July 2006

A Simple FIFO-Based Scheme for Differentiated Loss Guarantees

Yaqing Huang
University of Pennsylvania, yaqingh@gmail.com

Roch A. Guérin
University of Pennsylvania, guerin@acm.org

A Simple FIFO-Based Scheme for Differentiated Loss Guarantees

Abstract
Today's Internet carries traffic from a broad range of applications with different requirements. This has stressed its original, one-class, best-effort model, and has been a major driver of the many efforts aimed at introducing QoS. These efforts have, however, been met with only limited success, in part because the complexity they add is often at odds with the scalability requirements of the Internet. This has motivated many investigations for solutions that offer a better trade-off between service differentiation and complexity. This paper shares similar goals and proposes a simple scheme, Bounded Random Drop (BRD), that supports multiple service classes and is implemented using a single FIFO queue and a basic random dropping mechanism. BRD focuses on loss differentiation, as although losses and delay are both important, the steady rise of Internet link speeds is progressively limiting the impact of delay differentiation. It offers strong loss differentiation capabilities, and does not require traffic profiles or admission controls. BRD guarantees each class losses that, when feasible, are no worse than a specified bound, while enforcing differentiation only when required to meet those bounds. The performance of BRD is investigated for a broad range of traffic mixes and shown to consistently achieve its design goals.

Keywords
QoS, Queue Management, Diff-Serv

Comments
Postprint version. Forthcoming in *Computer Networks*,
Publisher URL: http://www.elsevier.com/wps/find/journaldescription.cws_home/505606/description#description

A Simple FIFO-Based Scheme for Differentiated Loss Guarantees

Yaqing Huang and Roch Guérin

Department of Electrical and Systems Engineering, University of Pennsylvania
200 South 33rd Street, Philadelphia, PA 19104, USA
Tel: 1(215)898-9351, Fax: 1(215)573-2068

Abstract

Today's Internet carries traffic from a broad range of applications with different requirements. This has stressed its original, one-class, best-effort model, and has been a major driver of the many efforts aimed at introducing QoS. These efforts have, however, been met with only limited success, in part because the complexity they add is often at odds with the scalability requirements of the Internet. This has motivated many investigations for solutions that offer a better trade-off between service differentiation and complexity. This paper shares similar goals and proposes a simple scheme, Bounded Random Drop (BRD), that supports multiple service classes and is implemented using a single FIFO queue and a basic random dropping mechanism. BRD focuses on loss differentiation, as although losses and delay are both important, the steady rise of Internet link speeds is progressively limiting the impact of delay differentiation. It offers strong loss differentiation capabilities, and does not require traffic profiles or admission controls. BRD guarantees each class losses that, when feasible, are no worse than a specified bound, while enforcing differentiation only when required to meet those bounds. The performance of BRD is investigated for a broad range of traffic mixes and shown to consistently achieve its design goals.

Key words: QoS, Queue Management, Diff-Serv

1 Introduction

The traffic carried today over IP networks has 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. However, the success to date of these efforts has been limited. This has been attributed by many to the intrinsic conflict that exists between the added complexity

⋆ This work was supported by NSF grants ANI-9902943 and ITR-0085930, and presented in part at IWQoS 2004.

Email address: {yaqing@seas, guerin@ee}.upenn.edu (Yaqing Huang and Roch Guérin).

associated with service differentiation, and the scalability requirements of a continuously growing network. As a result, there have been a number of proposals aimed at offering some form of service differentiation while keeping complexity low. For example and of particular relevance to this paper, the Proportional Differentiated Services model [1,2] is one such effort.

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 Bounded Random Drop (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 bandwidth 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 [3]. 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 irrelevant and that only losses matter, but points to losses as the increasingly dominant metric. 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 [1,2,4], 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 what we consider in this paper. We will discuss these related schemes, particularly the schemes proposed in [5–7] and [8,9], and illustrate the differences that exist between them and our scheme in Section 3 and Section 5.1.

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 organized as follows: Section 2 articulates more precisely the goals and requirements of BRD. Section 3 reviews a number of other works that share to different degrees some of our goals and discusses major differences. Section 4 is devoted to a more formal description of the algorithm on which BRD relies, while Section 5 evaluates BRD's performance through simulations.

2 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, we assume there are *absolute loss guarantees* for each class, namely, each class specifies a bound on its loss rate LB_i, $i \in [1, N]$. Our definition of loss extends to both short-term and long-term loss performance. A significant portion of traffic flows in the current Internet, web traffic in particular, is of short duration [10,11]. Enforcing only long-term loss guarantees may, therefore, not be of much benefit to many applications. In addition, since some applications are more sensitive to losses or are simply deemed more important because their users are willing to pay more for better performances, it is natural to also require *relative loss guarantees* among the N classes. We adopt the definition of relative loss guarantees used in [1,2], i.e., *Class i always has better (or at least no worse) loss performance than Class j, $\forall j > i$, regardless of load variations*. As a result, we assume without loss of generality that $LB_i \leq LB_j, \forall i < j$, and we say that Class i has higher priority than Class j.

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 better service than needed at the cost of unnecessarily poor service to lower priority classes. Our goal is to control the quality level of each class, while avoiding unnecessary quality degradation of lower priority classes. Specifically, classes should experience the same loss performances as long as their absolute loss bounds are not violated. A higher priority class receives preferential loss treatment only

when required to avoid violating its own loss bound. In such instances, it experiences 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. Specifically, the loss rate of Class i exceeds its bound only after *all* packets of lower priority classes have been discarded.

In summary, BRD seeks to achieve the following performance goals:

- Satisfy absolute loss guarantees, i.e., the loss rate of Class i is bounded by LB_i, $\forall i \in [1, N]$. Those bounds should be enforced over both short-term and long-term time scales.
- Satisfy relative loss guarantees, i.e., Class i should have better (or at least no worse) loss performance than Class $j, \forall j > i$, regardless of load variations.
- Avoid unnecessary loss performance degradation to the lower priority classes.

The above performance goals ensure that lower priority classes receive the best possible loss performances subject to absolute and relative loss guarantees. In some cases, it may also be desirable to limit the impact of high priority classes over lower priority ones. This is because a large traffic burst from a high priority class coupled with its low absolute loss bound can literally starve lower priority classes. We discuss a simple extension of the original BRD scheme, in which an input rate limit is imposed on each traffic class (except the lowest priority class) to mitigate the impact of such scenarios. For each traffic class, the amount of traffic entitled to preferential treatment, as defined for this class, is upper-bounded by its input rate limit, and any traffic in excess of this limit receives the same treatment as traffic from the next lower priority class.

Finally, we want to achieve the above goals with as little added complexity as possible, i.e., little more than the basic FIFO queue that best-effort service calls for. We review this aspect next.

2.1 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 between them (see [12] 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 is a significant delta in complexity when compared to a single queue system. In addition, the presence of multiple queues also mandates the use of a scheduler to arbitrate transmissions from the different queues. The complexity of the scheduler increases with its level of sophistication for deciding which packet

to send next (see [13] for a recent survey). In general, the main benefit afforded by schedulers are the rate and delay guarantees they can provide to each queue. Given our focus on packet losses, the latter is only of limited benefit.

