Enhancing Class-Based Service Architectures with Adaptive Rate Allocation and Dropping Mechanisms[*]

Nicolas Christin
School of Information
Management and Systems
UC Berkeley
Berkeley, CA 94720

Jörg Liebeherr   *Tarek F. Abdelzaher*
Department of Computer Science
University of Virginia
Charlottesville, VA 22904

Abstract

Class-based service differentiation can be realized without resource reservation, admission control and traffic policing. However, the resulting service guarantees are only relative, in the sense that guarantees given to a flow class at any time are expressed with reference to the service given to other flow classes. While it is, in principle, not feasible to provision for absolute guarantees (i.e., to assure lower bounds on service metrics at all times) without admission control and/or traffic policing, we will show in this paper that such a service can be reasonably well emulated using adaptive rate allocation and dropping mechanisms at the link schedulers of routers. We name the resulting type of guarantees *best-effort bounds*. We propose mechanisms for link schedulers of routers that achieve these and other guarantees by adjusting the drop rates and the service rate allocations of traffic classes to current load conditions. The mechanisms are rooted in control theory and employ adaptive feedback loops. We demonstrate that these mechanisms can realize many recently proposed approaches to class-based service differentiation. The effectiveness of the proposed mechanisms are evaluated in simulation experiments as well as in measurement experiments of a kernel-level implementation on FreeBSD PC-routers with multiple 100 Mbps Ethernet interfaces.

Key Words: Service Differentiation, Buffer Management, Scheduling, Feedback Control, Best-Effort Bounds.

1 Introduction

Service architectures for packet networks can be distinguished according to two criteria. The first criterion is whether guarantees are expressed for individual traffic flows (*per-flow guarantees*), or for aggregates of flows with the same service requirements (*per-class guarantees*). With a per-flow architecture, a router must inspect each incoming packet to determine to which flow the packet belongs and match the packet with per-flow guarantees (classification). Generally, the classification overhead increases linearly with the number of flows present in the network. With per-class guarantees, flows are grouped in traffic classes. Each packet entering the network is marked with the traffic class to which it belongs, and routers in the network classify and transmit packets according to the service guarantees offered to traffic classes. Since there are generally only a few traffic classes in the network, the overhead incurred with per-class guarantees is smaller than that of per-flow guarantees. As a disadvantage, per-class service guarantees do not immediately translate into per-flow guarantees.

The second criterion to distinguish service architectures is whether guarantees are expressed with reference to guarantees given to other flows or classes (*relative guarantees*), or if guarantees are expressed as absolute bounds (*absolute guarantees*). As an example, an absolute guarantee can be of the form "Delay of flow i never exceeds 4 ms." Relative service guarantees are weaker than absolute guarantees, and can be further divided into qualitative guarantees and proportional guarantees. Qualitative guarantees specify a service differentiation of classes, but without quantifying the differentiation, as in "Class-2 delay is less than class-1 delay." Proportional guarantees quantify the differentiation between traffic classes in terms of ratios of the service metrics, as in "Class-1 delay is half of class-2 delay," but without specifying lower or upper bounds on the ratios.

The main advantage of absolute guarantees is that they provide lower bounds on the service received by a flow or a class of traffic. However, absolute guarantees impose a need to dedicate resources to traffic. This involves mechanisms to control the amount of traffic that enters the network, via admission control and traffic policing. Resource reservation schemes have been proposed for flow-based and class-based guarantees, where resource reservations are handled by a signaling protocol [11], a dedicated server [44], resource provisioning [18], or manual configuration [44]. Relative guarantees, on the other hand, do not require resource reservations, and, therefore, do not need admission control or traffic policing. Relative guarantees can be provided by appropriate scheduling and buffer management mechanisms at routers.

This paper is concerned with improving the capabilities of class-based service architectures for the Internet. The class-based service architecture proposed by the Internet Engineering Task Force, called Differentiated Services or *DiffServ* [8], consists of two services. The Expedited Forwarding (EF, [21]) service provides absolute delay guarantees to predefined amounts of traffic, and requires traffic policing, admission control, and resource reservations. The Assured Forwarding (AF, [30]) service enforces isolation between classes, and qualitative loss differentiation between different drop precedence levels within each class. The Proportional Service Differentiation architecture [22, 25] showed how to strengthen Assured Forwarding by adding proportional guarantees on delays and loss rates. Recently, several research efforts have explored

how to further enhance class-based services. Specifically, attempts have been made to support some level of absolute guarantees, yet without requiring resource reservation, admission control, or traffic policing [8, 29, 32, 36, 37]. Clearly, without asserting control over the amount of traffic injected in the network (through admission control and policing) it is not feasible to guarantee absolute guarantees at all times. On the other hand, if one permits routers to selectively drop traffic, one can provide absolute guarantees to the traffic that is not dropped. The Alternative Best-Effort (ABE, [32]) is an example of such a service. ABE supports differentiation for two traffic classes, where the first class obtains an absolute delay bound, and the second class is given a better loss rate than the first class, but has no delay guarantees. To meet these guarantees, ABE is permitted to drop any amount of traffic from the first class.

In this paper, we generalize the enhancements to class-based service differentiation proposed in the literature [8, 29, 32, 36, 37] by introducing the notion of **best-effort bounds**. We refer to a service with best-effort bounds as a service that emulates absolute guarantees in a network without admission control and policing. The difference between absolute guarantees and best-effort bounds is that the former assumes a network with admission control and policing. By limiting the number of flows via admission control and by limiting the amount of traffic per flow via policing, such a network can deliver absolute guarantees at all times. In contrast, a network with best-effort bounds achieves absolute guarantees by dropping traffic or changing its traffic rate allocation. In situations when this is not feasible, a best-effort bound may be violated for some time. Best-effort bounds can be verified by comparing them to absolute guarantees in a reference network with admission control and policing. If the reference network can support a set of absolute guarantees for a certain amount of traffic, then the same network without admission control should be able to satisfy the corresponding best-effort bounds.

Best-effort bounds are much weaker than the corresponding absolute guarantees. On the other hand, best-effort bounds enhance the existing framework of feasible class-based guarantees without introducing a need for mechanisms to control the amount of traffic entering the network. Given that a service with absolute guarantees at all times requires admission control and policing, best-effort bounds are possibly the closest approximation of such a service in a network without these mechanisms. As we will show in this paper, even in times of high traffic load, appropriate adaptive rate allocation and dropping mechanisms can enforce a wide range of best-effort bounds and provide proportional service differentiation at the same time, thereby generalizing the service differentiation offered by any of the previously proposed class-based services.

The main challenge for realizing best-effort bounds is to find mechanisms for routers that can meet a wide range of bounds for a large number of classes by selectively dropping traffic and by adjusting the traffic rate allocated to a class. The main contribution of this paper is that we propose and evaluate such mechanisms which can meet a broad range of best-effort bounds as well as proportional guarantees on delay, loss, and throughput. The mechanisms employ adaptive feedback loops at link schedulers of routers, which adjust the drop rates and the service rate allocations of traffic classes to current load conditions. To our knowledge, the feasibility of using packet-level feedback loops at high data rates for the purpose of service differentiation has not been demonstrated. We evaluate the effectiveness of our adaptive rate allocation and dropping mechanisms in simulation experiments as well as in a kernel-level software implementation in

FreeBSD PC routers. This implementation is currently being disseminated as part of the ALTQ [12] and KAME [4] packages.

The remainder of this paper is organized as follows. In Section 2, we expand our discussion of the related work. In Section 3, we present a formal description of the proposed service. In Sections 4 and 5, we discuss the mechanisms that enforce the desired differentiation of loss, delay and throughput for classes by adjusting the service rate allocation to classes and by selectively dropping traffic. We apply linear feedback control theory for the design of these mechanisms. In Section 6, we present an implementation of the mechanisms in FreeBSD PC-routers. We evaluate our implementation in Section 7 and present brief conclusions in Section 8.

2   Related Work

Proportional service differentiation, originally proposed by Dovrolis et al. [22, 23, 24, 25] is perhaps the best known effort to enhance class-based services with relative guarantees. In the proportional service differentiation architecture, relative differentiation of losses and delays experienced by traffic classes, as in "Class-2 delay $\geq$ class-3 delay," is guaranteed under any traffic load. Furthermore, proportional differentiation of loss and delay, as in "Class-2 loss / Class-3 loss = 2," is enforced whenever feasible.

Most mechanisms for proportional service differentiation use independent algorithms for delay and loss differentiation. Proportional differentiation of delays can be implemented with appropriate scheduling algorithms. Priority-based scheduling algorithms such as Waiting-Time Priority, Hybrid Proportional Delay [25], Local-Optimal Proportional Differentiation [48], the dynamic priority scheduler proposed in [26] or Mean-Delay Proportional [42] can enforce proportional delay differentiation by dynamically adjusting the priority of a given class as a function of the waiting-time experienced by packets from that class. Alternatively, rate-based schedulers such as the Proportional Queue Control Mechanism [41], or Backlog Proportional Rate [24] can be used to provide proportional delay differentiation, by dynamically changing the service rates allocated to classes. A slightly different approach pursued by the Weighted-Earliest-Due-Date scheduler of [9] provides proportional differentiation in terms of probabilities of a deadline violation.

Proportional loss differentiation can be implemented by buffer management algorithms that choose which class to drop from in order to reach steady-state proportional loss differentiation [22]. Enhancements to the mechanisms discussed in [22] can provide proportional loss differentiation over arbitrary timescales [10].

More recent works have attempted to expand the range of traffic conditions under which proportional service differentiation can be enforced, by combining the scheduling and dropping decisions in a single algorithm [37, 52]. For instance, in [52], packet drops and packet transmissions are viewed as transitions in a state diagram, where states represent the experienced level of delay and loss differentiation. Packet scheduling and dropping is performed to reach states that match the desired proportional delay and loss differentiation.

The service proposed in [37] further enhances class-based service differentiation by providing limited

support for absolute bounds on loss and delay. To that effect, the authors of [37] present a Joint Buffer Management and Scheduling algorithm (JoBS), which expresses the scheduling and dropping decisions as the solution to an optimization problem, whose constraints are defined by the service guarantees, and the objective function aims at minimizing packet losses and changes in the rate allocation. The drawback of JoBS is that solving a non-linear optimization problem, even if approximated by a heuristic method [37], can incur a significant computational overhead when performed on a per-packet basis.

There are many other service proposals (e.g., ABE) that have explored the design space of class-based architectures and we refer the reader to [13] for a more comprehensive discussion. For instance, the Dynamic Core Provisioning service [36] supports absolute delay bounds, and qualitative loss and throughput differentiation, but no proportional differentiation. The mechanisms used in [36] enforce service guarantees by dynamically adjusting scheduler service weights and packet dropping thresholds in core routers. Traffic aggregates are dimensioned at the network ingress by a distributed admission control mechanism that uses knowledge of the entire traffic present in the network. Since, in practice, full knowledge of the traffic traversing a network is generally not available, the algorithm needs to be approximated when deployed in a large network.

The majority of related work focuses on particular scheduling and dropping algorithms and investigates the degree to which class-based service guarantees can be enhanced with the proposed algorithms. The work presented in this paper takes a different approach. We first state the desired service guarantees (a superset of the guarantees of all works cited above), then formulate requirements on rate allocation and dropping mechanisms, and, eventually, arrive at mechanisms that satisfy the specified requirements.

3   Class-Based Service with Adaptive Rate Allocation and Dropping

In this section, we describe a service that, in the absence of admission control, traffic policing, signaling or resource reservation, offers both best-effort bounds and proportional differentiation to traffic classes. The proposed service gives, on a per-hop basis, best-effort bounds and proportional service guarantees to traffic classes. All guarantees can be expressed for loss rates, delays, or throughput, and are assumed to be configured on routers by a network operator.

Example: As an example for a mix of guarantees for three traffic classes, one could specify the following best-effort bounds for a network interface of a router:

  (G1) "Class-1 delay $\leq$ 2 ms,"
  (G2) "Class-2 loss rate $\leq$ 1%,"
  (G3) "Class-3 service rate $\geq$ 1 Mbps,"

and the following proportional guarantees:

(G4) "Class-2 delay/class-1 delay ≈ 4,"

(G5) "Class-3 loss rate/class-2 loss rate ≈ 2."

