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

Nicolas Christin, *Member, IEEE,* Jörg Liebeherr, *Senior Member, IEEE,* and Tarek Abdelzaher, *Member, IEEE*

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 measurement experiments of a kernel-level implementation in FreeBSD PC-routers with multiple 100 Mbps Ethernet interfaces, complemented with simulations of larger scale networks.**

Keywords— **Service Differentiation, Buffer Management, Scheduling, Feedback Control, Best-Effort Bounds.**

I. 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 usually 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 [3], a dedicated server [4], resource provisioning [5], or manual configuration [4]. 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* [6], consists of two services. The Expedited Forwarding (EF, [7]) service provides absolute delay guarantees to predefined amounts of traffic, and requires traffic policing, admission control, and resource reservations. The Assured Forwarding (AF, [8]) service enforces isolation between classes, and qualitative loss differentiation between different drop precedence levels within each class. The Proportional Service Differentiation architecture [9, 10] 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 [6, 11, 12, 13]. 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

N. Christin is with Carnegie Mellon University, Information Networking Institute and CyLab Japan. E-mail: nicolasc@cmu.edu.

J. Liebeherr is with the Department of Electrical and Computer Engineering, University of Toronto. E-mail: jorg@comm.utoronto.ca.

T. Abdelzaher is with the Department of Computer Science, University of Illinois at Urbana-Champaign.

This work is supported in part by the National Science Foundation through grants ANI-9730103 and ANI-0085955. Most of the work was done while all three authors were with the Department of Computer Science, University of Virginia. Portions of this paper appeared in a preliminary form in [1] and in [2].

(ABE, [11]) 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 [6,11,12,13] 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 a kernel-level software implementation in FreeBSD PC routers. This implementation is currently being disseminated as part of the ALTQ [14] and KAME [15] packages.

The remainder of this paper is organized as follows. In Section II, we expand our discussion of the related work. In Section III, we present a formal description of the proposed service. In Sections IV and V, 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 VI, we present an implementation of the mechanisms in FreeBSD PC-routers. We evaluate our implementation in Section VII and present brief conclusions in Section VIII.

II. Related Work

Proportional service differentiation, originally proposed by Dovrolis et al. [16], 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 [10], Local-Optimal Proportional Differentiation [17], or Mean-Delay Proportional [18] 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 [19], or Backlog Proportional Rate [16] 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 [20] 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 [9]. Enhancements to the mechanisms discussed in [9] can provide proportional loss differentiation over arbitrary timescales [21].

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 [13, 22]. For instance, in [22], 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 [13] further enhances class-based service differentiation by providing limited support for absolute bounds on loss and delay. To that effect, the authors of [13] 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 [13], 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 [1] for a more comprehensive discussion. For instance, the Dynamic Core Provisioning service [12] supports absolute delay bounds, and qualitative loss and throughput differentiation, but no proportional differentiation. The mechanisms used in [12] 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.

III. 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 ≤ 2 ms,"
(G2) "Class-2 loss rate $\leq 1\%$,"
(G3) "Class-3 service rate ≥ 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 (G2), and both (G1) and (G2) can be violated to satisfy (G3). (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 [23].

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

A. Service Provisioning

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 [24, 25]. We will refer to Fig. 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 traffic is actually sent in discrete sized packets. In Section VI, 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 gener-

[1] The beginning of the current busy period is defined as the last time when the transmission queue at the output link was empty.

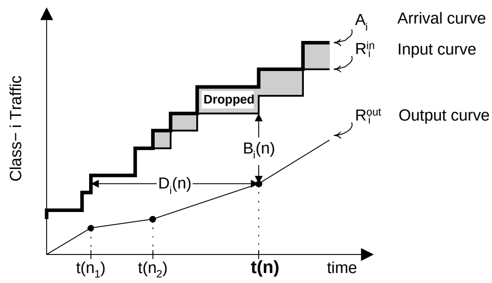


Fig. 1. **Delay and backlog at the transmission queue of an output link.** A_i is the arrival curve, R_i^{in} is the input curve and R_i^{out} is the output curve.

ally 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

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

In Fig. 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 Fig. 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

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

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

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

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

$$D_i(n) \leq d_i \ , \qquad (4)$$

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

$$p_i(n) \leq L_i \ . \qquad (5)$$

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

$$r_i(n) \geq \mu_i \ . \qquad (6)$$

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

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

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

and

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

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

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 infeasible sets of proportional service guarantees, there should be at most one proportional guarantee between two classes with consecutive indices. For instance, between class 1 and class 2, there should not be a both a proportional throughput guarantee and a proportional delay guarantee.
- 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 [26]. 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 VI

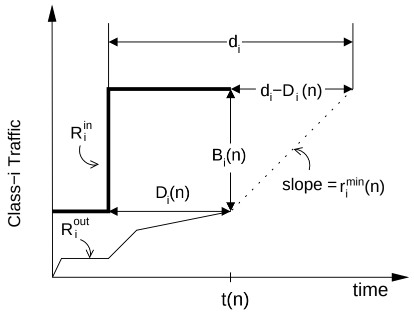


Fig. 2. **Determining service rates for delay bounds.**

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.