For all of the above reasons, we concentrate on trying to meet our goals within the framework of single (FIFO) queue systems. 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 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. As a result, we investigate first the use of such systems.

When packet dropping decisions are made only upon arrivals, enforcing service differentiation typically calls for a "proactive" packet dropping mechanism. Clearly, packets need to be dropped whenever the buffer is full, but this does not offer much in terms of service differentiation capabilities. This calls for dropping decisions that are made for each arriving packet based on additional information such as the class of the packet, the buffer state, and estimates of the current performance and traffic characteristics of each class. RED [14] and CHoKe [15] 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 guarantees and input traffic intensities of all classes. The added complexity of BRD, when compared to the simple single class FIFO queue is, therefore, small. It consists of the initial packet classification, which is required by any scheme that needs to identify packets as belonging to different classes, and the dropping logic that, as we shall see, can be implemented relatively easily.

It is worth mentioning that schemes within the RED family, such as WRED [16] or MRED [17], 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 the loss probabilities. This makes controlling loss rates without estimating arrival rates difficult. Alternatively, loss bounds can also be guaranteed by controlling the amount of traffic entering the network through ingress policing. The main disadvantages are the added cost of the policing mechanisms, and the limited ability to offer multiple levels of service.

3 Related Works

As mentioned earlier, many of our goals have been shared to different extents by a number of earlier works. We briefly review the most relevant, and identify what we

consider to be key differences between their contributions and ours.

The original proportional differentiated services model [1,2,4] 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 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, [18–21], rely on admission control or adaptive class selection. Others, such as JoBS [5–7], and the scheme of [8,9] (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 performance and directly control the actual level of service. We focus next on JoBS and PractQoS as they are the most relevant to our work.

JoBS and PractQoS extend the proportional service model by providing both absolute loss and delay guarantees and proportional differentiation. 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 sets of constraints cannot be simultaneously satisfied.

By further specializing JoBS and PractQoS to operate with a proportional ratio of 1 across classes, they can be seen as targeting the same performance goals as the ones described earlier for BRD. It is, therefore, important to identify how they differ from the approach we take. The main difference has its root in the fact that both schemes were originally designed to meet a more complex set of goals, 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 both 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. [1] In

[1] It is worth mentioning that the ADD scheme proposed in [22], which is an extension of the proportional loss differentiation framework [2], also attempts to overcome the disadvantages of using loss history. ADD improves loss rate estimate by using the notion of

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 5.

Relying on the loss process to make packet drop decisions, also involves counters to track the number of packets received and lost in each class. Those counts are used to compute 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 that of wrap-around. Even if wrap-arounds can be avoided, e.g., through the use of very long counters, counters still need to be reset every so often, as a large count "history" limits the ability to react quickly to traffic changes. Conversely, resetting counters too frequently can limit the ability to enforce long term guarantees. In general, selecting the right counter resetting strategy is a difficult task that involves multiple compromises, even if some reasonably successful adaptive approaches exits, e.g., the active counter resetting process of [8]. Furthermore, as we illustrate in Section 5, the incorporation of loss bounds often conflicts with an active counter resetting process. As a result, both JoBS and PractQoS can exhibit significant 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 these issues.

The other major difference between BRD and both JoBS and PractQoS is in terms of implementation. BRD uses a single FIFO queue and a logic that enforces random drops 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. However, this is because of their concern for both delay and loss. Simpler schedulers, e.g., round-robin, could be used if only loss guarantees were targeted, but even these remain more complex than BRD's FIFO scheduler.

In summary, the less ambitious goals of BRD translate into several benefits in terms of its ability to offer tight loss guarantees, and in the cost of offering these 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 packet drop decision logic.

"average drop distance", namely, the average number of successfully transferred packets between two lost packets. However, ADD targets only relative loss performance goals, and it is not clear if and how absolute loss guarantees can be directly incorporated into ADD.

4 Algorithm Description

As described in Section 2, BRD's performance goal is to minimize loss rate differences between classes subject to absolute and relative loss guarantees. These requirements can be formulated as an optimization problem, as expressed in equations (1) to (6). LB_i and r_i are the loss bound and input rate of Class i, respectively, C is the total output bandwidth, and the $p_i, i \in [1, N]$, are target loss probabilities to be optimized.

Equation (1) reflects BRD's goal of avoiding unnecessary performance degradation for lower priority classes. Equations (2) and (3) are work-conserving constraints, and Eq. (4) is because p_i are probabilities. Eq. (5) reflects the relative loss guarantees of BRD. Eq. (6) is a *key constraint* that not only reflects the absolute loss guarantees of BRD, but also specifies the condition under which the loss rate of a traffic class can exceed its absolute bound. Specifically, *Class i exceeds its bound only after all packets from lower priority classes have been dropped, and while ensuring that higher priority classes experience loss rates that exactly match their bounds*. This avoids loss rate increases in low priority classes unless required to meet the bounds of higher priority classes, and if absolute bounds cannot all be met, they are relaxed in the order of class priorities.

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

$$\text{such that} \quad \sum_{i=1}^{N} r_i(1 - p_i) \le C \tag{2}$$

$$\text{if} \quad \sum_{i=1}^{N} p_i > 0 \quad \text{then} \quad \sum_{i=1}^{N} r_i(1 - p_i) = C \tag{3}$$

$$0 \le p_i \le 1 \qquad \forall i \in [1, N] \tag{4}$$

$$p_i \le p_j \qquad \forall i < j \tag{5}$$

$$\forall j \in [1, N] \quad \text{if} \quad p_j > LB_j \quad \text{then} \quad p_i = LB_i, \quad \forall i \in [1, j) \quad \text{and} \quad p_k = 1, \quad \forall k \in (j, N] \tag{6}$$

This optimization problem has an explicit, unique optimal solution, as stated in the next theorem.