(G6) "Class-1 throughput/class-3 throughput ≈ 2."

Guarantee (G1) states that class-1 packets do not experience a delay greater than two milliseconds, (G2) ensures that the loss rate of class 2 never exceeds 1%, and (G3) states that the aggregate throughput of all flows in class 3 should be at least 1 Mbps. (G4) expresses that class-2 packets experience delays roughly twice as large as class-1 packets, (G5) states that class-3 packets experience twice the loss rate of class-2 packets, and finally, (G6) indicates that the aggregate throughput of all flows in class 1 should be twice as large as the aggregate throughput of all flows in class 3. When all best-effort bounds cannot be enforced simultaneously, the best effort bounds are relaxed in some order. Here, we specify that the best effort bounds should be relaxed in the order (G1), (G2) and (G3). Thus, if necessary, guarantee (G1) can be violated in order to meet guarantee (G3), and that (G3) will be violated last. Note that, as long as the available link bandwidth is at least 1 Mbps, (G3) can be satisfied at all times.

With these guarantees, we can emulate Assured Forwarding, by assigning each AF drop level to a separate traffic class. Further, we can implement ABE by selecting a delay bound for one class, and proportional differentiation yielding lower loss rates to another class. More generally, it can be argued that delay, loss, and throughput differentiation can be used to express guarantees on other service metrics, such as traffic burstiness [51].

We next give a formal description of the service, and outline a solution for an algorithm that realizes the service.

3.1 Formal Description of Rate Allocations and Dropping Decisions

The provisioning of per-class service differentiation in our proposed service is expressed in terms of the backlog behavior at a single transmission queue of the output link of a router. The discussion draws inspiration from Cruz's network calculus [19, 20]. We will refer to Figure 1 for an illustration.

We assume that all traffic that arrives to the transmission queue of the output link of a router is marked to belong to one of N classes. We use a convention whereby a class with a lower index receives a better service. We consider a discrete, event-driven time model, where events are traffic arrivals. We use $t(n)$ to denote the time of the n-th event in the current busy period,[1] and $\Delta t(n)$ to denote the time elapsed between the n-th and $(n+1)$-th events. We use $a_i(n)$ and $l_i(n)$, respectively, to denote the class-i arrivals and the amount of class-i traffic dropped (lost) at the n-th event. We use $r_i(n)$ to denote the service rate allocated to class-i at the time of the n-th event. The service rate of class i is a fraction of the output link capacity, and can vary over time. The service rate of class i is set to zero if there is no backlog of class-i traffic in the transmission queue. For the time being, we assume a fluid-flow service, that is, the output link is viewed as simultaneously serving traffic from several classes. Such a fluid-flow interpretation is idealistic, since

---

[1]The beginning of the current busy period is defined as the last time when the transmission queue at the output link was empty.
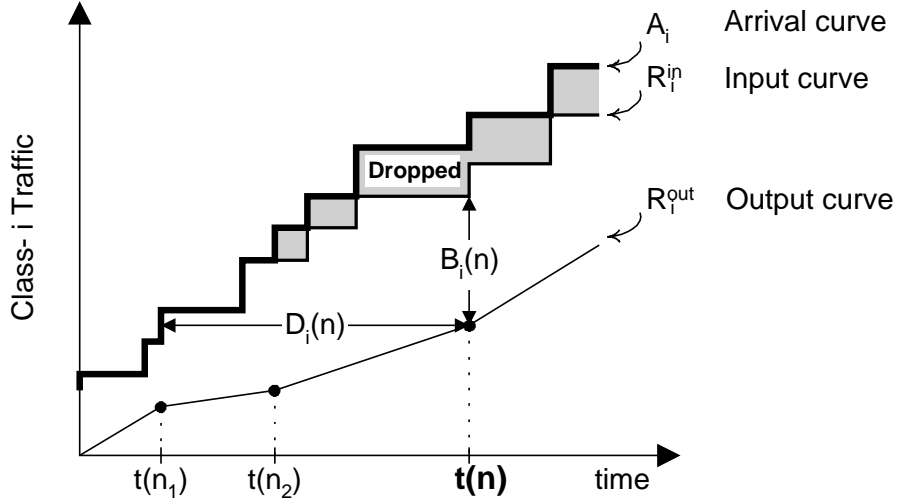
Figure 1: **Delay and backlog at the transmission queue of an output link.** A_i is the arrival curve, R_i^{in} is the input curve and R_i^{out} is the output curve.

traffic is actually sent in discrete sized packets. In Section 6, we discuss how the fluid-flow interpretation is realized in a packet network.

Service differentiation will be enforced over the duration of a busy period. An advantage of enforcing service differentiation over short time intervals is that the output link can react quickly to changes of the traffic load. Further, providing differentiation only within a busy period requires little state information, and, therefore, keeps the implementation overhead limited. As a possible disadvantage, at times of low load, when busy periods are short, providing service differentiation only with information on the current busy period can be unreliable. However, when busy periods are short, the transmission queue is generally underloaded, and all service classes receive a high-grade service.

Let $t(0)$ define the beginning of the busy period. The arrival curve for class i at the n-th event, $A_i(n)$, is the total traffic that has arrived to the transmission queue of an output link at a router since the beginning of the current busy period, that is

$$A_i(n) = \sum_{k=0}^{n} a_i(k) \ .$$

The input curve, $R_i^{in}(n)$, is the traffic that has been entered into the transmission queue at the n-th event,

$$R_i^{in}(n) = A_i(n) - \sum_{k=0}^{n} l_i(k) \ .$$

The output curve is the traffic that has been transmitted since the beginning of the current busy period, that is

$$R_i^{out}(n) = \sum_{k=0}^{n-1} r_i(k)\Delta t(k) \ . \tag{1}$$

7

In Figure 1, we illustrate the concepts of arrival curve, input curve, and output curve for class-i traffic. At any time $t(n)$, the service rate is the slope of the output curve. In the figure, the service rate is adjusted at times $t(n_1), t(n_2)$ and $t(n)$.

As illustrated in Figure 1, for event n, the vertical and horizontal distance between the input and output curves, respectively, denote the class-i backlog $B_i(n)$ and the class-i delay $D_i(n)$. For the n-th event, we have

$$B_i(n) = R_i^{in}(n) - R_i^{out}(n) \; ,$$

and

$$D_i(n) = t(n) - t\big(\sup\{k < n \mid R_i^{out}(n) \leq R_i^{in}(k)\}\big) \; . \tag{2}$$

Eqn. (2) characterizes the delay of the class-i traffic that departs at the n-th event.

We define the *loss rate* to be the ratio of dropped traffic to the arrivals. That is

$$p_i(n) = \frac{A_i(n) - R_i^{in}(n)}{A_i(n)} \; . \tag{3}$$

Since, from the definition of $A_i(n)$ and $R_i^{in}(n)$, the $p_i(n)$ are computed only over the current busy period, they correspond to long-term loss rates only if busy periods are long. We justify our choice with the observation that traffic is dropped only at times of congestion, i.e., when the link is overloaded, and, hence, when the busy period is long.

We use the above metrics to express best-effort bounds and proportional differentiation of delay, loss, and throughput. A best-effort delay bound on class i for all events n with $B_i(n) > 0$ is specified as

$$D_i(n) \leq d_i \; , \tag{4}$$

where d_i is the desired upper bound on the delay of class i. Similarly, a best-effort loss rate bound for class i is defined by

$$p_i(n) \leq L_i \; . \tag{5}$$

A best-effort throughput bound for class i is specified as

$$r_i(n) \geq \mu_i \; . \tag{6}$$

Proportional differentiation on delay, loss, and throughput, respectively, is defined, for all n such that $B_i(n) > 0$ and $B_{i+1}(n) > 0$, as

$$\frac{D_{i+1}(n)}{D_i(n)} = \alpha_i^{\text{del}} \; , \tag{7}$$

$$\frac{p_{i+1}(n)}{p_i(n)} = \alpha_i^{\text{loss}} \; , \tag{8}$$

and

$$\frac{r_{i+1}(n)}{r_i(n)} = \alpha_i^{\text{tput}} \; , \tag{9}$$

where $\alpha_i^{\text{del}} > 1$, $\alpha_i^{\text{loss}} > 1$, and $\alpha_i^{\text{tput}} > 1$ are constants that quantify the proportional differentiation desired.

We make the following important remarks about the guarantees:

- Without additional assumptions about the per-class backlogs, offering proportional guarantees simultaneously for delay and throughput may result in an infeasible set of service guarantees. As an example, from the relationship between backlog, delay, and throughput of a given class, it is easy to see that "Class-2 delay/class-1 delay = 2" and "Class-2 throughput/class-1 throughput = 2" is feasible only if the backlog of class 2 is four times as large as the backlog of class 1. To avoid such infeasible sets of proportional service guarantees, we require there be at most one proportional guarantee between two classes with consecutive index. So, between class 1 and class 2, there cannot be a both a proportional throughput guarantee and a proportional delay guarantee.

- We note that even if the above constraints on proportional differentiation are respected, a set of proportional service differentiation guarantees could be infeasible under certain traffic conditions, as shown in [35]. Therefore, we allow some slack, generally, a few percent of the current values, in the ratios of loss rates, delays and throughputs to be enforced.

- Since we do not assume admission control or traffic policing, it may not be feasible to enforce all best-effort bounds at all times if the traffic volume in the network is too high. When all best-effort bounds cannot be satisfied, we allow some bounds to be temporarily relaxed according to a specified relaxation order. For instance, the implementation that we discuss in Section 6 adopts a relaxation order that gives loss guarantees priority over delay or rate guarantees, and best-effort bounds priority over proportional differentiation. We emphasize that, while a relaxation order on the service guarantees is needed, the mechanisms we propose in this paper are, unless otherwise noted, independent of the specific relaxation order chosen.

3.2 Rate Allocation and Drop Decisions

We now sketch a solution for realizing the service differentiation specified in Eqs. (4)–(8) at the output link of a router with capacity C and buffer size B. We assume per-class buffering of incoming traffic, and each class is transmitted in a First-Come-First-Served manner. The service rates $r_i(n)$ and the amount of dropped traffic $l_i(n)$ are adjusted at each event n so that the constraints defined by Eqs. (4)–(8) are met. If not all constraints in Eqs. (4)–(8) can be met at the n-th event, then some service differentiation parameters need to be temporarily relaxed. We assume that the order in which differentiation parameters are relaxed is given.

The best-effort delay bound on class i, d_i, imposes a minimum required service rate in the sense that all backlogged class-i traffic at the n-th event will be transmitted within its delay bound d_i if

$$r_i(n) \geq \frac{B_i(n)}{d_i - D_i(n)} \, , \tag{10}$$

for all n. This condition can be verified by inspection of Figure 2. In the figure, a thick line is used to denote the input curve, a thin line represents the output curve, and $t(n)$ is the present time. The delay of the traffic in transmission at $t(n)$ is $D_i(n)$. Because all traffic backlogged at time $t(n)$ arrived in a single burst, the amount of time remaining to transmit the traffic at the tail of the queue within the best-effort delay bound d_i
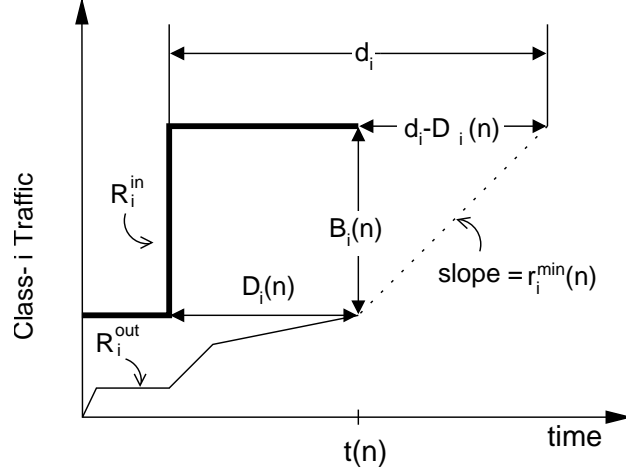
Figure 2: **Determining service rates for delay bounds.**

is given by $d_i - D_i(n)$. Hence, the output curve at time $t(n) + d_i - D_i(n)$ should have at least a value of $R_i^{out}(n) + B_i(n)$ so that all traffic backlogged at $t(n)$ meets its delay bound d_i. So, the minimum service rate, $r_i^{\min}(n)$, required to meet d_i is given by the slope $B_i(n)/(d_i - D_i(n))$.