B. 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 Fig. 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 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]

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

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

So, the service rate can take any value $r_i(n)$ such that $r_i^{\min}(n) \leq r_i(n) \leq C - \sum_{j \neq i} r_j^{\min}(n)$, 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 $k < n$. 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, at the n-th event, no feasible service rate allocation for realizing all desired delay and throughput differentiation exists, or if there is a buffer overflow, 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, after simplification, $p_i(n) = (A_i(n-1) - R_i^{in}(n-1))/A_i(n) + l_i(n)/A_i(n)$, 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 VI, we describe the interaction of the two feedback problems.

IV. 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 [27,28,29,30]. 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 [31], resort to algorithms such as gradient estimators. However, 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.

A. 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:

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

Let $m_i = \prod_{j=1}^{i-1} \alpha_j^{\text{del}}$ 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, for all i, j and n, we have $D_i^*(n) = D_j^*(n)$, or, equivalently, for all i and n, $D_i^*(n) = \overline{D}^*(n)$, where

$$\overline{D}^*(n) := \frac{1}{N} \sum_i D_i^*(n) \;. \tag{16}$$

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 $(B_{i+1}(n) \cdot D_i(n))/(B_i(n) \cdot D_{i+1}(n)) = \alpha_i^{\text{tput}}$, 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. We will argue in the stability analysis of the delay feedback loops that we can 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.

B. 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.,

$$e_i(n) = \overline{D}^*(n) - D_i^*(n) \;. \tag{17}$$

Note that the sum of the errors is always zero, that is, for all n, $\sum_i e_i(n) = N\overline{D}^*(n) - \sum_i D_i^*(n) = 0$. 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. (17), 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

$$\Delta r_i(n) = K(n) \cdot e_i(n) \;, \tag{18}$$

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

$$\sum_i \Delta r_i(n) = K(n) \sum_i e_i(n) = 0\ . \qquad (19)$$

From Eqn. (19), 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.

C. 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 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 VII 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:

$$\overline{r_i}(n) = \frac{B_i(n-\tau_i)}{D_i(n)}\ . \qquad (20)$$

From Eqn. (20), $\overline{r_i}(n)$ 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:

$$\overline{r_i}(n+1) = \frac{(\tau_i - 1)\overline{r_i}(n) + r_i(n)}{\tau_i}\ . \qquad (21)$$

Let us now define

$$\Delta\overline{r_i}(n+1) = \overline{r_i}(n+1) - \overline{r_i}(n)\ . \qquad (22)$$

Combining Eqs. (21) and (22), we get

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

Eqn. (23) 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. (20), $D_i(n) = B_i(n-\tau_i)/\overline{r_i}(n)$, and $D_i(n+1) = B_i(n+1-\tau_i)/\overline{r_i}(n+1)$, we get

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

Eqn. (24) is not linear in $\overline{r_i}$. We use (A4) to linearize Eqn. (24), by means of a first order Taylor series expansion around $\overline{r_i}(n)$. (A4) indeed 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. (24) as

$$\Delta D_i(n+1) = -\frac{B_i(n-\tau_i)}{\overline{r_i}^2(n)} \Delta\overline{r_i}(n+1) + \omega_i(n) , \quad (25)$$

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

$$D_i(n+1) = \sum_{k=0}^{n+1} \Delta D_i(k) , \quad (26)$$

$D_i(n+1)$ is used to compute $D_i^*(n+1)$, using Eqn. (15). Finally, from Eqs. (16) and (17), 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.

D. 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.

Frequency-domain expression for the feedback loop. We express the delay feedback loop in the frequency domain using using z-transforms of Eqs. (15)–(26). 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) and (17) are unchanged when using z-transforms. Assumptions (A2)–(A4) enable us to approximate $K(n)$ by a constant over the time a given packet is backlogged, and to assume that Eqn. (18) remains unchanged when using z-transforms. Likewise, per (A3) and (A4), $B_i(n-\tau_i)$ and $\overline{r_i}(n)$ are assumed constant over the time a given packet is backlogged, so that we can assume that Eqn. (25) is also unchanged when using z-transforms. Eqn. (23) 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, after reordering, that

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

Because $D_i(n) = 0$ for any $n \leq 0$, $\Delta D_i(0) = 0$. So, the z-transform of Eqn. (26) 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 [32], 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 [31].

Stability analysis. We obtain stability bounds on $K(n)$ from a standard control theory result [32]. 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

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

The characteristic equation of the approximate system, $1 + G'(z) = 0$, has exactly one root, $\hat{z} = 1 + \left(\left(\prod_{j \neq i} m_j\right) B_i(n-\tau_i) K(n)\right) / \overline{r_i}^2(n)$. The stability condition, $|\hat{z}| < 1$, implies

$$1 + \frac{\left(\prod_{j \neq i} m_j\right) B_i(n-\tau_i) K(n)}{\overline{r_i}^2(n)} \geq -1 \quad (27)$$

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

All quantities in Eqn. (28), with the exception of $K(n)$, are positive. Hence, the condition described by Eqn. (28) reduces to $K(n) \leq 0$. Eqn. (27) becomes, after reordering,

$$K(n) \geq -2 \cdot \frac{\overline{r_i}^2(n)}{\left(\prod_{j \neq i} m_j\right) B_i(n-\tau_i)} . \quad (29)$$

Since, with Eqn. (20), we can write $\overline{r_i}^2(n)/B_i(n-\tau_i) = B_i(n-\tau_i)/D_i(n)^2$, Eqn. (29) can be expressed as