Theorem 1 *For the optimization problem defined in equations (1) to (6), there exists a unique optimal solution $\overline{p^*} = (p_1^*, \ldots, p_N^*)$ defined by Eq. (7).*

A detailed proof of Theorem 1 is provided in Appendix A, and here we briefly provide some intuition for Eq. (7). When the input rates are low enough, it is easy to see that the target function Eq. (1) is minimized when there is no loss differentiation. As the input rates increase, loss differentiation is required and the absolute loss bounds are enforced, starting from the highest priority to the lowest priority class. As the input rates further increase, the loss rates of the traffic classes, starting

from the lowest priority class and finally to the highest priority class, will exceed their absolute bounds, according to the constraint set forth in Eq. (6). The optimal solution shown in Eq. (7) covers all possible input scenarios and is organized in the above order. Furthermore, from Eq. (7), we see that we can assume without loss of generality that $LB_N = 1$, i.e., no loss bound is needed for the lowest priority class, and it is guaranteed the best possible loss treatment subject to the absolute and relative loss constraints of higher priority classes.

$$\overline{p^*} = \begin{cases} (0,0,\ldots,0), & \text{if } \sum_{i=1}^{N} r_i \le C; \\ \left(1-\frac{C}{\sum_{i=1}^{N} r_i},\ldots,1-\frac{C}{\sum_{i=1}^{N} r_i}\right), & \text{if } 0 < 1-\frac{C}{\sum_{i=1}^{N} r_i} \le LB_1; \\ \left(LB_1,\ldots,LB_k,1-\frac{C-\sum_{i=1}^{k} r_i(1-LB_i)}{\sum_{i=k+1}^{N} r_i},\ldots,1-\frac{C-\sum_{i=1}^{k} r_i(1-LB_i)}{\sum_{i=k+1}^{N} r_i}\right), & \\ \quad \text{if } LB_k < 1-\frac{C-\sum_{i=1}^{k} r_i(1-LB_i)}{\sum_{i=k+1}^{N} r_i} \le LB_{k+1},\ k=1,\ldots,N-2; & \\ \left(LB_1,\ldots,LB_{N-1},1-\frac{C-\sum_{i=1}^{N-1} r_i(1-LB_i)}{r_N}\right), & \\ \quad \text{if } LB_{N-1} < 1-\frac{C-\sum_{i=1}^{N-1} r_i(1-LB_i)}{r_N} \le 1; & \\ \left(LB_1,\ldots,LB_{k-1},1-\frac{C-\sum_{i=1}^{k-1} r_i(1-LB_i)}{r_k},1,\ldots,1\right), & \\ \quad \text{if } LB_k < 1-\frac{C-\sum_{i=1}^{k-1} r_i(1-LB_i)}{r_k} \le 1,\ k=2,\ldots,N-1; & \\ \left(1-\frac{C}{r_1},1,\ldots,1\right), & \text{if } LB_1 < 1-\frac{C}{r_1}. \end{cases} \tag{7}$$

4.1 BRD algorithm

Based on the previous expressions for $\overline{p^*}$, the BRD algorithm proceeds 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[2] 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 using $r_i = (1-\alpha)r_i + \alpha A_i/\Delta t, \forall i \in [1, N]$. The target loss probabilities, $p_i, \forall i \in [1, N]$, are then computed based on the r_i's and all counters are reset.

Note that the choice of parameters α and Δt affects the performance of BRD. The choice of Δt embodies a trade-off between accuracy and complexity. A smaller Δt allows for more frequent updates of rate estimates and drop probabilities, but also means more computations. The choice of α embodies a different performance trade-off. A large α results in faster detection of traffic variations, but less stable estimates. In Section 4.2, we discuss the impact

[2] Because counters are reset every sampling period, the issues mentioned earlier with respect to JoBS and PractQoS do not apply.

of different choice of parameters on BRD's ability to protect higher priority classes against rate variations of lower priority classes.

In our experiments, $\alpha = 0.125$ and $\Delta t = 1ms$ were used and performed well in our simulations, as reported in Section 5.

(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 a small number of operations upon each packet arrival and during each update procedure. Those operations are simply additions and multiplications, and are therefore, of a small overall complexity, especially when the number of classes is small, as will typically be the case.

4.2 Impact of parameters

Because BRD drops packets according to drop probabilities computed based on rate estimates, we need to investigate how different choices of rate estimation parameters affect its performance. In particular, we are concerned with BRD's ability to protect higher priority classes from traffic variations of lower priority classes, namely, we want to minimize the difference between the actual loss rate and the target loss rates the higher priority classes should have experienced. To this end, we consider a simplified two class *bufferless* systems, in which only the low priority class exhibits rate variations, and investigate the impact of BRD's parameters, in particular the parameter α, on its ability to rapidly detect and react to these changes.

We distinguish between three categories of rate variations of the low priority classes as a function of their time scale. The first consists of variations on the time scale of a single sampling-period, i.e., variations that occur within a sampling period without affecting the value of the sampled rate at the end of the sampling period. Such short-term variations cannot be detected, no matter what value is chosen for the parameter α. The only option for detecting them, and therefore limit their potential impact, is to decrease the duration Δt of the sampling period. The second time scale we consider extends over multiple sampling periods, i.e., the rate remains constant when measured over several sampling periods, but sampling rates in consecutive sampling periods can differ. Because those variations extend over multiple periods, BRD's ability to detect them depends on the selected value of α. Finally, we consider a long-term time scale with rate changes that are permanent, i.e., exhibit a step function behavior from one value to another. Here again, the selected value of

α determines how fast BRD can respond to such changes.

For traffic variations that are confined within a sampling-period, it can be shown (see [23, Appendix B] for a detailed derivation) that in the worst case, such variations can cause deviations between the actual and the target loss rate of the high priority class of up to 25%. As mentioned before, detecting such variations, and therefore reduce their impact, can only be done by decreasing the duration of the sampling period. The shorter the sampling period, the harder it is for the low priority class to precisely time its traffic so as to achieve such worst case patterns. In general, the combination of the difficulty of creating the required finely timed traffic patterns and their limited impact for reasonably small sampling intervals, implies that BRD's sensitivity to traffic variations at this time scale is quite limited.

The situation is somewhat different when considering variations that extend over multiple sampling periods, while preserving a constant rate when averaged over these periods. Specifically, it is shown in [23, Appendix C] that it is possible to construct traffic patterns that span three sampling periods such that the rate varies across each successive period but is constant over that entire duration, and for which the worst case optimum configuration of BRD is $\alpha = 0$. In other words, $\alpha = 0$ minimizes the loss rate deviation for the high priority class under the worst case input variations of the low priority class. Furthermore, as shown in [23, Appendix C], the worst case deviation in loss rate grows at least linearly when α is small (i.e., when $\alpha \leq 2LB$), and is at least $0.25 + LB/3$ when $\alpha > 2LB$, where LB denotes the loss bound of the high priority traffic.

The previous result indicates that when input variations are short-term, BRD should ignore such variations and use a smaller α to achieve more stable rate estimates. However, picking the "optimal" value of $\alpha = 0$ is likely to be a very poor choice in the presence of more persistent variations. For example, consider a scenario where the rate of the low priority traffic jumps by a certain amount, and remains at that level after that. Quickly detecting such a rate change, and therefore adjusting dropping probabilities accordingly, calls for using a large α. For example (see [23]), when $\alpha = 0.25$ it takes 8 sampling periods for the rate estimate to recover more than 90% of the initial difference between the estimated rate and the true (new) rate, while this increases to 17 sampling periods when a value of $\alpha = 0.125$ is used.

The presence of buffers [3] will clearly help mitigate the impact of estimation inaccuracies that result from a particular choice of parameters in BRD. However, it should be clear from the previous discussion that no optimal solution exists that offers good performance across all possible scenarios. Any particular choice will embody a trade-off in BRD's ability to accurately handle different types of traffic variations. Based on our experience with a broad range of settings, we believe that $\alpha = 0.125$ represents a good trade-off in terms of sensitivity to both short-term and

[3] Recall that the results were obtained under the assumption of a bufferless system.

long-term variations, and we validate this through experiments in Section 5.