If the condition of Eqn. (10) holds for any n, the delay bound d_i is never exceeded. If class i has, in addition, a throughput bound μ_i, the expression for the minimum rate needed by class i at the n-th event becomes[2]

$$r_i^{\min}(n) = \sup \left\{ \frac{B_i(n)}{d_i - D_i(n)}, \mu_i \cdot \chi_{B_i(n)>0} \right\} . \tag{11}$$

The service rate that can be allocated to class i is upper bounded by the difference of the output link capacity and the minimum service rates needed by the other classes, that is,

$$r_i^{\max}(n) = C - \sum_{j \neq i} r_j^{\min}(n) .$$

Therefore, the service rate can take any value $r_i(n)$ with

$$r_i^{\min}(n) \leq r_i(n) \leq r_i^{\max}(n) ,$$

subject to the constraint $\sum_i r_i(n) \leq C$. Given this range of feasible values, $r_i(n)$ can be selected to satisfy proportional delay and throughput differentiation.

We view the computation of $r_i(n)$ in terms of the recursion

$$r_i(n) = r_i(n-1) + \Delta r_i(n) , \tag{12}$$

where $\Delta r_i(n)$ is selected such that the constraints of proportional delay and throughput differentiation are satisfied at event n. From Eqs. (1) and (2), the delay $D_i(n)$ at the n-th event is a function of $r_i(k)$ with

---

[2]We define $\chi_{expr} = 1$ if $expr$ is true and $\chi_{expr} = 0$ otherwise.

10

$k < n$. By monitoring $D_i(n)$ we can determine the deviation from the desired proportional differentiation due to past service rate allocations, and infer the adjustment $\Delta r_i(n) = f(D_i(n))$ needed to attenuate this deviation.

If no feasible service rate allocation for realizing all desired delay and throughput differentiation exists at the n-th event, or if there is a buffer overflow at the n-th event, traffic must be dropped, either from a new arrival or from the current backlog. Loss differentiation determines which class(es) suffer(s) traffic drops at the n-th event.

To enforce loss differentiation, we rewrite the loss rate, as a difference equation. We use $A_i(n) = A_i(n-1) + a_i(n)$, and $R_i^{in}(n) = R_i^{in}(n-1) + a_i(n) - l_i(n)$ in Eqn. (3), and obtain

$$p_i(n) = \frac{A_i(n-1) + a_i(n) - (R_i^{in}(n-1) + a_i(n) - l_i(n))}{A_i(n)} ,$$

or, after simplification,

$$p_i(n) = \frac{A_i(n-1) - R_i^{in}(n-1)}{A_i(n)} + \frac{l_i(n)}{A_i(n)} ,$$

which, using Eqn. (3), allows us to express $p_i(n)$ as a function of $p_i(n-1)$:

$$p_i(n) = p_i(n-1)\frac{A_i(n-1)}{A_i(n)} + \frac{l_i(n)}{A_i(n)} . \tag{13}$$

From Eqn. (13), we can determine how the loss rate of class i evolves if traffic is dropped from class i at the n-th event. Thus, we can determine the set of classes that can suffer drops without exceeding best-effort loss bounds. In this set, we choose the class whose loss rate differs by the largest amount from the objective of Eqn. (7).

The recursive expressions for service rates and the loss rates from Eqs. (12) and (13) can be used to characterize the service rate allocation and dropping decisions as feedback control problems. In the next sections, we will describe two feedback problems: one for delay and rate differentiation (delay feedback loop), and one for loss differentiation (loss feedback loop). In Section 6, we describe the interaction of the two feedback problems.

4   The Delay Feedback Loop

In this section, we present feedback loops that enforce the desired delay and rate differentiation given by Eqs. (4), (6), and (7). We have one feedback loop for each class with proportional delay and/or rate differentiation. In the feedback loop for class i, we characterize changes to service rate $\Delta r_i(n)$ by approximating the non-linear effects of the service rate adjustment on the delays by a linear system, and derive a stability condition for the linearized control loop, similar to a technique used in [31, 38, 39, 46]. While the stability condition derived does not ensure that the non-linear control loop converges, the stability condition gives useful guidelines for selecting the configuration parameters of the loop.

An alternative to using a linear approximation of the non-linear system under consideration is to directly apply non-linear control techniques to derive the stability conditions. Non-linear control techniques, e.g., adaptive control [7], resort to algorithms such as gradient estimators. The practicality of a gradient estimator implementation to be executed for each packet arrival is questionable. Furthermore, adaptive control theory is used to dynamically estimate unknown parameters that remain constant over time, whereas all quantities in the feedback loops we are studying vary over time. This implies that some approximations have to be made to use adaptive control theory. The necessary approximations, e.g., assuming that the backlog remains constant over a very short time interval, are similar to the approximations we will use to linearize the feedback loops, so that there is no immediate advantage of using adaptive control in the design of our algorithm.

4.1 Expressing the Objective of Proportional Delay and Rate Differentiation

Let us assume for now that all classes are offered proportional delay differentiation, and that we do not have any proportional throughput differentiation. Later, these assumptions will be relaxed. The set of constraints given by Eqn. (7) leads to the following system of equations:

$$
\begin{aligned}
D_2(n) &= \alpha_1^{\mathrm{del}} \cdot D_1(n) \,, \\
&\vdots \\
D_N(n) &= \left(\prod_{j=1}^{N-1} \alpha_j^{\mathrm{del}} \right) D_1(n) \,.
\end{aligned}
\tag{14}
$$

Let $m_i = \prod_{j=1}^{i-1} \alpha_j^{\mathrm{del}}$ for $i > 1$, and $m_1 = 1$. We define a weighted delay of class i at the n-th event, denoted by $D_i^*(n)$, as

$$
D_i^*(n) = \left(\prod_{k=1,\ k\neq i}^{N} m_k \right) D_i(n) \,.
\tag{15}
$$

The weighted delay $D_i^*(n)$ is the delay of class i at the n-th event, multiplied by a scaling factor expressing the proportional delay differentiation desired. By multiplying each line of Eqn. (14) with $\prod_{j\neq i} m_j$, we see that the desired proportional delay differentiation is achieved for all classes if

$$
\forall i, j \,, \forall n : D_i^*(n) = D_j^*(n) \,.
\tag{16}
$$

Eqn. (16) is equivalent to

$$
\forall i \,, \forall n : D_i^*(n) = \overline{D}^*(n) \,,
$$

where

$$
\overline{D}^*(n) := \frac{1}{N} \sum_i D_i^*(n) \,.
\tag{17}
$$

We set $\overline{D}^*(n)$ to be the *set point* common to all delay feedback loops. The feedback loop for class i reduces the difference $|\overline{D}^* - D_i^*(n)|$ of class i from the common set point $\overline{D}^*(n)$.

When some classes are not offered proportional delay differentiation we extend the above analysis as follows. If proportional delay differentiation is requested for some, but not for all classes, constraints as in Eqn. (14) can be defined for each group of classes with contiguous indices. Then, the feedback loops are constructed independently for each group.

We include proportional throughput differentiation in our analysis as follows. If we assume that no traffic is ever dropped to satisfy proportional delay or rate guarantees, we can express proportional throughput differentiation between two classes in terms of proportional delay differentiation. Indeed, from the relationship between delay, backlog and rate, we have

$$\frac{r_{i+1}(n)}{r_i(n)} = \frac{B_{i+1}(n)}{D_{i+1}(n)} \frac{D_i(n)}{B_i(n)} ,$$

which, from the proportional throughput guarantee defined in Eqn. (9), reduces to

$$\frac{B_{i+1}(n)}{B_i(n)} \frac{D_i(n)}{D_{i+1}(n)} = \alpha_i^{\text{tput}} ,$$

which we can rearrange as

$$\frac{D_{i+1}(n)}{D_i(n)} = \frac{1}{\alpha_i^{\text{tput}}} \frac{B_{i+1}(n)}{B_i(n)} .$$

Recall that we have imposed that no pair of classes can be subject to both proportional delay and throughput differentiation. Thus, we can express the proportional throughput guarantee as a proportional delay guarantee α_i^{del}, with

$$\alpha_i^{\text{del}} = \frac{1}{\alpha_i^{\text{tput}}} \frac{B_{i+1}(n)}{B_i(n)} .$$

In other words, proportional throughput differentiation can be viewed as proportional delay differentiation where the desired ratio of delays α_i^{del} varies over time. Some of the assumptions we will discuss in the stability analysis of the delay feedback loops will allow us to neglect the time-dependency of this ratio over short time intervals such as the current busy period. Thus, for the sake of simplicity, we will only consider proportional delay differentiation in the remainder of this paper, and we will consider that proportional throughput differentiation can always be obtained through proportional delay differentiation.

4.2 Service Rate Adjustment

Next, we determine how to adjust the service rate to achieve the desired delay differentiation. Let $e_i(n)$, referred to as *error*, denote the deviation of the weighted delay of class i from the set point, i.e.,

$$e_i(n) = \overline{D}^*(n) - D_i^*(n) . \tag{18}$$

Note that the sum of the errors is always zero, that is, for all n,

$$\sum_i e_i(n) = N\overline{D}^*(n) - \sum_i D_i^*(n) = 0 .$$

13

If proportional delay differentiation is achieved, we have $e_i(n) = 0$ for all classes. We use the error $e_i(n)$ to compute the service rate adjustment $\Delta r_i(n)$ needed for class i to satisfy the proportional delay differentiation constraints. From Eqn. (18), we note that if $e_i(n) < 0$, $D_i^*(n) > \overline{D}^*(n)$, class i delays are too high with respect to the desired proportional delay differentiation. Therefore, $r_i(n)$ must be increased. Conversely, $e_i(n) > 0$ indicates that class i delays are too low, and $r_i(n)$ must be decreased. Hence, the rate adjustment $\Delta r_i(n)$ is a decreasing function of the error $e_i(n)$, written as $\Delta r_i(n) = f(e_i(n))$, where $f(.)$ is a monotonically decreasing function. We choose

$$\Delta r_i(n) = K(n) \cdot e_i(n) \,, \tag{19}$$

where $K(n) < 0$ is called the controller. An advantage of this controller is that only a single multiplication is needed to obtain the rate adjustment. Another advantage is that, at any n, we have

$$\sum_i \Delta r_i(n) = K(n) \sum_i e_i(n) = 0 \,. \tag{20}$$

From Eqn. (20), the controller imposes a work-conserving system, as long as the initial condition $\sum_i r_i(0) = C$ is satisfied. Note that systems that are not work-conserving, i.e., where the link may be idle even if there is a positive backlog, may be undesirable for networks that need to achieve a high resource utilization.

We next linearize the delay feedback loop to obtain a condition on $K(n)$ to ensure that the delay feedback loops are stable, in the sense that they attenuate the errors $e_i(n)$ over time. We later derive an additional condition on $K(n)$ so that the rate adjustments $\Delta r_i(n)$ do not create a violation of the best-effort delay and throughput bounds.

4.3 Linearization of the Delay Feedback Loop

The non-linearities in the delay feedback loop primarily result from the non-linear relationship between the service rate adjustments Δr_i and the delays D_i. We introduce a set of assumptions needed to linearize the delay feedback loops, before discussing the linearized relationship between Δr_i and D_i.

Assumptions. We use four assumptions, labeled (A1)–(A4), to linearize the control loop.

(A1) Consider a virtual time axis, where the event numbers, n, are equidistant sampling times. We assume that the skew between virtual time and real time can be neglected. Since events are traffic arrivals from any class, the assumption holds when the aggregate traffic arrival rate is almost constant. Over a busy period, if the aggregate arrival rate remains below the link capacity for too long, the queue becomes empty and the busy period ends. So, the assumption is accurate unless the considered output link is constantly overloaded, and further subject to a highly variable load.

(A2) We assume that, for any class i, the delay of class-i traffic does not vary significantly between events n and $(n + 1)$, i.e.,

$$D_i(n + 1) \approx D_i(n).$$

This assumption is accurate when class i remains backlogged between events n and $(n + 1)$, and changes to the service rate r_i between n and $(n + 1)$ remain modest, i.e., $\Delta r(n)$ is relatively small. This assumption may not hold when the time elapsed between the n-th and $(n + 1)$-th event is large, i.e., when the arrival rate of traffic from all classes is low. However, a low aggregate arrival rate generally results in the current busy period ending quickly.