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

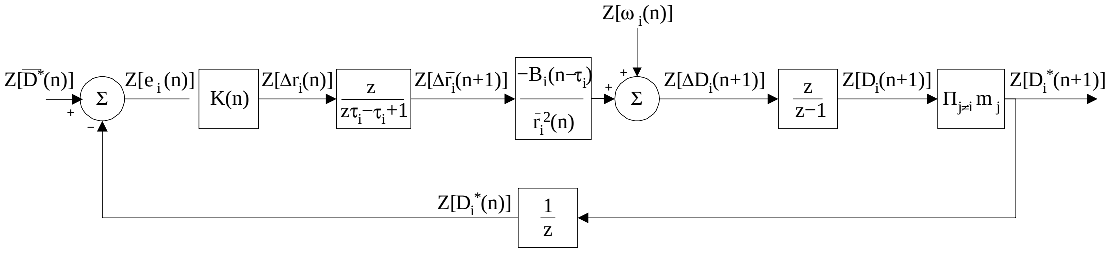


Fig. 3. **The class-i delay feedback loop.** This model uses z-transforms of the relationships derived in Section IV-B.

The condition given by Eqn. (30) 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. (30). 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. (19) to hold, we finally get

$$-2 \cdot \min_i \left\{ \frac{B_i(n)}{\prod_{j\neq i} m_j \cdot D_i^2(n)} \right\} \leq K(n) \leq 0 \ . \tag{31}$$

The condition in Eqn. (31) 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. (31) ensures stability of the (non-linear) delay feedback loops. However, we can use Eqn. (31) as a design guideline for $K(n)$.

E. Including the Absolute Delay and Rate Constraints

The condition on $K(n)$ we obtained in Eqn. (31) 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. (19). To respect the saturation constraint, $K(n)$ has to satisfy $r_i(n-1) + K(n)e_i(n) \geq r_i^{\min}(n)$. Applying that $K(n)$ to all control loops implies

$$K(n) \geq \max_i \left(\frac{r_i^{\min}(n) - r_i(n-1)}{e_i(n)} \right) \ . \tag{32}$$

If $\max_i((r_i^{\min}(n) - r_i(n-1))/e_i(n)) > 0$, we cannot satisfy both Eqn. (32) and $K(n) \leq 0$, required by Eqn. (31). 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. (31) or Eqn. (32) according to a given relaxation order. For instance, giving best-effort bounds higher precedence than proportional differentiation results in relaxing Eqn. (31) and satisfying Eqn. (32).

V. 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 IV-A.

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

$$B_k(n) = B_k(n-1) + a_k(n) - l_k(n) - \Delta t(n-1) r_k(n) \ . \tag{33}$$

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. (33) and the work-conserving property $\sum_k r_k(n) = C$, becomes

$$\sum_{k=1}^{N} \big(B_k(n-1) + a_k(n) - l_k(n) \big) - \Delta t(n-1) C \leq B \ . \tag{34}$$

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. (33), we obtain the following condition:

$$\begin{aligned} \textstyle\sum_{k=1}^{N} \max\{ & \tfrac{B_k(n-1) - r_k(n-1)\Delta t(n-1) + a_k(n) - l_k(n)}{d_k - D_k(n)}, \\ & \mu_k \cdot \chi_{B_k(n)>0} \} \leq C \ . \end{aligned} \tag{35}$$

If either of Eqs. (34) or (35) 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, for all i, j, and n, $p_i^*(n) = p_j^*(n)$, which is equivalent to $p_i^*(n) = \bar{p}^*(n)$, 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. (34) and (35) 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. (34) and (35) 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:

$$l_i(n) = \begin{cases} l^*_i(n) & \text{if } i = i_1, i_2, \ldots, i_{\hat{k}-1} \ , \\ \hat{x}(n) & \text{if } i = i_{\hat{k}} \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{36}$$

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 (35), 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 VI, loss bounds take precedence over delay and rate bounds, and condition (35) 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 $\overline{p}^*(n)$. 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.

VI. Implementation

We implemented the feedback loops presented in Sections IV and V in PC-routers running the FreeBSD 4.3 [33] operating system, using the `altq-3.0` package [14]. `altq` allows programmers to modify the operations of the transmission queue in the IP layer of the FreeBSD kernel. Our implementation [34] has been included in `altq-3.1`, and in KAME [15]. For a detailed discussion of the implementation issues, we refer the reader to [35]. 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).

We use the DiffServ codepoint (DSCP, [36]) 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.

A. 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 V to determine if and how much traffic needs to be dropped from each class. In our implementation, the algorithm of Section V is run twice. The first time, buffer overflows are resolved by ignoring condition (35); The second time, potential violations of delay and throughput bounds are resolved by ignoring condition (34).

Next, the `enqueue` procedure computes new values for $r_i^{\min}(n)$ from Eqn. (11), and determines new service rates, using Eqs. (12) and (18), with the constraints on $K(n)$ given in Eqs. (31) and (32). If no feasible value for $K(n)$ exists, Eqn. (31) is ignored, thereby giving delay bounds precedence over proportional delay (and throughput) differentiation.

B. 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 [37] or PGPS [38]. These scheduling algorithms translate a rate allocation, into virtual deadlines of individual packets. In our implementation, we use a modified Deficit Round Robin [39] 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 $i = \arg\max_k \{R_k^{out}(n) - Xmit_k(n)\}$. In other words, the `dequeue` procedure selects the class whose service is the most behind its allocated service rate.