4.3 Input rate limit extension

From the perspective of traffic classes that are low in priority, BRD provides them with the best possible loss performance *subject to meeting the loss guarantees of higher priority classes*. This implies that higher priority classes can still starve lower priority classes when they have low loss bounds and large traffic volume. This may not always be desirable, and for that purpose we explore an extension of the original BRD scheme that can mitigate this effect. In the extended BRD, the amount of traffic that can received the preferential loss treatment entitled to a traffic class is limited to a predetermined amount, while the amount of traffic exceeding such a limit will receive the *exact same* loss performance that the next traffic class lower in priority receives.

When incorporating an input rate limit into the original algorithm, it is important to preserve the following properties of the original BRD:

Property 1: Relativeness The extended BRD must satisfy the relative loss performance guarantees.

Property 2: Monotonicity The target loss rate of each class should remain a monotone non-decreasing function of the input traffic rates. i.e., for any two input traffic rate vectors $\bar{r}$ and $\bar{r}'$ [4], we require if $\bar{r} \leq \bar{r}'$, then $p_i(\bar{r}) \leq p_i(\bar{r}'), \forall i \in [1, N]$.

In the extended BRD, if Class i exceeds its input rate limit, denoted as $rlim_i$, its excess traffic is first degraded to Class $i+1$. If the resulting *total input* rate of Class $i+1$ (the original Class $i+1$ traffic plus the exceeding part of traffic from Class i) exceeds the rate limit of Class $i+1$, then the corresponding excess traffic (consisting of both the excess Class i and original Class $i+1$ traffic in the *same proportion* as in the total Class $i+1$ traffic) is further degraded to Class $i+2$, and so on. By doing so, we essentially transform the original vector of input rates $\bar{r}$ to a vector of effective input rates $\bar{r}'$, such that $\bar{r}' \leq \overline{rlim}$, $\overline{rlim} = (rlim_1, \ldots, rlim_N)$ [5]. The optimal target loss rates for the effective input $\bar{r}'$ are then computed using Eq. (7). Finally, the target loss rate of each class is determined based on its optimal target loss rate, the effective input $\bar{r}'$, and the transformation from $\bar{r}$ to $\bar{r}'$.

The extended BRD algorithm not only prevents the higher priority classes from starving the lower priority classes, it also satisfies both of the above properties [6]. The detailed description of the extended BRD is as follows:

[4] Note that $\bar{r}$ and $\bar{r}'$ are N-dimensional vectors, i.e., $\bar{r} = (r_1, \ldots, r_N), \bar{r}' = (r'_1, \ldots, r'_N)$ and we assume that $\bar{r} \leq \bar{r}'$ means $r_j \leq r'_j, \forall j \in [1, N]$

[5] There is no input rate limit on Class N, i.e.,$rlim_N = \infty$

[6] See [23] for discussions on alternative ways of enforcing input rate limits, which violate Property 1 and 2.

$C = 10$ Mbps, $LB_1 = 1\%$, $LB_2 = 10\%$, $r_2 = 5$ Mbps, $r_3 = 3$ Mbps.
Class 1 limit 4 Mbps, Class 2 input limit 5 Mbps.

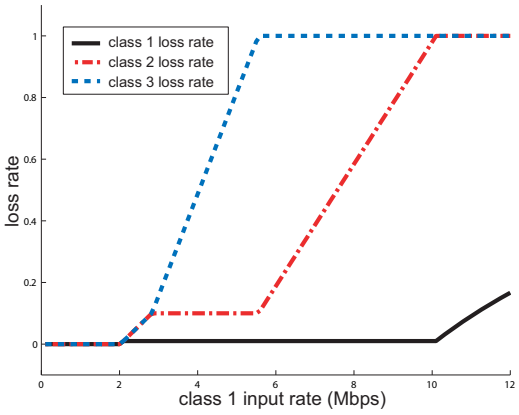

Fig. 1. Loss rate by the original BRD.

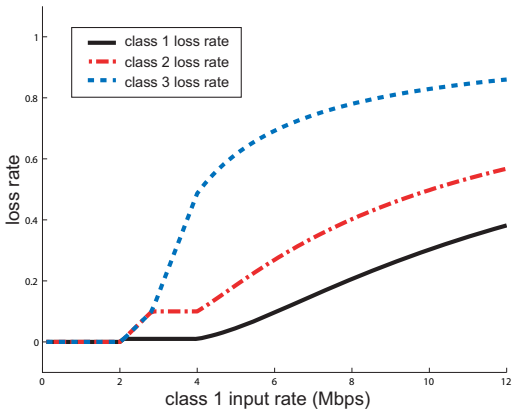

Fig. 2. Loss rate by the extended BRD.

(1) (a) Estimate the input rate $\bar{r} = (r_1, \ldots, r_N)$ as step (1) in the original BRD.
(b) Compute the effective input rate $\bar{r}' = (r'_1, \ldots, r'_N)$ using

$$\begin{aligned} r'_1 &= \min\left(r_1,\ \ rlim_1\right) \\ r'_i &= \min\left(\sum_{k=1}^{i} r_k - \sum_{k=1}^{i-1} r'_k,\ \ rlim_i\right), \quad 1 < i < N \\ r'_N &= \sum_{k=1}^{N} r_k - \sum_{k=1}^{N-1} r'_k \end{aligned} \tag{8}$$

Furthermore, $\forall i \le j$, the amount of traffic that originally belongs to Class i but becomes part of the effective input r'_j is denoted as r_{ij}. We have

$$r_{ij} = \begin{cases} r_N & \text{if } i = j = N; \\ \frac{r_i r'_i}{\sum_{k=1}^{i} r_k - \sum_{k=1}^{i-1} r'_k} & \text{if } 1 \le i = j < N; \\ \frac{r_i r_{(i+1)j} \sum_{k=1}^{i} (r_k - r'_k)}{r_{i+1}\left(\sum_{k=1}^{i} r_k - \sum_{k=1}^{i-1} r'_k\right)} & \text{if } 1 \le i < j \le N. \end{cases} \tag{9}$$

(c) Compute the optimal $\bar{p}'$ based on Eq. (7) and using $\bar{r}'$.
(d) Compute the actual target loss rate of Class i as

$$p_i = \begin{cases} p'_N & \text{if } i = N; \\ \sum_{j=i}^{N} \frac{r_{ij}}{r_i} p'_j & \text{if } i = 1, \ldots, N-1 \end{cases} \tag{10}$$

(2) Upon the arrival of a packet belonging to class i, the packet is dropped with probability p_i, otherwise it enters the buffer. Again, the random packet dropping is executed only when the buffer is more than 50% full.