(A3) We assume that the backlog of class-i traffic $B_i(n)$ does not vary significantly over the time $D_i(n)$ spent by class-i traffic in the transmission queue. The assumption is accurate when the delays D_i are small and traffic arrivals are relatively smooth. The assumption is not accurate when traffic arrivals are extremely bursty over very short time intervals.

(A4) We assume that the service rate r_i is not subject to large variations over short intervals of time. The assumption is likely to hold unless the proportional coefficient $K(n)$ is chosen very large. The assumption may not be accurate when the backlog of class-i frequently oscillates between zero and a positive value, because r_i is reset every time class-i is not backlogged.

Clearly, the above assumptions are idealistic, and stability under these assumptions does not guarantee stability of the actual delay feedback loops. However, the numerical data in Section 7 suggests that the loops converge adequately well.

Relating delays D_i to rate adjustments Δr_i. We next describe the effect of the rate adjustment Δr_i on the delay D_i under (A1)–(A4). To that effect, we relate Δr_i to the average rate $\overline{r_i}$ experienced by the class-i traffic over the time this class-i traffic was backlogged. Then, we relate $\overline{r_i}$ to D_i.

Let us define $\tau_i(n)$ as:
$$D_i(n) = t(n) - t(n - \tau_i(n)) .$$

In other words, $\tau_i(n)$ denotes the number of events that occurred over the time interval during which the class-i traffic leaving at $t(n)$ was backlogged. From (A1) and (A2), we can write

$$\tau_i(n) \approx \tau_i(n + 1) ,$$

and will, from now on, use τ_i to refer to both $\tau_i(n)$ and $\tau_i(n + 1)$.

We relate $\overline{r_i}$ to B_i and D_i as follows. By definition of τ_i and D_i, traffic leaving at $t(n)$ entered the queue at time $t(n - \tau_i)$ and spent $D_i(n)$ in the queue. Thus, the average service rate $\overline{r_i}(n)$ received by the traffic leaving at $t(n)$ is given by:

$$\overline{r_i}(n) = \frac{B_i(n - \tau_i)}{D_i(n)} . \tag{21}$$

From Eqn. (21), $\overline{r_i}(n)$ is the average class-i service rate averaged over $[t(n - \tau_i), t(n))$, whereas, by definition, $r_i(n)$ denotes the class-i service rate over $[t(n), t(n+1))$. We use this observation and (A1) to express $\overline{r_i}$ as a function of r_i, as follows:

$$\overline{r_i}(n + 1) = \frac{(\tau_i - 1)\overline{r_i}(n) + r_i(n)}{\tau_i} . \tag{22}$$

15

Let us now define

$$\Delta\overline{r_i}(n+1) = \overline{r_i}(n+1) - \overline{r_i}(n) \ . \tag{23}$$

Combining Eqs. (22) and (23), we get

$$\Delta\overline{r_i}(n+1) = \frac{(\tau_i - 1)\Delta\overline{r_i}(n) + \Delta r_i(n)}{\tau_i} \ . \tag{24}$$

Eqn. (24) describes the relationship between a change in the service rate and a change in the average rate.

We now derive the relationship between $\Delta\overline{r_i}(n)$ and a change in the delay of class i, denoted as $\Delta D_i(n)$, and defined by

$$\Delta D_i(n+1) = D_i(n+1) - D_i(n) \ .$$

Since we have, from Eqn. (21),

$$D_i(n) = \frac{B_i(n - \tau_i)}{\overline{r_i}(n)} \ ,$$

and

$$D_i(n+1) = \frac{B_i(n+1-\tau_i)}{\overline{r_i}(n+1)} \ ,$$

we get

$$\Delta D_i(n+1) = \frac{B_i(n+1-\tau_i)}{\overline{r_i}(n+1)} - \frac{B_i(n - \tau_i)}{\overline{r_i}(n)} \ . \tag{25}$$

Eqn. (25) is not linear in $\overline{r_i}$. We use (A4) to linearize Eqn. (25), by means of a first order Taylor series expansion. (A4) implies that

$$\Delta\overline{r_i}(n+1) \ll \overline{r_i}(n) \ ,$$

which, using $B_i(n+1-\tau_i) \approx B_i(n-\tau_i)$ obtained from (A3), allows to rewrite Eqn. (25) as

$$\Delta D_i(n+1) = -\frac{B_i(n - \tau_i)}{\overline{r_i}^2(n)}\Delta\overline{r_i}(n+1) + \omega_i(n) \ , \tag{26}$$

where $\omega_i(n)$ is the error in the evaluation of $\Delta D_i(n+1)$ resulting from (A1)–(A4). Then, the relationship between delay variations and the delay is given by

$$D_i(n+1) = \sum_{k=0}^{n+1}\Delta D_i(k) \ , \tag{27}$$

$D_i(n+1)$ is used to compute $D_i^*(n+1)$, using Eqn. (15). Finally, from Eqs. (17) and (18), the error at the $(n+1)$-th event, $e_i(n+1)$, is obtained from $D_i^*(n+1)$. This completes the description of the linearized delay feedback loop. We now turn to the derivation of a stability condition on the linearized delay feedback loop.

4.4 Stability Condition on the Linearized Delay Feedback Loop

We derive the stability condition of the linearized delay feedback loop using a two-step process. We first express the delay feedback loop in the frequency domain, using z-transforms, and then apply a standard stability argument to the frequency-domain expression of feedback loop to obtain bounds on $K(n)$ that ensure stability of the linearized feedback loop.
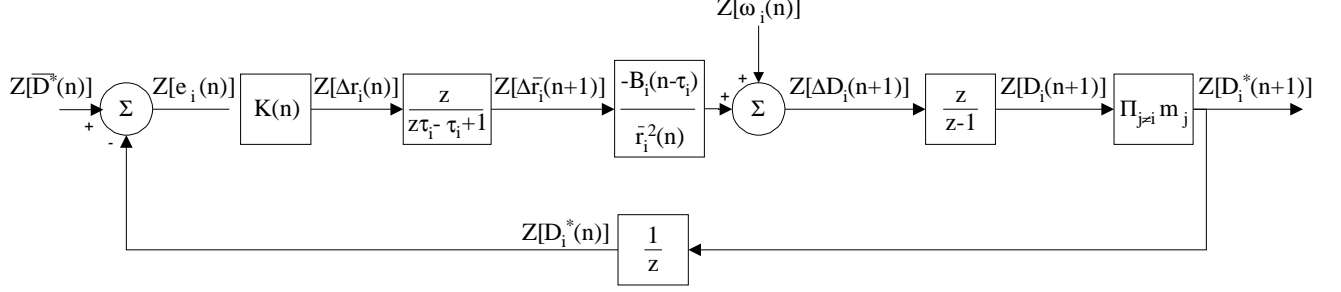
Figure 3: **The class-i delay feedback loop.** This model uses z-transforms of the relationships derived in Section 4.2.

Frequency-domain expression for the feedback loop. We express the delay feedback loop in the frequency domain using using z-transforms of Eqs. (15)–(27). We denote the z-transform of a function $f(n)$ by $Z[f(n)]$, defined as

$$Z[f(n)] = \sum_{n=0}^{+\infty} f(n)z^{-n} .$$

Eqs. (15), (18), (19), (26) are unchanged when using z-transforms. Eqn. (24) yields

$$Z[\Delta\overline{r_i}(n+1)] = (\tau_i - 1) \cdot \frac{Z[\Delta\overline{r_i}(n)]}{\tau_i} + \frac{Z[\Delta r_i(n)]}{\tau_i} ,$$

which, using the property that, for any continuous function f, $Z[f(n)] = \frac{1}{z}Z[f(n+1)]$, implies that

$$Z[\Delta\overline{r_i}(n+1)] = (\tau_i - 1) \cdot \frac{Z[\Delta\overline{r_i}(n+1)]}{z\tau_i} + \frac{Z[\Delta r_i(n)]}{\tau_i} .$$

By reordering terms we obtain

$$Z[\Delta\overline{r_i}(n+1)] = \frac{z}{z\tau_i - \tau_i + 1}Z[\Delta r_i(n)] .$$

The z-transform of Eqn. (27) is

$$Z[D_i(n+1)] = \frac{z}{z-1}Z[\Delta D_i(n+1)] .$$

Also, the relationship between $D_i^*(n)$ and $D_i^*(n+1)$ in the frequency domain is given by

$$Z[D_i^*(n)] = \frac{1}{z}Z[D_i^*(n+1)] .$$

Figure 3 illustrates the frequency-domain expression of the delay feedback loop. In the figure, each block maps an input variable to an output variable by multiplying the input variable by the contents of the block. For instance, the first block maps $Z[e_i(n)]$ to $Z[\Delta r_i(n)]$ by multiplying $Z[e_i(n)]$ by $K(n)$. The product of all individual blocks is called the *loop gain*.

We notice that in the class-i delay feedback loop of Figure 3, some quantities (e.g., τ_i, B_i, $\overline{r_i}$) are time-dependent. Therefore, the loop gain is time-dependent. Classical linear control theory [28], on the other hand, generally requires the loop gain to be time invariant to obtain stability conditions. However, stability can still be achieved with a time-dependent loop gain, if the loop gain is not increasing unboundedly over time [7].

Stability analysis We obtain stability bounds on $K(n)$ from a standard control theory result [28]. Denoting the loop gain by $G(z)$, the loop is stable if and only if the roots of the characteristic equation $1+G(z) = 0$ have a module less than one. We obtain an expression for $G(z)$ by taking the product of all blocks in Figure 3,

$$G(z) = -\frac{1}{z}\frac{z}{z-1}\frac{\left(\prod_{j\neq i} m_j\right) B_i(n - \tau_i)K(n)}{\overline{r_i}^2(n)}\frac{z}{z\tau_i - \tau_i + 1} \ .$$

The negative sign comes from the expression of $e_i(n)$, where $D_i^*(n)$ is subtracted from $\overline{D}^*(n)$. We use (A4) to further simplify the expression for $G(z)$. Under (A4), $\Delta\overline{r_i}(n+1) \approx \Delta r_i(n)$, which enables us to approximate the gain of the block $z/(z\tau_i + 1 - \tau_i)$ by 1. With this approximation, we obtain a new loop gain $G'(z)$ as follows

$$G'(z) = -\frac{1}{z}\frac{z}{z-1}\frac{\left(\prod_{j\neq i} m_j\right) B_i(n - \tau_i)K(n)}{\overline{r_i}^2(n)} \ .$$

The characteristic equation of the approximate system is

$$1 - \frac{1}{z-1}\frac{\left(\prod_{j\neq i} m_j\right) B_i(n - \tau_i)K(n)}{\overline{r_i}^2(n)} = 0 \ ,$$

which has exactly one root,

$$\hat{z} = 1 + \frac{\left(\prod_{j\neq i} m_j\right) B_i(n - \tau_i)K(n)}{\overline{r_i}^2(n)} \ .$$

With the root $\hat{z}$, we obtain the following stability condition

$$\left| 1 + \frac{\left(\prod_{j\neq i} m_j\right) B_i(n - \tau_i)K(n)}{\overline{r_i}^2(n)} \right| \leq 1 \ ,$$

or, equivalently,

$$1 + \frac{\left(\prod_{j\neq i} m_j\right) B_i(n - \tau_i)K(n)}{\overline{r_i}^2(n)} \geq -1 \tag{28}$$

$$1 + \frac{\left(\prod_{j\neq i} m_j\right) B_i(n - \tau_i)K(n)}{\overline{r_i}^2(n)} \leq 1 \ . \tag{29}$$

All quantities in Eqn. (29), with the exception of $K(n)$, are positive. Hence, the condition described by Eqn. (29) reduces to $K(n) \leq 0$.