Class	Service Guarantees				
	d_i	L_i	μ_i	α_i^{del}	α_i^{loss}
1	8 ms	1 %	–	–	–
2	–	–	35 Mbps	2	2
3	–	–	–	2	2
4	–	–	–	N/A	N/A

(a) Desired service differentiation.

Class	No. of flows	Type	
		Protocol	Traffic
1	12	UDP	On-off
2	12	TCP	Greedy/On-off
3	12	TCP	Greedy/On-off
4	12	TCP	Greedy/On-off

(b) Traffic mix.

TABLE I

TESTBED MEASUREMENTS: EXPERIMENTAL PARAMETERS.

VII. EXPERIMENTAL EVALUATION

In this section, we present experimental measurements of our FreeBSD implementation described in Section VI in a testbed of PC-routers, which we complement with larger scale simulation experiments. We point out that [1] contains additional experiments.

A. *Testbed Measurements*

We first demonstrate that the mechanisms we propose can be implemented at relatively high-speeds, and achieve the desired service differentiation for a mix of best-effort bounds and proportional differentiation. We consider TCP and UDP traffic competing at a single bottleneck link of capacity $C = 100$ Mbps, governed by a router interface with a buffer size set to $B = 200$ packets. The router and the sources of traffic are Dell PowerEdge 1550 servers with 1 GHz Intel Pentium-III processors and 256 MB of RAM. The system software is FreeBSD 4.3 and `altq-3.0`. The router is equipped with multiple 100 Mbps-Ethernet interfaces.

We consider four traffic classes and we provide the service differentiation described in Table I(a). The traffic mix, the number of flows per class, and the characterization of the flows for each source is as shown in Table I(b). Class 1 traffic consists of on-off UDP flows, and the other classes consist of TCP (Reno) flows. Traffic is generated using *netperf* v2.1pl3 [40]. We configure the TCP sources to be greedy during time intervals $[0s, 10s]$, $[20s, 30s]$ and $[40s, 50s]$. In the remaining time intervals $(10s, 20s)$, $(30s, 40s)$, and $(50s, 60s)$, the TCP sources send chunks of 8KB of data and pause for 175 ms between the transmission of each chunk. 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. The bottleneck link is shared by both data packets and TCP acknowledgments coming back from the destinations. We plot the offered load at the router in Fig. 4. When all sources are simultaneously transmitting, congestion control at the TCP sources maintains the total load at a level of about 99% of the link capacity. As soon as TCP sources act as on-off sources, the load suddenly drops to about 30% of the link capacity.

In Fig. 5, we present our measurements of the service received at the bottleneck link. All datapoints correspond to moving averages over sliding windows of size 0.5 seconds, except in Fig. 5(b), which presents the delays of each class-1 packet.

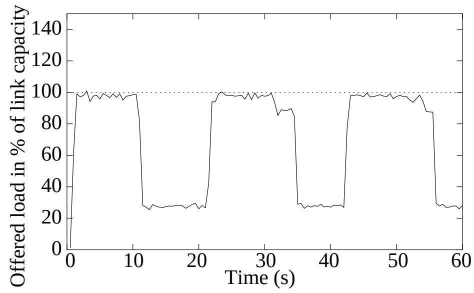
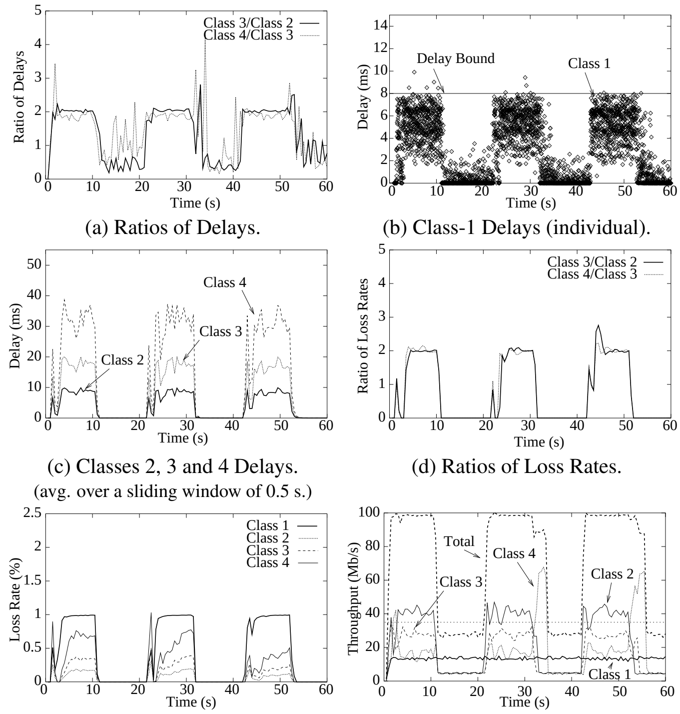


Fig. 4. **Testbed measurements: Offered load.** The graph shows the offered load at the bottleneck link.



(a) Ratios of Delays. (b) Class-1 Delays (individual).

(c) Classes 2, 3 and 4 Delays. (avg. over a sliding window of 0.5 s.) (d) Ratios of Loss Rates.

(e) Loss Rates. (f) Throughput.