We show next through a simple example involving only $N = 3$ traffic classes, the effectiveness of the extended BRD in mitigating the impact of higher priority

classes. Assume $C = 10$ Mbps, $LB_1 = 1\%$, $LB_2 = 10\%$, and no loss bound on Class 3. The target loss rates computed by the original BRD for the three classes are shown in Fig. 1, as a function of the Class 1 rate. Notice that the impact of Class 1 over Classes 2 and 3 is significant as the increase in Class 1 input quickly increases the loss rates of Class 3 and then Class 2, to 100%. This impact is mitigated when the extended BRD is used. Fig. 2 shows the target loss rates of the three classes when an input limit of 4 Mbps is imposed on Class 1, and an input limit of 5 Mbps is imposed on Class 2. Fig. 2 also illustrates that the extended BRD satisfies both Properties 1 and 2.

5 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 specifically designed to illustrate when and why BRD is better suited to meet the 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 our experiments, 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 $\alpha = 0.125$ and $\Delta t = 1$ms. Actual loss rates are monitored and computed every 1ms.

5.1 Scenario 1: Performance comparison with JoBS and PractQoS

In comparing the performance of BRD to that of JoBS and PractQoS, we used the JoBS module implemented in ns-2.26 [24] and implemented a module for PractQoS following the specifications put forth in [8]. 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. 3 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. 4. Before time 200s, the total input is 10.47 Mbps, which allows 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. 5(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, Class 2 increases its rate so that it is impossible to satisfy the loss bounds of both Class 1 and 2 simultaneously, even after dropping all Class 3 packets. Class 2's loss bound will, therefore, be relaxed beyond 20%. Dropping all Class 3 packets because of the rate

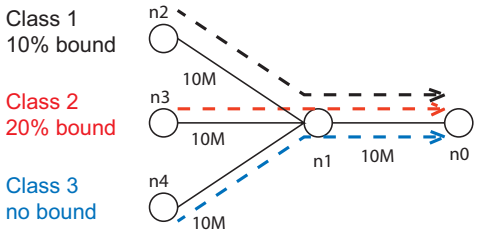

Fig. 3. Scenario 1: configuration.

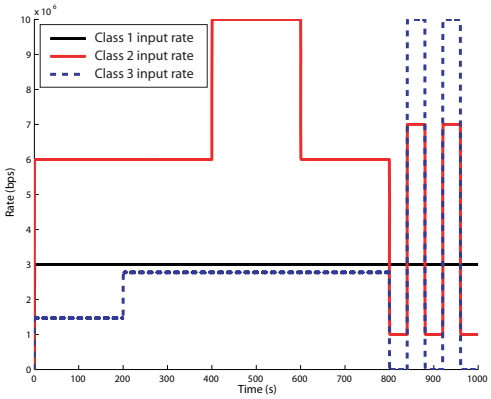

Fig. 4. Scenario 1: Input rates.

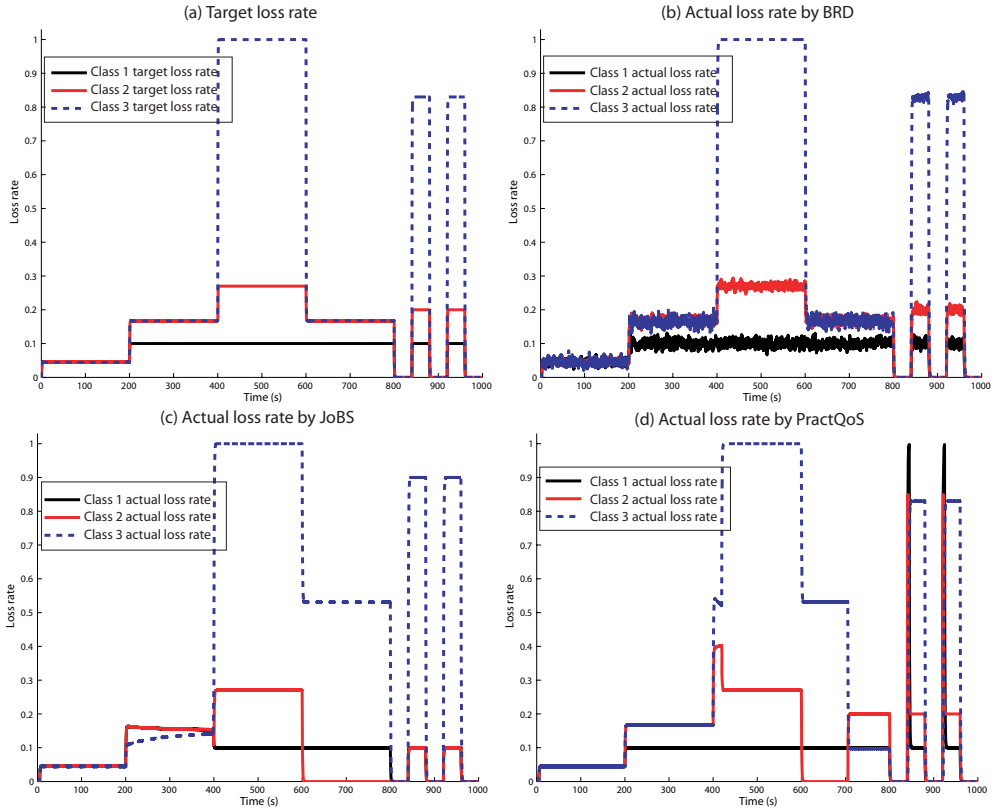

Fig. 5. Scenario 1: Performance differences between BRD, JoBS and PractQoS.

increase in Class 2 can be avoided by introducing input rate limits as mentioned in Section 4.3, and we discuss this later in this section. At time 600s, the input of Class 2 drops back down to 6 Mbps, and the target loss rates of the three classes return to the values they had during $[200, 400]$s. Finally, during time $[800, 1000]$s, the input rates of all three classes exhibit on-off behavior causing sudden bouts of congestion with link load fluctuating between 40% and 200%. The target loss rates of the three classes decrease to 0 when the link load is 40%, but increase to 10%, 20% and 83% for Class 1, 2 and 3, respectively, when the link load is 200%. To summarize, the target loss rates are shown in Fig. 5(a).

In evaluating the performance of the three schemes for the above scenario, we see from Fig. 5(b) 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. 5 (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 guarantees, 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. 5(c), during which Class 1 violates its loss target; and during $[600, 1000]$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. The fact that Class 2 receives better than necessary loss 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 $[600, 1000]$s. Therefore, during $[600, 1000]$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 largest among the three classes. As a result, Class 2 receives better than necessary performance during $[600, 1000]$s, which is achieved at the cost of a much higher loss rate in Class 3. This also violates the relative loss guarantees by allowing Class 2 to enjoy better loss performance than Class 1 does during $[600, 800]$s.