The condition in Eqn. (28) becomes, after reordering,

$$K(n) \geq -2 \cdot \frac{\overline{r_i}^2(n)}{\left(\prod_{j\neq i} m_j\right) B_i(n - \tau_i)} \ . \tag{30}$$

Since, with Eqn. (21), we can write

$$\frac{\overline{r_i}^2(n)}{B_i(n-\tau_i)} = \frac{B_i(n-\tau_i)}{D_i(n)^2} ,$$

Eqn. (30) can be expressed as

$$K(n) \geq -2 \cdot \frac{B_i(n-\tau_i)}{\left(\prod_{j\neq i} m_j\right) D_i^2(n)} . \tag{31}$$

The condition given by Eqn. (31) requires to keep a history of the backlogs. The need to maintain a backlog history can be alleviated, using (A2) and writing $B_i(n-\tau_i) \approx B_i(n)$, which allows us to simplify Eqn. (31). Combining with $K(n) \leq 0$, we obtain the following expression for the stability condition for the class-i delay feedback loop:

$$-2 \cdot \frac{B_i(n)}{\prod_{j\neq i} m_j \cdot D_i^2(n)} \leq K(n) \leq 0 .$$

Since $K(n)$ must be common to all classes for Eqn. (20) to hold, we finally get

$$-2 \cdot \min_i \left\{ \frac{B_i(n)}{\prod_{j\neq i} m_j \cdot D_i^2(n)} \right\} \leq K(n) \leq 0 . \tag{32}$$

The condition in Eqn. (32) ensures that the linearized delay feedback loops will not engage in divergent oscillations. We cannot be certain that the assumptions made to linearize the delay feedback loops hold in practice, and cannot claim that Eqn. (32) ensures stability of the (non-linear) delay feedback loops. However, we can use Eqn. (32) as a design guideline for $K(n)$.

4.5   Including the Absolute Delay and Rate Constraints

The condition on $K(n)$ we obtained in Eqn. (32) is needed to enforce proportional differentiation. So far, we have not considered the best-effort delay and rate bounds in the construction of the delay feedback loops. These best-effort delay and rate bounds are viewed as a saturation constraint on the rate adjustment, and yield a second condition on $K(n)$. To satisfy the constraints $r_i(n) \geq r_i^{\min}(n)$, we may need to clip $\Delta r_i(n)$ when the new rate is below the minimum. This, however, may violate the work-conserving property resulting from Eqn. (20). To respect the saturation constraint, $K(n)$ has to satisfy

$$r_i(n-1) + K(n)e_i(n) \geq r_i^{\min}(n) ,$$

and apply that $K(n)$ to all control loops. The above implies that we must have

$$K(n) \geq \max_i \left(\frac{r_i^{\min}(n) - r_i(n-1)}{e_i(n)} \right) . \tag{33}$$

We note that if

$$\max_i \left(\frac{r_i^{\min}(n) - r_i(n-1)}{e_i(n)} \right) > 0 ,$$

19

we cannot satisfy both Eqn. (33) and $K(n) \leq 0$, required by Eqn. (32). In other words, we cannot satisfy best-effort delay and throughput bounds and proportional delay differentiation at the same time. In such a case, we relax either Eqn. (32) or Eqn. (33) according to a given relaxation order. For instance, giving best-effort bounds higher precedence than proportional differentiation results in relaxing Eqn. (32) and satisfying Eqn. (33).

5 The Loss Feedback Loop

We now describe the feedback loop which controls the traffic dropped from class i to satisfy proportional loss differentiation within the limits imposed by the best-effort loss bounds. As before, we first assume that all classes are offered proportional loss differentiation. We relax this assumption in the same manner as we relaxed the assumption that all classes are offered proportional delay differentiation in Section 4.1.

Traffic must be dropped at the n-th event either if there is a buffer overflow or if best-effort delay bounds cannot be satisfied given the current backlog. For any class k, we can express the class-k backlog at the n-th event, $B_k(n)$, in function of the arrivals $a_k(n)$, the losses $l_k(n)$ and the service rate $r_k(n)$ as

$$B_k(n) = B_k(n-1) + a_k(n) - l_k(n) - \Delta t(n-1) r_k(n) . \tag{34}$$

With a buffer size B, to prevent buffer overflows at the n-th event, we need $\sum_k B_k(n) \leq B$, which, using Eqn. (34) and the work-conserving property $\sum_k r_k(n) = C$, becomes

$$\sum_{k=1}^{N} \big(B_k(n-1) + a_k(n) - l_k(n) \big) - \Delta t(n-1) C \leq B . \tag{35}$$

We must ensure that $\sum_k r_k^{\min}(n) \leq C$ to be able to provide delay and throughput bounds. Using the definition of $r_k^{\min}(n)$ given by Eqn. (11), and Eqn. (34), we obtain the following condition:

$$\sum_{k=1}^{N} \max \left\{ \frac{B_k(n-1) - r_k(n-1)\Delta t(n-1) + a_k(n) - l_k(n)}{d_k - D_k(n)} , \mu_k \cdot \chi_{B_k(n)>0} \right\} \leq C . \tag{36}$$

If either of Eqs. (35) or (36) is violated, traffic is dropped to enforce proportional loss differentiation. To describe how proportional loss differentiation is enforced, let us define a weighted loss rate as

$$p_i^*(n) = \left(\prod_{j=1, \, j \neq i}^{N} m_j' \right) p_i(n) ,$$

where $m_i' = \prod_{j=1}^{i-1} \alpha_j^{\text{loss}}$ for $i > 1$ and $m_1' = 1$. With this definition, Eqn. (8) reduces to

$$\forall (i,j) , \forall n : p_i^*(n) = p_j^*(n) ,$$

which is equivalent to

$$\forall i , \forall n : p_i^*(n) = \bar{p}^*(n) ,$$

where

$$\bar{p}^*(n) = \frac{1}{N} \sum_i p_i^*(n) \,.$$

We set $\bar{p}^*(n)$, as the set point for the loss feedback loop, and use $\bar{p}^*(n)$ to compute an error

$$e_i'(n) = \bar{p}^*(n) - p_i^*(n) \,.$$

The desired proportional loss differentiation is achieved when $e_i'(n) = 0$ for all i. The loss feedback loops decrease the errors $e_i'(n)$ by increasing $p_i^*(n)$ for classes that have $e_i'(n) > 0$ as follows. Let $\langle i_1, i_2, \ldots, i_R \rangle$ be an ordering of the class indices from all backlogged classes, that is, $B_{i_k}(n) > 0$ for $1 \leq k \leq R$, such that $e_{i_s}'(n) \geq e_{i_r}'(n)$ if $i_s < i_r$. Traffic is dropped in the order of $\langle i_1, i_2, \ldots, i_R \rangle$.

Best-effort loss rate bounds impose an upper bound, $l_i^*(n)$, on the traffic that can be dropped at event n from class i. The value of $l_i^*(n)$ is determined from Eqs. (5) and (13) as

$$l_i^*(n) = A_i(n)L_i - p_i(n-1)A_i(n-1) \,.$$

If the conditions in Eqs. (35) and (36) are violated, traffic is dropped from class i_1 until the conditions are satisfied, or until the maximum amount of traffic $l_{i_1}^*(n)$ has been dropped. Then traffic is dropped from class i_2, and so forth. Suppose that the conditions in Eqs. (35) and (36) are satisfied for the first time if $l_j^*(n)$ traffic is dropped from classes $j = i_1, i_2, \ldots, i_{\hat{k}-1}$, and $\hat{x}(n) \leq l_{\hat{k}}^*(n)$ traffic is dropped from class $i_{\hat{k}}$, then we obtain:

$$l_i(n) = \begin{cases} l_i^*(n) & \text{if } i = i_1, i_2, \ldots, i_{\hat{k}-1} \,, \\ \hat{x}(n) & \text{if } i = i_{\hat{k}} \,, \\ 0 & \text{otherwise} \,. \end{cases} \tag{37}$$

If $l_k(n) = l_k^*(n)$ for all $k = i_1, i_2, \ldots, i_R$, we have the choice between dropping more traffic, thereby relaxing a best-effort loss bound, or ignoring condition (36), thereby relaxing a best-effort delay or rate bound. A predetermined precedence order is used to choose which bound is relaxed. For instance, in the implementation discussed in Section 6, loss bounds take precedence over delay and rate bounds, and condition (36) is relaxed.

The loss feedback loop never increases the maximum error $e_i'(n)$, if $e_i'(n) > 0$, and more than one class is backlogged. Thus, the errors remain bounded and the algorithm presented will not engage in divergent oscillations around the target value $\bar{p}^*(n)$. Additionally, the loss feedback loop and the delay feedback loops are independent of each other. Indeed, since we always drop traffic from the tail of each per-class buffer, losses do not have any effect on the delays of traffic admitted into the transmission queue.

6 Implementation

We implemented the feedback loops presented in Sections 4 and 5 in PC-routers running the FreeBSD 4.3 [3] operating system, using the `altq-3.0` package [12]. `altq` allows programmers to modify the operations of the transmission queue in the IP layer of the FreeBSD kernel. Our implementation [1] has

been included in `altq-3.1`, and more recently in KAME [4]. Inclusion in the base BSD distributions is under consideration. For a detailed discussion of the implementation issues, we refer the reader to [14]. In this section, we only discuss the operations performed when a packet is entered into the transmission queue of an IP router (packet enqueuing) and when a packet is selected for transmission (packet dequeuing), and then argue how the implementation can be deployed.

We use the DiffServ codepoint (DSCP, [43]) in the header of a packet to identify the class index of an IP packet. The DSCP field is set by the edge router; in our implementation, this is the first router traversed by a packet. We chose the following precedence order for relaxing constraints: Best-effort loss rate bounds have higher precedence than delay and throughput bounds, which have in turn higher precedence than proportional differentiation.

6.1   Packet Enqueuing

The `enqueue` procedure are the operations executed in the IP layer when a packet is entered into the transmission queue of an output link. Since, in FreeBSD 4.x, the FreeBSD kernel is single-threaded, the execution of the `enqueue` procedure is strictly sequential.

The `enqueue` procedure performs the dropping decisions and the service rate allocation. We avoid floating point operations in the kernel of the operating system, by expressing delays as machine clock cycles, service rates as bytes per clock cycle (multiplied by a scaling factor of 2^{32}), and loss rates as fractions of 2^{32}. Then, 64-bit (unsigned) integers provide a sufficient degree of accuracy.

In our modified `enqueue` procedure, the transmission queue of an output link has one FIFO queue for each class, implemented as a linked list. We limit the total number of packets that can be queued to $B = 200$. Whenever a packet is entered into the FIFO queue of its class, the arrival time of the packet is recorded, and the waiting times of the packets at the head of each FIFO queue are updated.

The `enqueue` procedure uses the loss feedback loop described in Section 5 to determine if and how much traffic needs to be dropped from each class. In our implementation, the algorithm of Section 5 is run twice. The first time, buffer overflows are resolved by ignoring condition (36); The second time, potential violations of delay and throughput bounds are resolved by ignoring condition (35).

Next, the `enqueue` procedure computes new values for $r_i^{\min}(n)$ from Eqn. (11), and determines new service rates, using Eqs. (12) and (19), with the constraints on $K(n)$ given in Eqs. (32) and (33). If no feasible value for $K(n)$ exists, Eqn. (32) is ignored, thereby giving delay bounds precedence over proportional delay (and throughput) differentiation.

6.2   Packet Dequeuing

The `dequeue` procedure selects one packet from the backlog for transmission. In our implementation, `dequeue` selects one of the traffic classes, and picks the packet at the head of the FIFO queue for this class.

The `dequeue` procedure uses a rate-based scheduling algorithm to adapt the transmission rates $r_i(n)$ from a fluid-flow view to a packet-level environment. Such an adaptation can be performed using well-

known rate-based scheduling algorithm techniques, e.g., Virtual Clock [54] or PGPS [45]. These scheduling algorithms translate a rate allocation, into virtual deadlines of individual packets. In our implementation, we use a modified Deficit Round Robin [49] scheduling algorithm. Let $Xmit_i(n)$ denote the number of bytes of class-i traffic that have been transmitted in the current busy period, the scheduler selects a packet from class i for transmission if

$$i = \arg \max_k \left\{ R_k^{out}(n) - Xmit_k(n) \right\} \ .$$

In other words, the `dequeue` procedure selects the class whose service is the most behind its allocated service rate.