Fig. 5. **Testbed measurements: Service differentiation at bottleneck link.** The graphs show the service obtained by each class at the bottleneck link.

In Fig. 5(a), we present the ratios of the delays of classes 4 and 3, and the delays of classes 3 and 2. We observe that when the load is high, in time intervals $[0s, 10s]$, $[20s, 30s]$, and $[40s, 50s]$, the target value of $\alpha_2^{\mathrm{del}} = \alpha_3^{\mathrm{del}} = 2$ is achieved. When the load is low, we observe oscillations in the ratios of delays, but all classes get low delays, as shown in Figs. 5(b) and 5(c), and one can argue that there is no need for differentiation since all classes receive a high-grade service. We also see that, at times $t = 0$, $t = 20$ and $t = 40$, when the load increases abruptly over a short period of time, the delay differentiation is realized almost immediately. This confirms that the algorithm quickly reacts to rapid increases in the offered load.

In Fig. 5(b), we show the delay of class-1 packets. The best-effort delay bound of $d_1 = 8$ ms is satisfied for most ($> 99\%$) 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 ms at either Router 1 or 2. Fig. 5(c) indicates that delay values of other classes are in the range 10-50 ms.

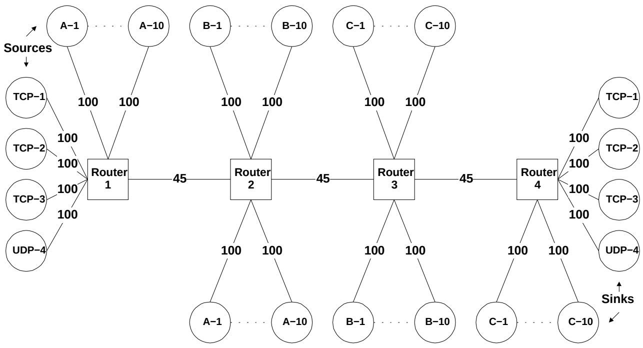


Fig. 6. **Simulation: Network topology.** The labels of the links denote the links capacities in Mbps.

Fig. 5(d) presents plots of the ratios of loss rates averaged over a sliding window of size 0.5 s, and show that proportional loss differentiation is realized, with the desired factor $\alpha_2^{\text{loss}} = \alpha_3^{\text{loss}} = 2$, whenever there are packet losses. Fig. 5(e) shows the loss rate experienced by class-1 traffic, and we see that, even at times of packet drops, the loss rate of class 1 remains below the loss guarantee of 1%. Loss rates of other classes are below 1%, which indicates that traffic is dropped mostly to satisfy the delay bound on Class 1.

We graph the throughput experienced by each class and by the traffic aggregate in Fig. 5(f). The figure illustrates that the throughput guarantee $\mu_2 = 35$ Mbps is met, that no class experiences starvation, and that our implementation in PC-routers with a 1 GHz processor can fully utilize the capacity of a 100 Mbps link when needed.

Finally, we measure the number of CPU cycles consumed by the `enqueue` and `dequeue` procedures at the bottleneck link, by reading the timestamp counter register of the Pentium processor. We compute the average and standard deviation of the number of cycles over 500,000 packet transmissions. In the following table, we compare measurements for a set of four classes with differentiation parameters given in Table I(a), to a system of four classes without any guarantees:

Guarantees	Enqueue		Dequeue	
	Avg.	Std. Dev.	Avg.	Std. Dev.
with	11323	3140	1057	316
without	2573	668	1078	343

The table shows that the overhead for the `enqueue` operation, which implements the feedback algorithms, is significant. At the same time, a back-of-the-envelope estimation (ignoring other tasks the router may have to perform in the background) indicates that a 1 GHz PC can enqueue and dequeue more than 80,000 packets per second. Considering that the average size of an IP packet on the Internet is $\overline{P} \approx 450$ bytes [42], this results in a maximum throughput of about 290 Mbps. We refer to [1] for a more detailed evaluation of the overhead associated to our proposed algorithms.

B. Simulation of Multiple Bottlenecks Links

We complement our testbed measurements by simulating a network with multiple hops and propagation delays. To that effect, we implemented our closed-loop algorithm in the *ns-2* network simulator [41]. This implementation has been included in the standard *ns-2* distribution, since *ns*-2.26.

Class	Service Guarantees			
	d_i	L_i	α_i^{del}	α_i^{loss}
1	2 ms	0.5 %	–	–
2	–	–	4	2
3	–	–	4	2
4	–	–	N/A	N/A

(a) Desired service differentiation.

Flow	Class	Type			
		Proto.	Traffic	On	Off
TCP-1	1	TCP	Greedy	N/A	N/A
TCP-2	2	TCP	Greedy	N/A	N/A
TCP-3	3	TCP	Greedy	N/A	N/A
UDP-4	4	UDP	Pareto on-off	10 ms	10 ms
A-1	1	TCP	Exp. on-off	650 pkts	200 ms
A-2, A-3	2	TCP	Exp. on-off	650 pkts	200 ms
A-4 to A-6	3	TCP	Exp. on-off	650 pkts	200 ms
A-7 to A-10	4	UDP	Pareto on-off	26 ms	100 ms

(b) Traffic mix.