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. 5(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 [8], 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

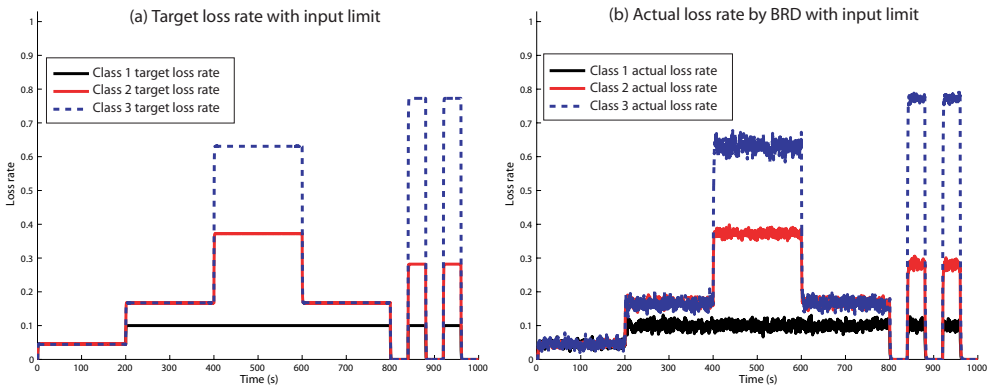

Fig. 6. Scenario 1 with input rate limit. Class 1 limit 3 Mbps, Class 2 limit 6 Mbps.

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 guarantees, and this aspect is not fully specified in [8]. If the same method as JoBS is used, PractQoS will exhibit a similar behavior. In Fig. 5(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 guarantees by allowing Class 3 to experience a smaller short-term loss rate than that of Class 2 during the time $[700, 800]$s.

As shown previously, the input rate increase in Class 2 during time $[400, 600]$s causes Class 3 to suffer a 100% loss rate. However, this can be avoided by imposing input rate limits, e.g., a 3 Mbps limit on Class 1 and a 6 Mbps limit on Class 2. Under such limits and based on the input limit algorithm explained in Section 4.3, the target loss rates and the actual loss rates for the three classes are shown in Fig. 6(a) and (b), respectively. Notice that the input rate limit affects the loss rates of the three classes only when necessary, i.e., during $[400, 600]$s, and the loss rates are adjusted in a way that satisfies both the relativeness and monotonicity properties.

Finally, we investigate by means of simulations the impact of the parameters α and Δt on BRD's performance. To this end, we use the same configuration as that of Scenario 1, except that the input rate of Class 1 experiences a spike at time 100s, as shown in Fig. 7 (a). Such a spike in traffic rate is a transient rate surge, that ideally should be ignored by BRD when computing loss probabilities. Fig. 7 (b) shows that, with proper parameter choices, specifically $\alpha = 0.125$ and $\Delta t = 1ms$, BRD is almost unaffected by the rate spike while responding in a timely fashion to normal traffic rate variations. Different choices of α and Δt can, however, affect performance as illustrated in Figs. 7 (c) and (d). A large Δt (combined with a small α), results in slow detection of rate variations causing loss rates to lag behind. Con-

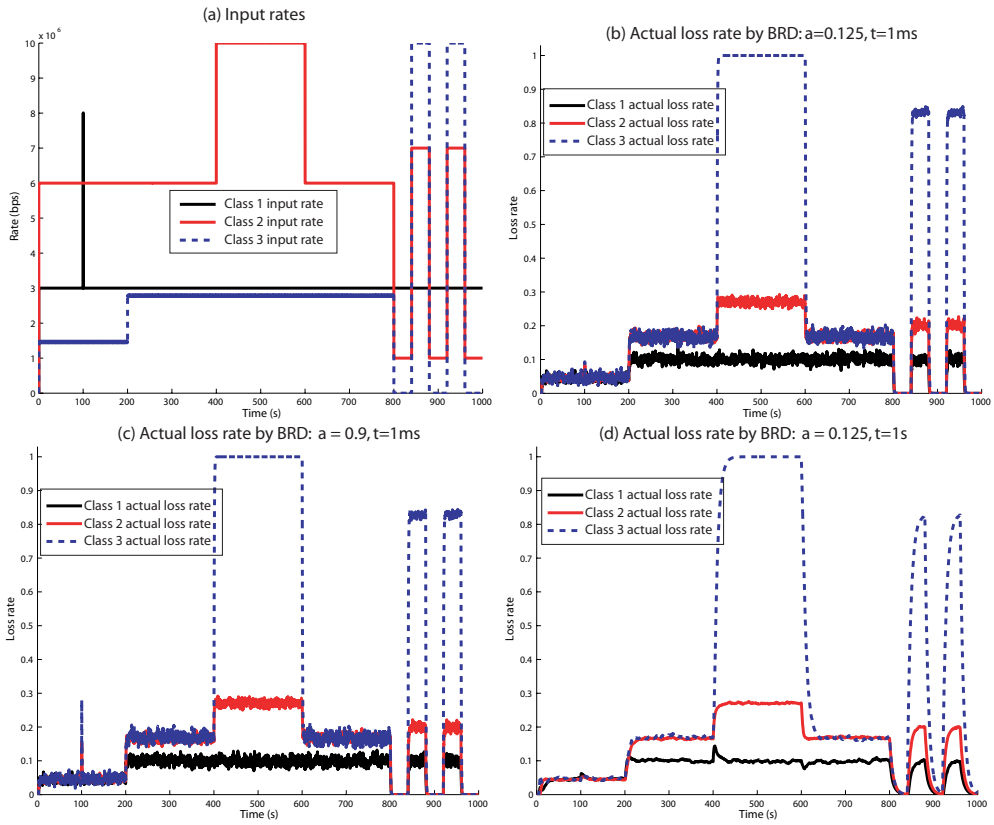

Fig. 7. Scenario 1: Impact of α and Δt.

versely, too large an α makes BRD too responsive to rate changes, to the point that loss rates are now affected by the transient rate spike of Class 1. In general, and although BRD's performance is not overly sensitive to the exact choice of parameters α and Δt, we have found that a relatively small α, i.e., around 0.125, together with a reasonable monitoring interval, e.g., $\Delta t \approx 1ms$, perform well over a wide range of traffic conditions.

In summary, the investigation of Scenario 1 has 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 guarantees interfered with the scheme's target loss proportions. The overall structure and mechanisms used by both JoBS and PractQoS are 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. We proceed next with further investigations of BRD's performance with scenarios involving more realistic traffic mixes and configurations.

5.2 Scenario 2: Additional Performance Investigations

In this section, we investigate the performance of BRD using what we consider to be a realistic traffic configuration consisting of both short-lived and long-lived TCP traffic, as well as UDP video traffic. Our investigation is carried out using "actual" traffic sources. For the UDP video traffic, we use MPEG-4 video traces of the movie "Jurassic Park I" [25]. Long-lived TCP flows are generated using simulated FTP connections, while short-lived TCP flows are generated using both simulated Web connections and short-lived exponential on-off TCP connections. In terms of performance, we focus not only on losses, but also on performance measures such as TCP throughput, HTTP response time, and FTP file transfer times. We compare and contrast these 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, short-lived TCP traffic generated by interactive applications that require low delay is particularly sensitive because losses are more likely to cause TCP timeouts 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.

