3 configuration.

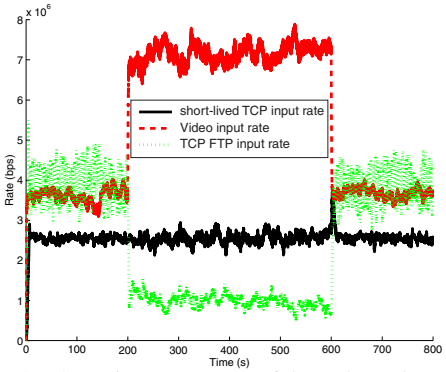

Fig. 8. Scenario 3: Input rates of the 3 classes by BRD.

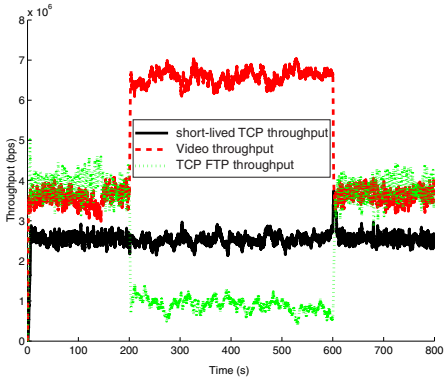

Fig. 9. Scenario 3: Throughput by BRD.

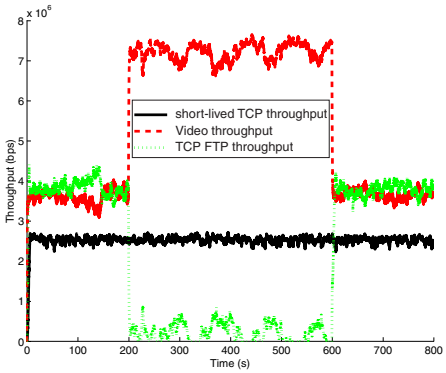

Fig. 10. Scenario 3: Throughput by Priority Queue.

that affects the performance of most web applications and it depends on both the characteristics of the requested pages and the service class to which the traffic is assigned. Similarly, while FTP traffic may be viewed as being less important and less sensitive to increases in total transfer times, it nevertheless calls for "reasonable" completion times in order to remain useful. It is, therefore, of interest to ensure that its performance is not degraded below an acceptable level.

We investigate first HTTP response times when the HTTP traffic is assigned to Class 1. In this case, the client at node $n6$ requests from the server at node $n1$ a page that has the same page size and number of images as a typical page from `www.amazon.com` *checkout*. This corresponds to relatively small pages for which the rapid completion of the underlying transaction is important. In particular, service and content providers such as `Amazon.com` may be willing to pay for a premium service when the traffic is generated by a client requesting a checkout web page, i.e., the completion of an order. Next, we investigate the response time of more standard web connections, namely, browsing of common web pages, with the HTTP traffic now assigned to Class 3. In this case, the client at node $n8$ requests from $n3$ pages that have the same page characteristics as the typical *front pages* from `www.amazon.com` and `www.cnn.com`. For the purpose of better assessing the impact of class selection on the response time of web connections, we also used a third setup where the same CNN page was transmitted using Class 1.

Overall, Table I illustrates that as expected the smaller the page size and the better the service class, i.e., the lower loss rate in BRD's case, the shorter the HTTP response time. When the `Amazon.com` *checkout* page is carried as Class 1 traffic, the HTTP response time is slightly shorter with a priority scheme than with BRD. However, BRD's response time (about 3s versus 1.5s for the priority scheme) remains well within the range of acceptable response times for interactive transactions. Furthermore, BRD clearly shows its advantage over the priority scheme, when it comes to the performance seen by normal web traffic, such as browsing the front page of `www.amazon.com` or `www.cnn.com`. Such traffic is clearly of lesser importance, but it nevertheless needs to be delivered with reasonable performance, if only to ensure that customers visit the web site in the first place. As we can see from Table I, when this traffic is sent as Class 3 traffic, the response time is about 43s for BRD versus 90s with a priority scheme. Similarly, the transfer of the (very large) Amazon front page is 111s with BRD and 238s with a priority scheme. These represent meaningful differences even if the progressive loading of a page will often allow the users to start browsing and acquiring useful information before the page is fully loaded. Overall, this illustrates the benefit afforded by BRD that can offer strong protection to sensitive traffic, while avoiding overly penalizing other traffic classes.

We also investigated the average FTP transfer time with different file sizes assuming that this traffic had been assigned to Class 3, as this is another important measure of the cost of giving better service to other classes. As shown in Table II, BRD again is able to mitigate the performance degradation experienced by Class 3 traffic while offering Class 1 (and 2) a level of service comparable to that of a priority queue scheme.

TABLE I

AVERAGE HTTP RESPONSE TIME

	amazon.com checkout	amazon.com front page	cnn.com front page	cnn.com front page
Main page size (bytes)	13132	109992	67590	67590
Avg. image size (bytes) × No. of images	8402×4	8456×62	2233×70	2233×70
Class priority	1	3	3	1
Response time by BRD (s)	3.3353	111.0078	43.3506	11.6489
Response time by the priority scheme (s)	1.4775	238.7476	90.1861	5.1734

TABLE II

AVERAGE CLASS 3 FTP FILE TRANSFER TIME

File size (bytes)	1000000	500000	100000
Avg. file transfer time by BRD (s)	185.8306	87.6049	22.4773
Avg. file transfer time by the priority scheme (s)	401.9695	179.744	45.3731

VI. CONCLUSION

In this paper, we investigated the feasibility of providing strong loss differentiation at a low additional cost. We proposed a new scheme called BRD that provides a relative service quality order across traffic classes and guarantees each class losses that, when feasible, are no worse than a specified bound and enforces differentiation only when required to meet those bounds. Those capabilities are achieved using a single FIFO queue and a simple random dropping mechanism.