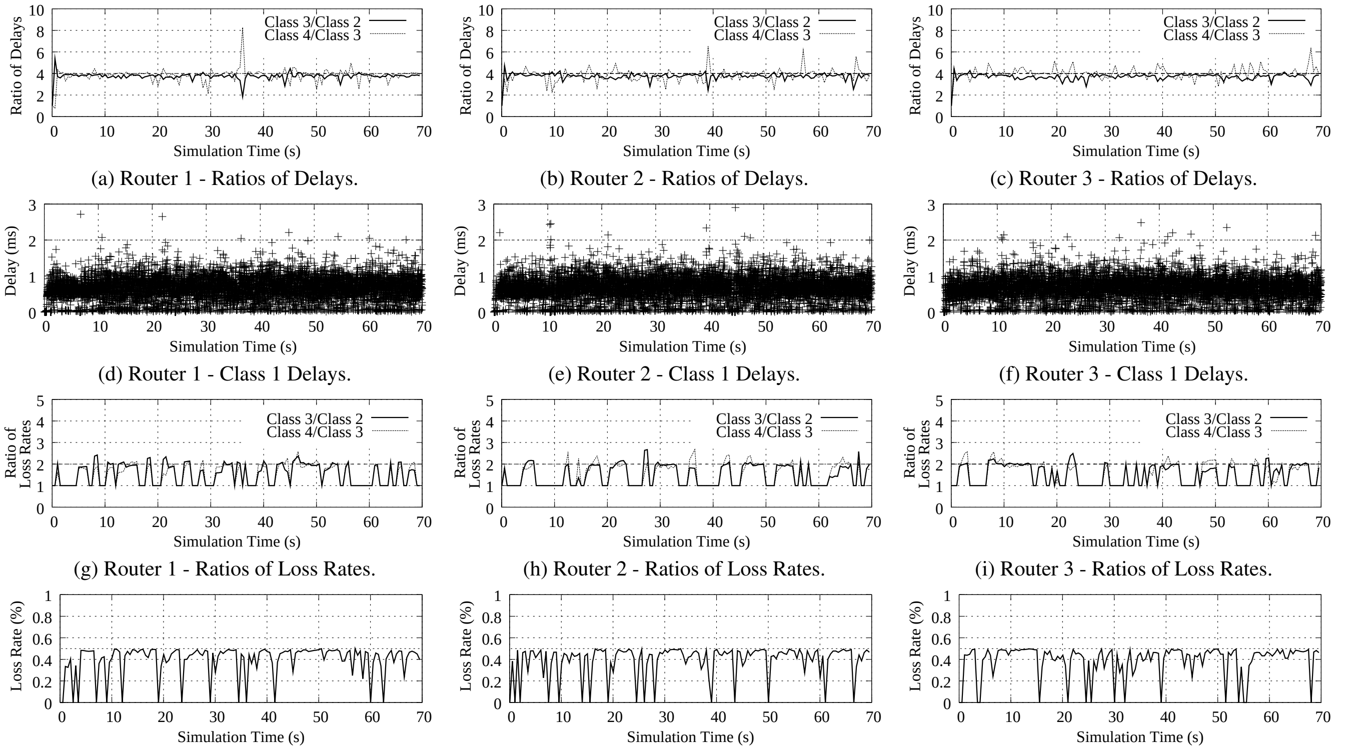
TABLE II
SIMULATION: EXPERIMENTAL PARAMETERS.

We simulate a network with a topology as shown in Fig. 6. We have four routers, each with a maximum buffer size of 500 packets, connected by three 45 Mbps links, and sources and sinks connected to the routers by independent 100 Mbps links. Each 45 Mbps link has a propagation delay of 3 ms, and each 100 Mbps link has a propagation delay of 1 ms. There are four classes of traffic. The service guarantees are given in Table II(a) and the composition of the traffic mix is given in Table II(b). Traffic consists of a mix of TCP (Reno) and UDP flows. TCP flows are either greedy, to model long file transfers, or on-off flows with exponentially distributed on and off periods, to model short, successive file transfers (e.g., HTTP requests). UDP flows are on-off flows using a Pareto distribution for the on and off periods, with a shape parameter $\alpha = 1.9$.

Cross-traffic flows (denoted by A-1, . . ., C-10) start transmitting at time $t = 0$ s. The flows TCP-1, TCP-2, TCP-3 and UDP-4 start transmitting at time $t = 10$ s. All flows consists of packets with a fixed size of 500 bytes, and the experiment lasts 70 seconds of simulated time. The aggregate load at the bottlenecks is roughly constant and equal to the capacity of the bottleneck links, but the introduction of new flows at $t = 10$ s significantly changes the contribution of each class to the traffic mix, and is likely to result in violations of Assumption (A3). Hence the proposed simulation should allow us to test the sensitivity of our approach to its design assumptions.

From Tables II(a) and (b), Classes 1, 2 and 3 only consist of TCP traffic, and Class 4 only consists of UDP traffic. Initially Class 1 contributes 10% of the aggregate cross-traffic, Class 2 contributes 20%, Class 3 contributes 30% and Class-4 contributes 40 %.

We graph the per-class queuing delays and per-class loss rates at each of the first three routers in Fig. 7, starting at time $t = 0$ s. Given that the aggregate arrival rate at Router 4 is always less than the total output capacity of Router 4, Router 4 never experiences any backlog, and the queuing delays and loss rates are constantly equal to zero. With the exception of Fig. 7(d), (e) and (f), where we plot individual packet delays, each point on Fig. 7 represents an average over a sliding window of size 0.5 s. Fig. 7 shows that the proposed algorithm manages to enforce all



(a) Router 1 - Ratios of Delays. (b) Router 2 - Ratios of Delays. (c) Router 3 - Ratios of Delays.