As shown in Fig. 8, our setup involves a mixture of all three traffic types [7] . 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 sensitive to losses as was mentioned before, and we assume they are also most important to service subscribers. 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 web traffic. We assume that it is of least importance in this setting. Therefore, no loss bound is required for this Class 3 traffic.

As shown in Fig. 8, 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 of a cross continental path. We implement BRD on $n4$

[7] See [23] for additional results focusing on BRD's performance in scenarios involving only one type of traffic.

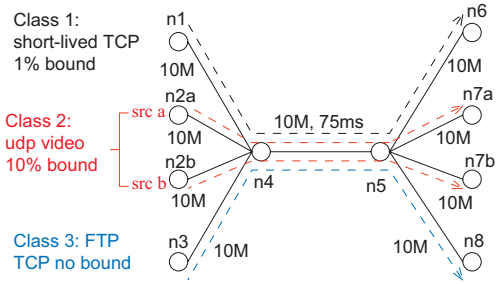

Fig. 8. Scenario 2: configuration.

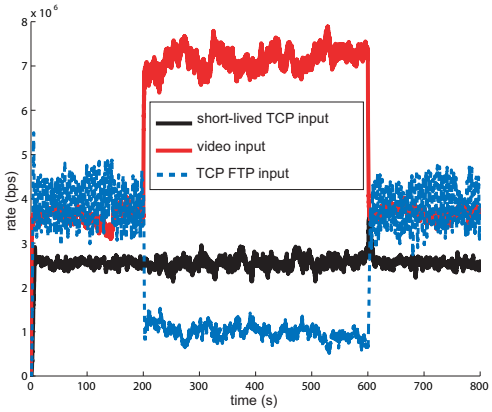

Fig. 9. Scenario 2: Input rates by BRD.

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 are active except connections from node $n2b$, which are only active in the $[200, 600]$s interval, and therefore 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 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 [24] connections. Both of the HTTP connections generate HTTP requests with exponentially distributed inter-arrival time. 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 web pages 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 Queue scheme that is commonly used to provide service differentiation. In the Priority Queue 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 Queue 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 queue is full.

The input rate of the three classes using BRD is shown in Fig. 9. 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. 9 and 10. Although Class 1 is also protected from the lower priority classes when the Priority Queue is used, the throughput of Class 3 is much lower in that case. As we can see in Fig. 11, the throughput of Class 3 is actually

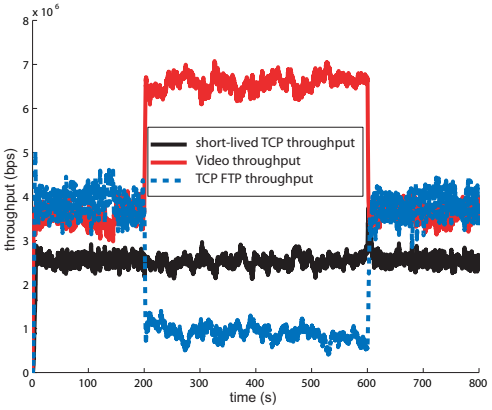

Fig. 10. Scenario 2: Throughput by BRD.

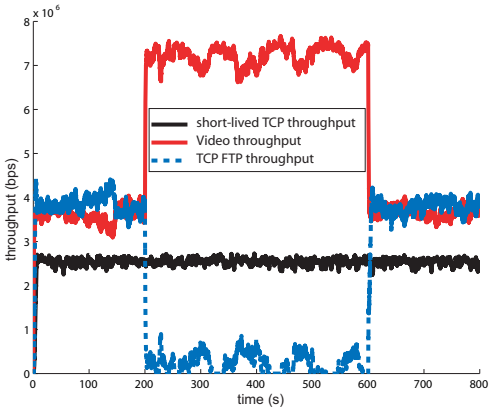

Fig. 11. Scenario 2: Throughput by Priority Queue.

close to zero during time $[200, 600]$s. This illustrates a key deficiency of a Priority Queue scheme when compared to BRD in that it "over-penalizes" lower classes by giving unnecessarily good performance to higher priority classes. In [23], we also examined the throughput achieved by JoBS and PractQoS in this scenario, and confirmed that PractQoS performs better than JoBS, but not as good as BRD.

Next, we further quantify the performance difference by looking at HTTP response times and FTP file transfer times. The average HTTP response time is an important factor 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 1 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 re-

Table 1
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)	8402	8456	2233	2233
Avg. no. of images	4	62	70	70
Class priority	1	3	3	1
Response time by BRD (s)	3.3353	111.0078	43.3506	11.6489
Response time by the PQ (s)	1.4775	238.7476	90.1861	5.1734

Table 2
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 PQ (s)	401.9695	179.744	45.3731

sponse 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 (3.3s 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 when it comes to the performance seen by normal web traffic, such as browsing the front page of `www.cnn.com` or `www.amazon.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 shown in Table 1, when this traffic is sent as Class 3 traffic, the response time is about 43s for BRD versus 90s with Priority Queue. Similarly, the transfer of the (very large) Amazon front page is 111s with BRD and 238s with Priority Queue. 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 is carried by Class 3, as this is another important measure of the cost of giving better service to other classes. As shown in Table 2, BRD again is able to mitigate the performance degradation experienced by Class 3, i.e., provide better loss performance to Class 3, while offering Class 1 (and 2) a level of service comparable to that of a priority queue scheme. This is consistent with the difference in throughput between BRD and Priority Queue when comparing Fig. 10 and 11.

6 Conclusion

In this paper, we investigated the feasibility of providing strong loss differentiation at a low added cost. We proposed a new scheme called BRD that provides a relative service quality order across traffic classes. BRD guarantees each class losses that, when feasible, are no worse than a specified bound, and it enforces differentiation only when required to meet those bounds. This is 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, such as JoBS and PractQoS. From a performance perspective, BRD is capable of providing both long-term and short-term loss guarantees by relying on directly estimating the arrival rates. In contrast, as illustrated in Section 5.1, 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 reliance on the loss process itself as the base for making dropping decisions. From a complexity standpoint, 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 relatively low cost.

Appendix A

Proof (Theorem 1): The space of all possible input rates $\bar{r} = (r_1, r_2, \ldots, r_N)$ can be partitioned into the following $2N$ regions $R_0, R_1, \ldots, R_{2N-1}$ defined as:

$$R_0 = \left\{(r_1, \ldots, r_N) : \sum_{i=1}^{N} r_i \leq C\right\}$$

$$R_1 = \left\{(r_1, \ldots, r_N) : 0 < 1 - \frac{C}{\sum_{i=1}^{N} r_i} \leq LB_1\right\}$$

for $k = 1, \ldots, N-2$

$$R_{k+1} = \left\{(r_1, \ldots, r_N) : LB_k < 1 - \frac{C - \sum_{i=1}^{k} r_i(1 - LB_i)}{\sum_{i=k+1}^{N} r_i} \leq LB_{k+1}\right\}$$

$$R_N = \left\{(r_1, \ldots, r_N) : LB_{N-1} < 1 - \frac{C - \sum_{i=1}^{N-1} r_i(1 - LB_i)}{r_N} \leq 1\right\}$$

for $l = 1, \ldots, N-1$

$$R_{N+l} = \left\{(r_1, \ldots, r_N) : LB_{N-l} < 1 - \frac{C - \sum_{i=1}^{N-l-1} r_i(1 - LB_i)}{r_{N-l}} \leq 1\right\}$$

We now prove that in every one of these $2N$ regions, any feasible solution $\overline{p'} \neq \overline{p^*}$ yields $f(\overline{p'}) > f(\overline{p^*})$, in which $f(\overline{p}) = \min \sum_{i=1}^{N-1}(p_{i+1} - p_i) = \min(p_N - p_1)$.