6.3 Service Deployment

We conclude this section with a brief discussion how our proposed mechanisms can be deployed. We remark that service differentiation is only needed at potential bottleneck points. Indeed, if a link is never congested, incoming packets can be transmitted at once, using a First-Come-First-Served queuing policy, thereby getting a high-grade service (no loss, low delay, high throughput). Reports on the utilization of backbone links indicate that backbone links are used at about 60% of their capacity on average [53], which tends to show that the cores of the networks are generally underloaded. We can identify three locations where a bottleneck can occur: at the ingress or egress of an autonomous system or at peering points between two autonomous systems. Placing routers supporting the mechanisms discussed in this paper to regulate traffic at the bottleneck links can ensure that service differentiation is provided where it is needed, without generating any overhead for links that are mostly idle. Increasing the number of routers that implement our adaptive mechanisms never degrades the overall service differentiation observed in the entire network. Routers can be independently configured and can support completely different sets of service guarantees. (Of course, an inconsistent configuration at routers may result in a poor service and resource usage.)

Used in conjunction with routing mechanisms that can perform route-pinning, such as MPLS [47], our adaptive rate allocation and buffer management mechanisms can be used as a building block for end-to-end service differentiation. Based on the per-hop service guarantees, applications can infer end-to-end service differentiation, and select the most appropriate route given their service demands. A thorough inspection of the interaction of traffic engineering techniques (e.g., routing) with the per-hop service proposed here, to provide end-to-end service differentiation, is outside the scope of this paper. We refer the reader to [23] and [34] for a discussion of the mapping of per-hop differentiation as described in this paper to end-to-end service differentiation of loss and delay.

7 Evaluation

In this section, we first assess the stability of the feedback loops by conducting a simulation experiment on a single bottleneck link subject to a highly variable offered load. Then, we present experimental measurements
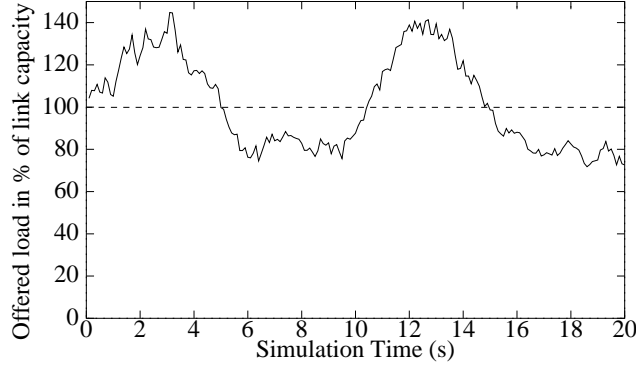
Figure 4: **Experiment 1: Offered Load.** The graph describes the offered load at the simulated 1 Gbps bottleneck link.

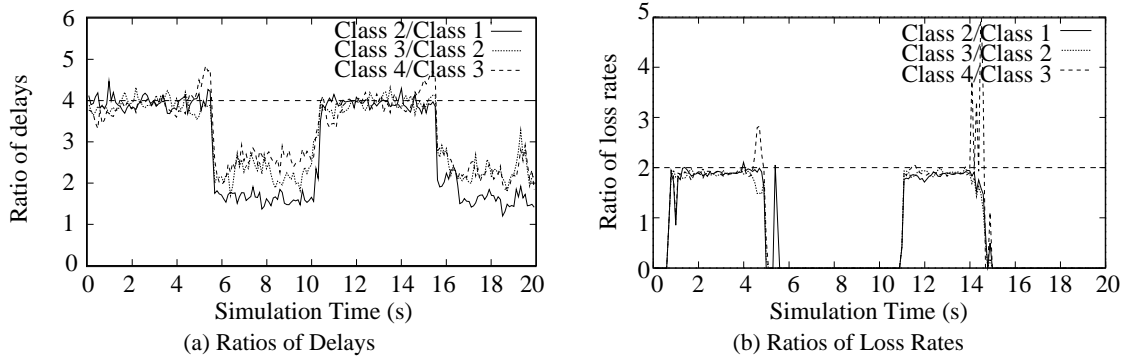|	
(a) Ratios of Delays	(b) Ratios of Loss Rates

Figure 5: **Experiment 1.** The graphs show the proportional differentiation obtained by each class at the simulated 1 Gbps bottleneck link.

of our FreeBSD implementation in a testbed of PC-routers. Last, we present an evaluation of the overhead associated to our proposed algorithms.

7.1 Experiment 1: Simulating a Single Link Topology with Highly Variable Load

To illustrate the stability of the feedback loops, we present a simulation experiment, using the simulator described in [2], with a single bottleneck link of capacity 1 Gbps. The offered load at the bottleneck link is represented in Figure 4, and is obtained by overlapping between 300 and 550 Pareto sources with a shape parameter $\alpha = 1.9$. The number of sources active at a given time is given by a sinusoidal function, which makes the offered load vary between underload and overload on timescales in the order of several seconds. Traffic is self-similar, due to the Pareto distributions used to determine the interarrival times of packets at each source. This accounts for the transient variations in the offered load on timescales in the order of 0.1 seconds. We consider four traffic classes, each contributing approximately a quarter of the offered load.
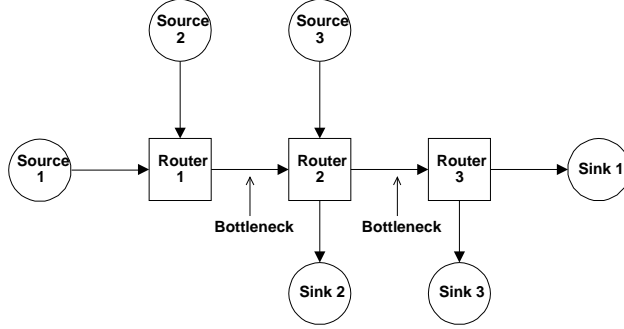
24

Figure 6: **Experiment 2: Network Topology.** All links have a capacity of 100 Mbps. We measure the service provided by Router 1 and 2 at the indicated bottleneck links.

In this experiment, we are interested in assessing the stability of the feedback loops subject to rapid variations of the offered load. In an effort to eliminate potential side effects, such as, for instance, TCP congestion control algorithms, we only use UDP sources, which start transmitting at time $t = 0$, for 20 seconds of simulated time. Also, because best-effort bounds are enforced by saturation constraints external to the feedback loops, we only use proportional differentiation in this first experiment. The objective is to enforce proportional delay differentiation with ratios $\delta_i^{\text{delay}} = 4$, and proportional loss differentiation with ratios $\delta_i^{\text{loss}} = 2$ for $i \in \{1, 2, 3\}$.

We plot the ratios of delays and loss rates of classes in Figure 5. Each datapoint corresponds to a moving average with a sliding window of size 0.1 seconds. The results obtained for the ratios of delays in Figure 5(a) show that proportional delay differentiation is achieved with good accuracy when the link is overloaded. Furthermore, the plots show that the feedback-based controller reacts immediately when the offered load goes from underload to overload, and reacts swiftly (between 0 and 0.2 seconds depending on the class concerned) when the link goes from overload to underload. This indicates that the delay feedback loops used in the closed-loop algorithm are stable. Proportional delay differentiation does not match the target proportional factors $\delta_i^{\text{delay}} = 4$ when the link is underloaded, due to the work-conserving property, which prevents our algorithms from artificially generating delays when the load is small.

Results for ratios of loss rates in Figure 5(b) indicate that the desired proportional loss differentiation is achieved when drops are needed due to an overload. The transient spikes during a transition between an overloaded and an underloaded system are caused by the low number of packets dropped when the link becomes underloaded, which makes the ratios of loss rates become less meaningful.

7.2   Experiment 2: Measurements over Multiple Bottleneck Links with Near-Constant Load

We next evaluate our implementation in PC-routers as described in Section 6. The objective here is to demonstrate that the feedback loops described in this paper can be implemented at relatively high-speeds, and achieve good service differentiation when a mix of best-effort bounds and proportional differentiation

Class	Service Guarantees				
	d_i	L_i	μ_i	δ_i^{delay}	δ_i^{loss}
1	8 milliseconds	1 %	–	–	–
2	–	–	35 Mbps	2	2
3	–	–	–	2	2
4	–	–	–	N/A	N/A

Table 1: **Experiment 2: Desired service differentiation.** Both routers have the same differentiation parameters.

Class	No. of flows	Type	
		Protocol	Traffic
1	6	UDP	On-off
2	6	TCP	Greedy
3	6	TCP	Greedy
4	6	TCP	Greedy

Table 2: **Experiment 2: Traffic mix.** The traffic mix is identical for each source-sink pair. The on-off UDP sources send bursts of 20 packets during an on-period, and have a 150-millisecond off-period. All TCP sources are greedy, i.e., they always have data to transmit, and run the *NewReno* congestion control algorithm.

is desired, and TCP and UDP traffic are competing at multiple bottlenecks. The PCs are Dell PowerEdge 1550 with 1 GHz Intel Pentium-III processors and 256 MB of RAM. The system software is FreeBSD 4.3 and `altq-3.0`. Each system is equipped with five 100 Mbps-Ethernet interfaces.

We use a local network topology using point-to-point Ethernet links as shown in Figure 6. All links are full-duplex and have a capacity of $C = 100$ Mbps. Three PCs are set up as routers, indicated in Figure 6 as Router 1, 2 and 3. Other PCs are acting as sources and sinks of traffic. The topology has two bottlenecks: the link between Routers 1 and 2, and the link between Routers 2 and 3. The buffer size at the output link of each router is set to $B = 200$ packets.

We consider four traffic classes and we provide the service differentiation described in Table 1. The traffic mix, the number of flows per class, and the characterization of the flows for each source is as shown in Table 2. Class 1 traffic consists of on-off UDP flows, and the other classes consist of greedy TCP flows. All sources start transmitting packets with a fixed size of 1024 bytes at time $t = 0$ until the end of the experiments at $t = 60$ seconds.

Sources 1, 2 and 3 send traffic to Sinks 1, 2 and 3, respectively. The traffic mix, the number of flows per class, and the characterization of the flows, is identical for each source, and as shown in Table 2. Each source transmits six flows from each of the classes. Class 1 traffic consists of on-off UDP flows, and the other classes consist of greedy TCP flows. All sources start transmitting packets with a fixed size of 1024 Bytes at time $t = 0$ until the end of the experiments at $t = 60$ seconds. Traffic is generated using the *netperf*
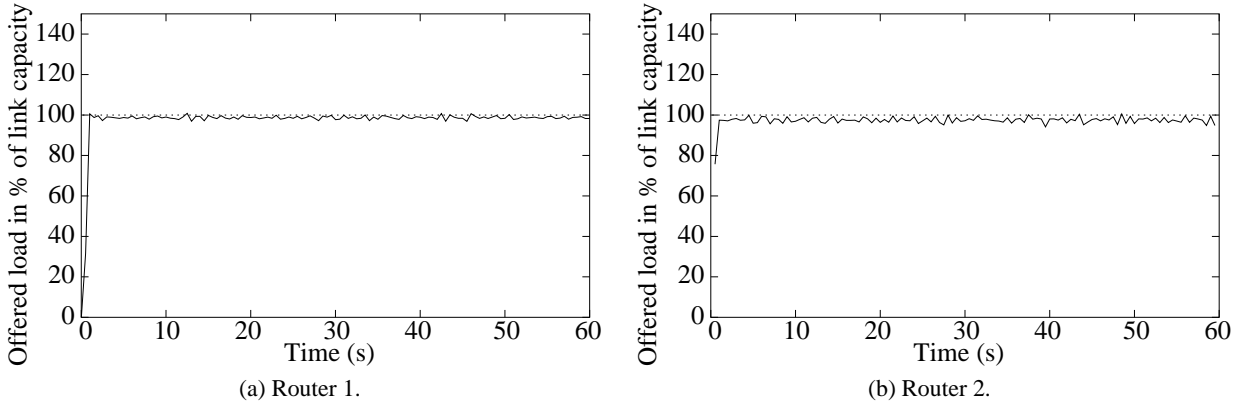
(a) Router 1.
(b) Router 2.