(d) Router 1 - Class 1 Delays. (e) Router 2 - Class 1 Delays. (f) Router 3 - Class 1 Delays.

(g) Router 1 - Ratios of Loss Rates. (h) Router 2 - Ratios of Loss Rates. (i) Router 3 - Ratios of Loss Rates.

(j) Router 1 - Class 1 Loss Rates. (k) Router 2 - Class 1 Loss Rates. (l) Router 3 - Class 1 Loss Rates.

Fig. 7. **Multiple node simulation.** The graphs show the delays and loss rates encountered at each router by Class 1 traffic, and the ratios of delays and the ratios of loss rates for Classes 2, 3 and 4 at each router. The absolute constraints and the target ratios are indicated by straight dashed lines.

proposed service guarantees at each router, with only a couple of transient violations of the absolute delay bound on Class 1 at Router 1, and that the algorithm seems to respond appropriately to transient changes such as the introduction of additional traffic at time $t = 10$ s.

VIII. Conclusions

This paper suggests improvements to the type of service guarantees that can be given in a class-based service architecture without resource reservation. We introduce the concept of best-effort bounds, defined as guarantees that emulate absolute guarantees in a network without admission control and policing. We propose 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 use 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 assess the stability of our adaptive rate allocation and dropping mechanisms through experiments in a testbed network of BSD PC-routers and simulations. Testbed measurements 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 publicly available [34], and is 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 [43,44], as shown in [45]. Used in conjunction with routing mechanisms that can perform route-pinning, such as MPLS [46], our adaptive rate allocation and buffer management mechanisms can be used as a building block for end-to-end service differentiation. We refer to [1] for a more thorough inspection of the interaction of end-to-end performance with the per-hop service proposed here.

References

[1] N. Christin, *Quantifiable Service Differentiation for Packet Networks*, Ph.D. thesis, Univ. Virginia, Aug. 2003.

[2] N. Christin, J. Liebeherr, and T. F. Abdelzaher, "A quantitative assured forwarding service," *Proc. IEEE INFOCOM'02*, New York, NY, June 2002, vol. 2, pp. 864–873.

[3] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (RSVP)," IETF RFC 2205, Sept. 1997.

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

[5] D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Trans. Net.*, vol. 6, no. 4, pp. 362–373, Aug. 1998.

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

[7] 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, Mar. 2002.

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

[9] C. Dovrolis and P. Ramanathan, "Proportional differentiated services, part II: Loss rate differentiation and packet dropping," *Proc. IWQoS'00*, Pittsburgh, PA, June 2000, pp. 52–61.

[10] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional differentiated

services: Delay differentiation and packet scheduling," *IEEE/ACM Trans. Net.*, vol. 10, no. 1, pp. 12–26, Feb. 2002.

[11] P. Hurley, J.-Y. Le Boudec, P. Thiran, and M. Kara, "ABE: providing low delay service within best effort," *IEEE Networks*, vol. 15, no. 3, pp. 60–69, May 2001, See also `http://www.abeservice.org`.

[12] R. Liao and A. Campbell, "Dynamic core provisioning for quantitative differentiated service," *Proc. IWQoS'01*, Karlsruhe, Germany, June 2001, pp. 9–26.

[13] J. Liebeherr and N. Christin, "JoBS: Joint buffer management and scheduling for differentiated services," *Proc. IWQoS'01*, Karlsruhe, Germany, June 2001, pp. 404–418.

[14] K. Cho, "A framework for alternate queueing: towards traffic management by PC-UNIX based routers," *Proc. USENIX'98*, New Orleans, LA, June 1998, pp. 247–258.

[15] "The KAME project," `http://www.kame.net`.

[16] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional differentiated services: Delay differentiation and packet scheduling," *Proc. ACM SIGCOMM'99*, Boston, MA, Aug. 1999, pp. 109–120.

[17] H. Saito, C. Lukovszki, and I. Moldován, "Local optimal proportional differentiated services scheduler for relative differentiated services," *Proc. IEEE ICCCN'00*, Las Vegas, NV, Oct. 2000, pp. 554–550.

[18] T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Barghavan, "Delay differentiation and adaptation in core stateless networks," *Proc. IEEE INFOCOM'00*, Tel-Aviv, Israel, Apr. 2000, pp. 421–430.

[19] Y. Moret and S. Fdida, "A proportional queue control mechanism to provide differentiated services," *Proc. ISCIS'98*, Belek, Turkey, Oct. 1998, pp. 17–24.

[20] S. Bodamer, "A scheduling algorithm for relative delay differentiation," *Proc. IEEE HPSR'00*, Heidelberg, Germany, June 2000, pp. 357–364.

[21] U. Bodin, A. Jonsson, and O. Schelen, "On creating proportional loss differentiation: predictability and performance," *Proc. IWQoS'01*, Karlsruhe, Germany, June 2001, pp. 372–386.

[22] A. Striegel and G. Manimaran, "Packet scheduling with delay and loss differentiation," *Comp. Comm.*, vol. 25, no. 1, pp. 21–31, Jan. 2002.