We believe that because of its narrower focus, BRD offers several advantages over previous comparable schemes, namely JoBS and PractQoS. From a performance perspective, BRD is capable of providing both long and short term performance guarantees by relying on directly estimating the arrival process. In contrast, as illustrated in Section V, there are scenarios where JoBS and PractQoS exhibit significant deviations from the desired short-term loss guarantees. We believe that this is in part caused by their choice of making dropping decisions based on the loss process itself. From a complexity perspective, the multi-queue structure of both JoBS and PractQos, while justifiable in the context of their broader goals, introduces additional complexity when compared to the single FIFO queue on which BRD relies. Through simulations, we showed that BRD delivers consistent loss guarantees across a broad range of traffic mixes. In particular, we saw that when compared to a simple priority scheme, BRD delivers a similar level of protection to high priority traffic without unnecessarily penalizing lower priority traffic.

We hope that this paper demonstrates that a scheme such as BRD can provide meaningful service differentiation at a low cost, and can be deployed incrementally over the Internet.

REFERENCES

[1] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional differentiated services: Delay differentiation and packet scheduling," *IEEE/ACM Trans. Netw.*, vol. 10, no. 1, pp. 12–26, February 2002.

[2] C. Dovrolis and P. Ramanathan, "Proportional differentiated services, part II: Loss rate differentiation and packet dropping," in *Proc. of IWQoS 2000*, Pittsburgh, PA, June 2000.

[3] ——, "A case for relative differentiated services and the proportional differentiation model," *IEEE Netw. Mag.*, vol. 13, no. 5, pp. 26–34, October 1999.

[4] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," *IEEE Netw. Mag.*, vol. 11, no. 6, November 1997.

[5] K. Claffy, G. J. Miller, and K. Thompson, "The nature of the beast : recent traffic measurements from an Internet backbone," in *Proc. of INET'98*, Geneva, Switzerland, July 1998.

[6] Y. Huang and R. Guérin, "A simple FIFO-based scheme for differentiated loss guarantees," Univ. of Pennsylvania, Tech. Rep., Feb. 2004.

[7] F. Kamoun and L. Kleinrock, "Analysis of shared finite storage in a computer network node environment under general traffic conditions," *IEEE Trans. Commun.*, vol. COM-28, no. 7, pp. 992–1003, July 1980.

[8] R. Guérin and V. Peris, "Quality-of-service in packet networks: Basic mechanisms and directions," *Computer Netw.*, vol. 31, no. 3, pp. 169–179, February 1999.

[9] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, August 1993.

[10] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe: A stateless active queue management scheme for approximating fair bandwidth allocation," in *Proc. of IEEE Infocom 2000*, Tel-Aviv, Israel, April 2000, pp. 942–951.

[11] "Weighted random early detection," http://www.cisco.com/.

[12] S. Sahu, P. Nain, C. Diot, V. Firoiu, and D. F. Towsley, "On achievable service differentiation with token bucket marking for TCP," in *Measurement and Modeling of Computer Systems*, 2000, pp. 23–33. [Online]. Available: citeseer.ist.psu.edu/sahu00achievable.html

[13] W. Wu, Y. Ren, and X. Shan, "Forwarding the balance between absolute and relative: a new differentiated services model for adaptive traffic," in *Proc. of the IEEE Workshop High Perf. Switching and Routing 2001*, Dallas, TX, May 2001, pp. 250–254.

[14] T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Gharghavan, "Delay differentiation and adaptation in core stateless networks," in *Proc. of IEEE Infocom 2000*, Tel-Aviv, Israel, March 2000.

[15] J. Shin, J.-G. Kim, J. Kim, and C.-C. J. Kuo, "Dynamic QoS mapping control for streaming video in relative service differentiation networks," *European Trans. Telecommun.*, vol. 12, no. 3, pp. 217–230, May-June 2001.

[16] C. Dovrolis and P. Ramanathan, "Dynamic class selection: from relative differentiation to absolute QoS," in *Proc. of the 2001 IEEE Intl. Conf. Netw. Prot.*, Riverside, CA, November 2001.

[17] J. Liebeherr and N. Christin, "Rate allocation and buffer management for differentiated services," *Comput. Netw.*, vol. 40, no. 1, pp. 89–110, May 2002.

[18] N. Christin, J. Liebeherr, and T. Abdelzaher, "A quantitative assured forwarding service," in *Proc. of IEEE Infocom 2002*, vol. 2, New York, NY, June 2002, pp. 864–873.

[19] J. Liebeherr and N. Christin, "A QoS architecture for quantitative service differentiation," *IEEE Commun. Mag.*, vol. 41, no. 6, pp. 38–45, June 2002.

[20] Y. Chen, M. Hamdi, D. H. K. Tsang, and C. Qiao, "Proportional QoS provision: A uniform and practical solution," in *Proc. of IEEE ICC 2002*, New York, NY, April 2002, pp. 2363–2367.

[21] Y. Chen, C. Qiao, M. Hamdi, and D. H. K. Tsang, "Proportional differentiation: A scalable QoS approach," *IEEE Commun. Mag.*, vol. 41, no. 6, pp. 52–59, June 2003.

[22] VINT Project, "The network simulator NS-2," UC Berkeley, LBL, USC/ISI, Xerox Parc, http://www.isi.edu/nsnam/vint.

[23] "MPEG-4 and H. 263 video traces for network performance evaluation," http://www-tkn.ee.tu-berlin,de/research/trace/trace.html.