Figure 7: **Experiment 2: Total Load.** The graphs show the offered load at Routers 1 and 2.

v2.1pl3 tool [33]. The network load overloads the bottleneck links of Figure 6. Congestion control at the TCP sources maintains the total load at a level of about 99% of the link capacity, as shown in Figure 7.
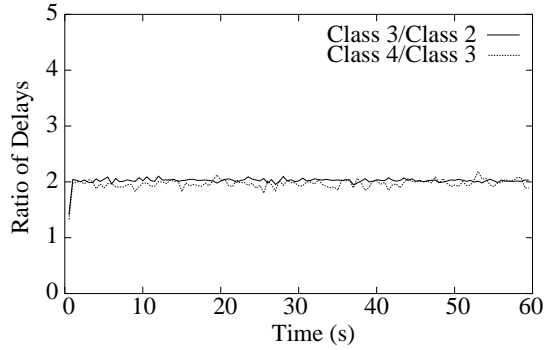
In Figures 8 and 9, we present our measurements of the service received at the bottleneck links of Routers 1 and 2, respectively. All datapoints correspond to moving averages over sliding windows of size 0.5 seconds, except in Figures 8(b) and 9(b), which present the delays of each class-1 packet.

Figures 8(a) and 9(a) depict the ratios of the delays of Classes 4 and 3, and the delays of Classes 3 and 2. The plots show that the target value of $k = 2$ (from Table 1) is achieved.
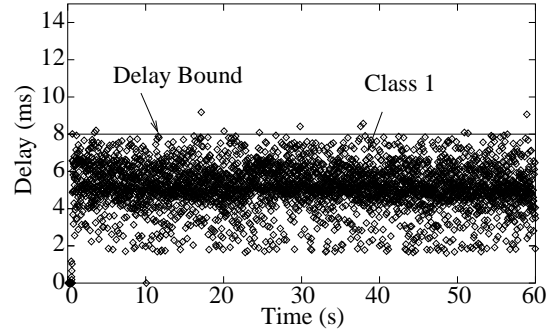
In Figures 8(b) and 9(b) we show the delay of class-1 packets at Router 1 and Router 2. The best-effort delay bound of $d_1 = 8$ milliseconds is satisfied for most ($> 98.5\%$) of the packets, despite the precedence order we chose for our best-effort bounds, i.e, in case of an infeasible set of differentiation constraints, delay bounds are relaxed in favor of loss rate bounds. No class-1 packet ever experiences a delay higher than 10 milliseconds at either Router 1 or 2. Figures 8(c) and 9(c) indicate that delay values of other classes are in the range 10-50 milliseconds.

In Figures 8(c) and (d), and Figures 9(c) and (d), we show the measurements of the loss rates. Figures 8(c) and 9(c) depict the ratios of loss rates for classes 4 and 3, and for classes 3 and 2. The desired ratios of $\delta_2^{\text{loss}} = \delta_3^{\text{loss}} = 2$ are maintained most of the time. As Figures 8(d) and 9(d) indicate, the bound on the loss rates for class 1 of $L_1 = 1$ % is always kept. We also see that the loss rate of class 1 may be higher than the loss rate of other classes, because class 1 is not tied to other classes by proportional guarantees. Our implementation always drops first from class 1, until the loss bound L_1 as been reached, before dropping to satisfy proportional loss guarantees. Note that much less traffic is dropped at Router 2, because Router 2 receives traffic from Source 3 and Router 1, instead of receiving traffic from two sources. Half of the traffic arriving at Router 2 has already been policed by Router 1, resulting in a lower loss rate. Finally, in Figures 8(f) and 9(f) we include the throughput measurements of all classes. We observe that the lower bound on the throughput of Class 2 of $\mu_2 = 35$ Mbps is maintained. The total throughput of all classes, labeled in Figures 8(f) and 9(f) as "Total", is close to the link capacity of 100 Mbps at each router.
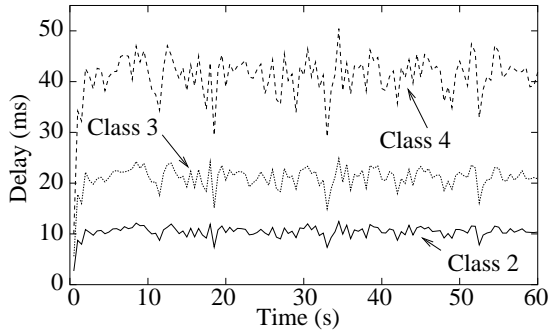
In summary, measurements obtained from a testbed network with multiple bottlenecks indicate that our

Figure 8: **Experiment 2: Router 1.** The graphs show the service obtained by each class at the output link of Router 1.
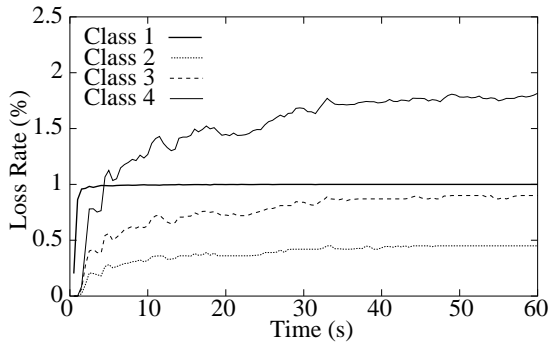
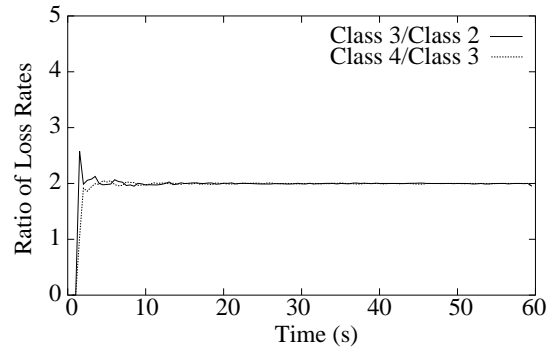(a) Ratios of Delays.

(b) Class-1 Delays (individual).
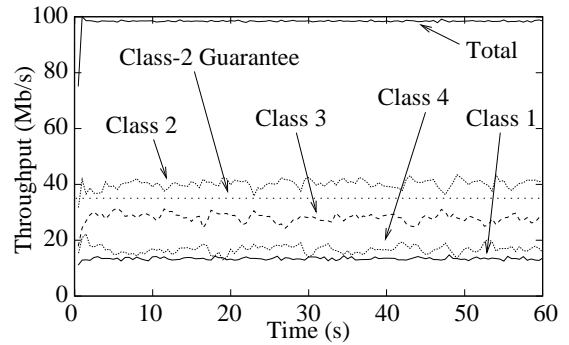
(c) Classes 2, 3 and 4 Delays

(averaged over a sliding window of 0.5 s)

(d) Ratios of Loss Rates.

(e) Loss Rates.

(f) Throughput.

Figure 9: **Experiment 2: Router 2.** The graphs show the service obtained by each class at the output link of Router 2.
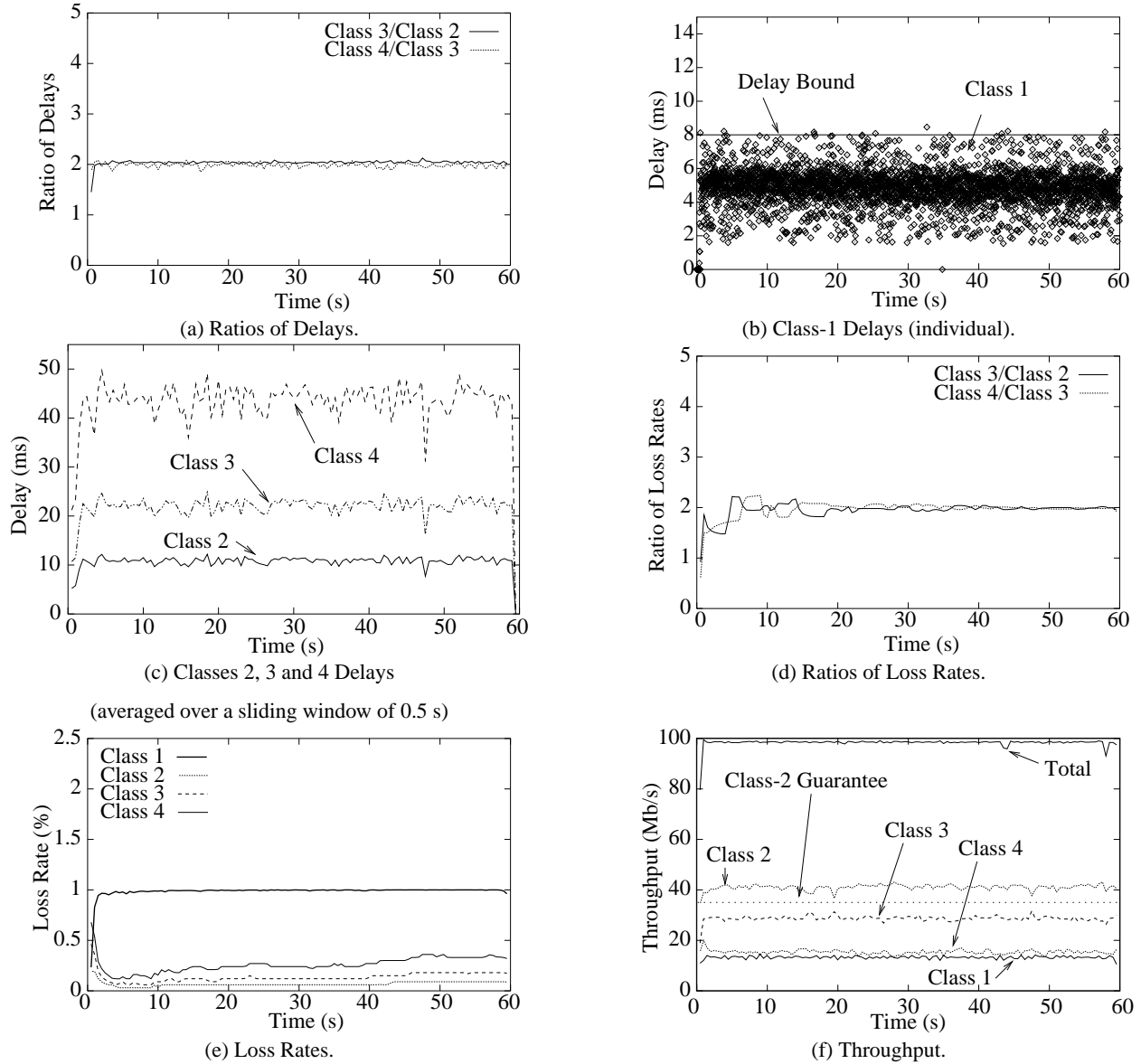
Set	enqueue		dequeue		Pred.
					μ_{pred}
	$\overline{X}$	s	$\overline{X}$	s	(Mbps)
1	**11323**	3140	**1057**	316	**291**
2	**10723**	2305	**1092**	340	**305**
3	**3039**	1512	**1138**	348	**864**
4	**2573**	668	**1078**	343	**988**

Table 3: **Overhead Measurements.** This table presents, for the four considered sets of service differentiation parameters, the average number of cycles ($\overline{X}$) consumed by the `enqueue` and `dequeue` operations, the standard deviation (s), and the predicted throughput μ_{pred} (in Mbps) that can be achieved. In the 1-GHz PCs we use, one cycle corresponds to one nanosecond.

feedback algorithms achieve the desired service differentiation, and utilize the entire available bandwidth, while maintaining stability throughout. Additional testbed experiments with variable loads are available in [16], and give further empirical evidence of the stability and robustness properties of the feedback algorithms.

7.3 Analysis of Overhead

Experiment 2 illustrated that our implementation in PC-routers with a 1 GHz processor can fully utilize the capacity of a 100 Mbps link. We next present a back-of-the-envelope analysis of the overhead of our implementation, where we attempt to predict the data rates that can be supported by our PC-router implementation, and where we measure the sensitivity of our implementation to the number of service constraints. We will show measurements of the `enqueue` and `dequeue` operations for four different sets of service differentiation parameters, tested for four traffic classes.

Set 1: Same differentiation as in Table 1.
Set 2: Set 1 with best-effort bounds from Set 1 removed.
Set 3: Set 2 with proportional differentiation from Set 1 removed.
Set 4: No service differentiation.

In the measurements we determine the number of cycles consumed for the `enqueue` and `dequeue` procedures. The TSC register of the Pentium processor is read at the beginning and at the end of the procedures, for each execution of the procedure.