[23] D. Stiliadis and A. Varma, "Latency-rate servers: a general model for analysis of traffic scheduling algorithms," *IEEE/ACM Trans. Net.*, vol. 6, no. 5, pp. 611–624, 1998.

[24] R. Cruz, "A calculus for network delay, part I: Network elements in isolation," *IEEE Trans. Info. Theory*, vol. 37, no. 1, pp. 114–131, Jan. 1991.

[25] R. Cruz, "A calculus for network delay, part II: Network analysis," *IEEE Trans. Info. Theory*, vol. 37, no. 1, pp. 132–141, Jan. 1991.

[26] M. Leung, J. Lui, and D. Yau, "Characterization and performance evaluation for proportional delay differentiated services," *Proc. ICNP'00*, Osaka, Japan, Nov. 2000, pp. 295–304.

[27] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," *Proc. IEEE INFOCOM'01*, Anchorage, AK, Apr. 2001, vol. 3, pp. 1726–1734.

[28] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son, "Feedback control real-time scheduling: Framework, modeling and algorithms," *J. Real-Time Sys.*, vol. 23, no. 1–2, July 2002.

[29] Y. Lu, A. Saxena, and T. F. Abdelzaher, "Differentiated caching services; A control-theoretical approach," *Proc. IEEE ICDCS'01*, Phoenix, AZ, Apr. 2001, pp. 615–624.

[30] 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," *J. Real-Time Sys.*, vol. 23, no. 1–2, July 2002.

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

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

[33] "The FreeBSD project," `http://www.freebsd.org`.

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

[35] N. Christin and J. Liebeherr, "The QoSbox: Quantitative service differentiation in BSD routers," *Computer Networks*, 2006. In print.

[36] 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, Dec. 1998.

[37] L. Zhang, "Virtual clock: A new traffic control algorithm for packet switched networks," *ACM Trans. Comp. Sys.*, vol. 9, no. 2, pp. 101–125, May 1991.

[38] A. Parekh and R. Gallagher, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Trans. Net.*, vol. 1, no. 3, pp. 344–357, June 1993.

[39] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round-robin," *IEEE/ACM Trans. Net.*, vol. 4, no. 3, pp. 375–385, June 1996.

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

[41] "*ns-2* network simulator," `http://www.isi.edu/nsnam/ns/`.

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

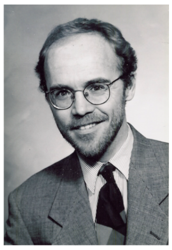
[43] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," IETF RFC 2581, Apr. 1999.

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

[45] N. Christin and J. Liebeherr, "Marking algorithms for service differentiation of TCP traffic," *Comp. Comm.*, vol. 28, no. 18, pp. 2058–2069, Nov. 2005.

[46] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," IETF RFC 3031, Jan. 2001.

Nicolas Christin (S'99, M'03) received the Engineering Degree from Ecole Centrale Lille, France, in 1999, and Master's and Ph.D. degrees in Computer Science from the University of Virginia in 2000 and 2003, respectively. In 2002-2003, he worked in Nortel's Advanced Technology group, From 2003 to 2005, he was a postdoctoral fellow in the School of Information Management and Systems at UC Berkeley. He is now a faculty in Carnegie Mellon University's Information Networking Institute, on international assignment in CyLab Japan. His research interests are in computer networks, network security, and network economics, and range from designing and evaluating formal models and algorithms to implementation aspects and measurements.



Jörg Liebeherr (S'88, M'92, SM'03) received the Ph.D. degree in Computer Science from the Georgia Institute of Technology in 1991. After a Postdoc at the University of California, Berkeley, he joined the Department of Computer Science at the University of Virginia in 1992. In 1997-1998 he was an Associate Professor in the Department of Electrical Engineering at Polytechnic University. Since Fall 2005, he is with the Department of Electrical and Computer Engineering of the University of Toronto as the Nortel Chair of Network Architecture and Services. He has served on editorial boards and program committees of several journals and conferences in computer networking. He was a Member-at-Large on the IEEE Communications Society Board of Governors in 2003-2005, and chair of the IEEE Communications Society Technical Committee on Computer Communications in 2004-2005.

Tarek Abdelzaher received his B.Sc. and M.Sc. degrees in Electrical and Computer Engineering from Ain Shams University, Cairo, Egypt, in 1990 and 1994 respectively. He received his Ph.D. from the University of Michigan in 1999. He has been an Assistant Professor at the University of Virginia, where he founded the Software Predictability Group. He is currently an Associate Professor at the Department of Computer Science, the University of Illinois at Urbana Champaign. He is Editor-in-Chief of the Journal of Real-Time Systems, an Associate Editor of the IEEE Transactions on Mobile Computing, the ACM Transaction on Sensor Networks, and the Ad Hoc Networks Journal, as well as Editor of ACM SIGBED Review. He authored more than 60 papers and served on numerous technical program committees in real-time computing, sensor networks, performance management, among others. He was Program Chair of RTAS 2004 and General Chair of RTAS 2005. He is currently Program Chair of RTSS 2006 and General Chair of IPSN 2007. Abdelzaher's research interests lie broadly in understanding and controlling the temporal properties of software systems in the face of increasing complexity, distribution, and degree of embedding in an external physical environment. Tarek Abdelzaher is a member of IEEE and ACM.