- **In Region R_0:**
  We have $\sum_{i=1}^{N} r_i \leq C$. By Eq. (3), $\overline{p^*} = \overline{0}$ is the only feasible solution.
- **In Region R_1:**
  We have $\sum_{i=1}^{N} r_i > C$. If $f(\overline{p'}) = f(\overline{p^*}) = 0$, by Eq. (5), we have $p'_1 = \cdots = p'_N$. By Eq. (3), $r_i(1 - p'_i) = C/N = r_i(1 - p^*_i)$, $\forall i \in [1, N]$. This contradicts $\overline{p'} \neq \overline{p^*}$. Therefore, $\overline{p^*}$ is the unique optimal solution.
- **In Region R_{k+1}, for $k = 1, \ldots, N-1$:**
  We have $\sum_{i=1}^{N} r_i > C$. By Eq. (3), $\sum_{i=1}^{N} r_i(1 - p'_i) = \sum_{i=1}^{N} r_i(1 - p^*_i) = C$. Since $\overline{p'} \neq \overline{p^*}$, there exist j, $1 \leq j \leq N$ such that $j = \min\{i : p'_i > p^*_i\}$.
  If $1 \leq j \leq k$, then $p'_j > p^*_j = LB_j$. By Eq. (6), we have $p'_i = LB_i = p^*_i$, for $1 \leq i < j$, and $p'_{j+1} = \cdots = p'_N = 1$. However, this contradicts Eq. (3) as $\sum_{i=1}^{N} r_i(1 - p^*_i) - \sum_{i=1}^{N} r_i(1 - p'_i) = r_j(p'_i - LB_j) + \sum_{i=j+1}^{N} r_i(1 - LB_i) > 0$.
  If $k < j \leq N$, we have $p'_1 \leq p^*_1$ and $p'_N \geq p'_j > p^*_j = p^*_N$. As a result $f(\overline{p'}) = p'_N - p'_1 > f(\overline{p^*})$.
  Therefore, $\overline{p^*}$ is uniquely optimal in region R_{k+1}, for $k = 1, \ldots, N-1$.
- **In Region R_{N+l}, for $l = 1, \ldots, N-1$:**
  Again, because of $\sum_{i=1}^{N} r_i > C$ and Eq. (3), there exists a j, $1 \leq j \leq N$ such that $j = \min\{i : p'_i > p^*_i\}$. Since $p^*_i = 1$, for $i = N - l + 1, \ldots, N$, we can only have $1 \leq j \leq N - l$, because of Eq. (4).
  When $1 \leq j \leq N - l$, we have $p'_j > p^*_j \geq LB_j$. By Eq. (6), we have $p'_i = LB_i = p^*_i$, for $1 \leq i < j$, and $p'_{j+1} = \cdots = p'_N = 1$. However, this contradicts Eq. (3) as $\sum_{i=1}^{N} r_i(1-p'_i) - \sum_{i=1}^{N} r_i(1-p^*_i) = r_j(p^*_j - p'_j) + \sum_{i=j+1}^{N} r_i(p^*_i - 1) < 0$.

In summary, across all possible $2N$ regions, $\overline{p^*}$ is uniquely optimal.

References

[1] C. Dovrolis, D. Stiliadis, P. Ramanathan, Proportional differentiated services: Delay differentiation and packet scheduling, IEEE/ACM Transactions on Networking 10 (1) (2002) 12–26.

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

[3] S. Iyer, R. R. Kompella, N. McKeown, Analysis of a memory architecture for fast packet buffers, in Proc. of IEEE Workshop on High Performance Switching and Routing, Dallas, TX, May 2001.

[4] C. Dovrolis, P. Ramanathan, A case for relative differentiated services and the proportional differentiation model, IEEE Network Magazine 13 (5) (1999) 26–34.

[5] J. Liebeherr, N. Christin, Rate allocation and buffer management for differentiated services, Computer Networks 40 (1) (2002) 89–110.

[6] N. Christin, J. Liebeherr, T. Abdelzaher, A quantitative assured forwarding service, in Proc. of IEEE Infocom 2002, New York, NY, June 2000 2 (2002) 864–873.

[7] J. Liebeherr, N. Christin, A QoS architecture for quantitative service differentiation, IEEE Commununications Magazine 41 (6) (2002) 38–45.

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

[9] Y. Chen, C. Qiao, M. Hamdi, D. H. K. Tsang, Proportional differentiation: A scalable QoS approach, IEEE Commununications Magazine 41 (6) (2003) 52–59.

[10] K. Thompson, G. J. Miller, R. Wilder, Wide-area internet traffic patterns and characteristics, IEEE Network Magazine 11 (6) (1997) 10–23.

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

[12] F. Kamoun, L. Kleinrock, Analysis of shared finite storage in a computer network node environment under general traffic conditions, IEEE Transactions on Communications COM-28 (7) (1980) 992–1003.

[13] R. Guérin, V. Peris, Quality-of-service in packet networks: Basic mechanisms and directions, Computer Networks 31 (3) (1999) 169–179.

[14] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking 1 (4) (1993) 397–413.

[15] R. Pan, B. Prabhakar, 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.

[16] Weighted random early detection, [online], http://www.cisco.com/.

[17] S. Sahu, P. Nain, C. Diot, V. Firoiu, D. F. Towsley, On achievable service differentiation with token bucket marking for TCP, Measurement and Modeling of Computer Systems (2000) 23–33.

[18] W. Wu, Y. Ren, 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.

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

[20] J. Shin, J.-G. Kim, J. Kim, C.-C. J. Kuo, Dynamic QoS mapping control for streaming video in relative service differentiation networks, European Transactions on Telecommunications 12 (3) (2001) 217–230.

[21] C. Dovrolis, P. Ramanathan, Dynamic class selection: from relative differentiation to absolute QoS, in Proc. of IEEE ICNP 2001, Riverside, CA, November 2001.

[22] U. Bodin, A. Jonsson, O. Schelen, On creating proportional loss-rate differentiation: Predictability and performance, in Proc. of IWQoS 2001, Karlsruhe, Germany, June 2001 (2001) 372–388.

[23] Y. Huang, R. Guérin, A simple FIFO-based scheme for differentiated loss guarantees, Tech. rep., University of Pennsylvania (March. 2006).

[24] VINT Project, The network simulator NS-2, [online] ,http://www.isi.edu/nsnam/vint.

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