We compiled our implementation with a code optimizer, in our case, we use the *gcc* v2.95.3 compiler [50] with the "-O2" flag set. The results of our measurements, collected under FreeBSD 4.5 and `altq-3.1`, are presented in Table 3, where we include the machine cycles consumed by the `enqueue` and `dequeue` operations. The measurements are averages of over 500,000 datagram transmissions on a heavily loaded link, using the same topology as in Figure 6. The measurements in Table 3 were collected at Router 1. Measurements collected at Router 2 showed deviations of no more than $\pm 5\%$ compared to Router 1.

Since the `enqueue` and `dequeue` operations are invoked once for each IP datagram, we can predict the maximum throughput of a PC-router to be

$$\mu_{pred} = \frac{F}{\alpha + \beta} \cdot \overline{P} \,, \tag{38}$$

where F denotes the CPU clock frequency in Hz, α denotes the number of cycles consumed by the `enqueue` operation, β denotes the number of cycles consumed by the `dequeue` operation, and $\overline{P}$ is the average size of a datagram. Eqn. (38) is only an estimate since it neglects operations that occur in an interrupt context (e.g., arrival of a packet at the input link). We point out, however, that operations performed in an interrupt context must have a negligible overhead for the router to operate properly [40]. In the case of our implementation in 1 GHz PCs, we have $F = 10^9$. Data from a recent report [5] indicates that the average size of an IP datagram on the Internet is $\overline{P} \approx 451$ bytes. Using these values for $\overline{P}$ and F in the above equation shows that, in the four sets of constraints considered, we estimate that our implementation can be run at data rates of at least 291 Mbps.[3] To forward packets at higher speeds, one can perform service rate adjustments only once in n packet arrivals. Such sampling comes at the expense of the accuracy of the service differentiation, but we did not observe much degradation in service differentiation for sampling intervals up to approximately 10-20 packets, which would be appropriate for Gbps links. We thus believe that our approach is viable at high speeds.

We next evaluate the sensitivity of the performance as a function of the number of constraints. The number of cycles consumed by the `dequeue` operation is independent of the set of constraints, since the `dequeue` operation solely uses the rate allocation provided. From Table 3, we observe that the overhead associated to the best-effort bounds (Set 3) is approximately 10% compared to a set with no service differentiation (Set 4). The overhead is 29% when comparing a set with best-effort bounds and proportional service differentiation (Set 1) to a set with proportional differentiation only (Set 2), which indicates that the overhead incurred by best-effort bounds is dependent on the presence of proportional differentiation. This result shows that the computation of the coefficient $K(n)$, used for proportional differentiation, is more complex when best-effort bounds are present. Proportional differentiation seems to incur more overhead than best-effort bounds, which is essentially due to the computations that need to be performed to dynamically update the value of the coefficient $K(n)$. However, in the set of constraints considered, there is a larger number of classes with proportional differentiation than classes with best-effort bounds, and thus, more computations are needed to enforce proportional differentiation.

8 Conclusions

This paper has suggested improvements to the type of service guarantees that can be given in a class-based service architecture without resource reservation. We introduced the concept of **best-effort bounds**.

---

[3]Previous measurements in [14] exhibited slightly more significant overhead under the same type of arrivals, but were collected using an older version of our software, and an older version of the FreeBSD operating system.

which were defined as guarantees that emulated absolute guarantees in a network without admission control and policing. We proposed mechanisms for routers that achieve best-effort bounds as well as proportional guarantees by selectively dropping traffic and by adjusting the traffic rate allocated to classes.

We used control theory to design the adaptive rate allocation and dropping mechanisms, by relying on feedback loops at link schedulers to enforce proportional differentiation of loss and delay and to give traffic classes best-effort bounds to loss, throughput and delay. The feedback loops do not require prior knowledge of traffic arrivals and do not require signaling. At times when not all best-effort bounds can be satisfied simultaneously, the feedback-based mechanisms relax some of the bounds using a predetermined precedence order.

We assessed the stability of our adaptive rate allocation and dropping mechanisms via simulation, and through experiments in a testbed network of BSD PC-routers. Testbed measurements allowed us to show that the implementation of the proposed mechanisms in 1 GHz PC-routers can fully utilize the available capacity of 100 Mbps, while enforcing the desired service differentiation. The implementation of our proposed mechanisms in PC-routers is available at `http://qosbox.cs.virginia.edu/software.html`, and has been included in the `altq` and KAME extensions to the BSD kernels.

As a final note, the mechanisms presented in this paper can be extended to include TCP congestion control algorithms [6, 27], as shown in [15].

References

[1] `http://qosbox.cs.virginia.edu/software.html`.

[2] `http://qosbox.cs.virginia.edu/snooplet.html`.

[3] The FreeBSD project. `http://www.freebsd.org`.

[4] The KAME project. `http://www.kame.net`.

[5] Packet sizes and sequencing, May 2001. `http://www.caida.org/outreach/resources/learn/packetsizes`.

[6] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. IETF RFC 2581, April 1999.

[7] K. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley, Reading, MA, 1995.

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. IETF RFC 2475, December 1998.

[9] S. Bodamer. A scheduling algorithm for relative delay differentiation. In *Proceedings of IEEE HPSR'00*, pages 357–364, Heidelberg, Germany, June 2000.

[10] U. Bodin, A. Jonsson, and O. Schelen. On creating proportional loss differentiation: predictability and performance. In *Proceedings of IWQoS'01*, pages 372–386, Karlsruhe, Germany, June 2001.

[11] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP). IETF RFC 2205, September 1997.

[12] K. Cho. A framework for alternate queueing: towards traffic management by PC-UNIX based routers. In *Proceedings of USENIX'98 Annual Technical Conference*, pages 247–258, New Orleans, LA, June 1998.

[13] N. Christin. *Quantifiable Service Differentiation for Packet Networks*. PhD thesis, University of Virginia, August 2003.

[14] N. Christin and J. Liebeherr. The QoSbox: A PC-router for quantitative service differentiation in IP networks. Technical Report CS-2001-28, University of Virginia, November 2001. `ftp://ftp.cs.virginia.edu/pub/techreports/CS-2001-28.pdf`. In revision.

[15] N. Christin and J. Liebeherr. Marking algorithms for service differentiation of TCP traffic. *Computer Communications*, 2004. In print.

[16] N. Christin, J. Liebeherr, and T. F. Abdelzaher. A quantitative assured forwarding service. Technical Report CS-2001-21, University of Virginia, August 2001. `ftp://ftp.cs.virginia.edu/pub/techreports/CS-2001-21.pdf`.

[17] N. Christin, J. Liebeherr, and T. F. Abdelzaher. A quantitative assured forwarding service. In *Proceedings of IEEE INFOCOM'02*, volume 2, pages 864–873, New York, NY, June 2002.

[18] D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, August 1998.

[19] R. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

[20] R. Cruz. A calculus for network delay, part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.

[21] B. Davie, A. Charny, J. Bennett, K. Benson, J.-Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. An expedited forwarding PHB. IETF RFC 3246, March 2002.

[22] C. Dovrolis and P. Ramanathan. Proportional differentiated services, part II: Loss rate differentiation and packet dropping. In *Proceedings of IWQoS'00*, pages 52–61, Pittsburgh, PA, June 2000.

[23] C. Dovrolis and P. Ramanathan. Dynamic class selection: From relative differentiation to absolute QoS. In *Proceedings of ICNP'01*, pages 120–128, Riverside, CA, November 2001.

[24] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proceedings of ACM SIGCOMM'99*, pages 109–120, Boston, MA., August 1999.

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

[26] L. Essafi, G. Bolch, and H. de Meer. Dynamic priority scheduling for proportional delay differentiated services. Technical Report TR-I4-01-03, University of Erlangen, March 2001.

[27] S. Floyd and T. Henderson. The NewReno modification to TCP's fast recovery algorithm. IETF RFC 2582, April 1999.

[28] G. Franklin, J. Powell, and M. Workman. *Digital control of dynamic systems*. Addison-Wesley, Menlo Park, CA, 3rd edition, 1998.

[29] B. Gaidioz, P. Primet, and B. Tourancheau. Differentiated fairness: Service model and implementation. In *Proceedings of IEEE HPSR'01*, pages 260–264, Dallas, TX, May 2001.

[30] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding PHB group. IETF RFC 2597, June 1999.

[31] C. V. Hollot, V. Misra, D. Towsley, and W. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *Proceedings of IEEE INFOCOM'01*, volume 3, pages 1726–1734, Anchorage, AK, April 2001.

[32] P. Hurley, J.-Y. Le Boudec, P. Thiran, and M. Kara. ABE: providing low delay service within best effort. *IEEE Networks*, 15(3):60–69, May 2001. See also `http://www.abeservice.org`.

[33] R. Jones. *netperf*: a benchmark for measuring network performance - revision 2.0. Information Networks Division, Hewlett-Packard Company, February 1995. See also `http://www.netperf.org`.

[34] A. Kumar, J. Kaur, and H. M. Vin. End-to-end proportional loss differentiation. Technical Report TR-01-33, University of Texas, February 2001.

[35] M. Leung, J. Lui, and D. Yau. Characterization and performance evaluation for proportional delay differentiated services. In *Proceedings of ICNP'00*, pages 295–304, Osaka, Japan, November 2000.

[36] R. Liao and A. Campbell. Dynamic core provisioning for quantitative differentiated service. In *Proceedings of IWQoS'01*, pages 9–26, Karlsruhe, Germany, June 2001.

[37] J. Liebeherr and N. Christin. JoBS: Joint buffer management and scheduling for differentiated services. In *Proceedings of IWQoS'01*, pages 404–418, Karlsruhe, Germany, June 2001.

[38] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son. Feedback control real-time scheduling: Framework, modeling and algorithms. *Journal of Real Time Systems*, 23(1–2), July 2002.

[39] Y. Lu, A. Saxena, and T. F. Abdelzaher. Differentiated caching services; A control-theoretical approach. In *Proceedings of the 21st International Conference on Distribute d Computing Systems*, pages 615–624, Phoenix, AZ, April 2001.

[40] J. Mogul and K. Ramakrishnan. Eliminating receive livelock in an interrupt-driven kernel. *ACM Transactions on Computer Systems*, 15(3):217–252, 1997.

[41] Y. Moret and S. Fdida. A proportional queue control mechanism to provide differentiated services. In *Proceedings of the International Symposium on Computer and Information Systems (ISCIS)*, pages 17–24, Belek, Turkey, October 1998.

[42] T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Barghavan. Delay differentiation and adaptation in core stateless networks. In *Proceedings of IEEE INFOCOM'00*, pages 421–430, Tel-Aviv, Israel, April 2000.

[43] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers. IETF RFC 2474, December 1998.

[44] K. Nichols, V. Jacobson, and L. Zhang. Two-bit differentiated services architecture for the Internet. IETF RFC 2638, July 1999.

[45] A. Parekh and R. Gallagher. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[46] S. Parekh, N. Gandhi, J. Hellerstein, D. Tilbury, T. S. Jayram, and J. Bigus. Using control theory to achieve service level objectives in performance management. *Journal of Real-Time Systems*, 23(1–2), July 2002.

[47] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. IETF RFC 3031, January 2001.

[48] H. Saito, C. Lukovszki, and I. Moldován. Local optimal proportional differentiated services scheduler for relative differentiated services. In *Proceedings of IEEE ICCCN'00*, pages 554–550, Las Vegas, NV, October 2000.

[49] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round-robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, June 1996.

[50] R. Stallman. *Using and Porting the GNU Compiler Collection (gcc).* iUniverse Inc., 2000.

[51] D. Stiliadis and A. Varma. Latency-rate servers: a general model for analysis of traffic scheduling algorithms. *IEEE/ACM Transactions on Networking*, 6(5):611–624, 1998.

[52] A. Striegel and G. Manimaran. Packet scheduling with delay and loss differentiation. *Computer Communications*, 25(1):21–31, January 2002.

[53] N. Taft, S. Bhattacharyya, J. Jetcheva, and C. Diot. Understanding traffic dynamics at a backbone POP. In *Proceedings of SPIE ITCOM Workshop on Scalability and Traffic Control in IP Networks*, number 4526, Denver, CO, August 2001.

[54] L. Zhang. Virtual clock: A new traffic control algorithm for packet switched networks. *ACM Transactions on Computer Systems*, 9(2):101–125, May 1991.