A Quantitative Assured Forwarding Service

Nicolas Christin, Jörg Liebeherr, and Tarek F. Abdelzaher
Department of Computer Science
University of Virginia
P.O. Box 400740
Charlottesville, VA 22904-4740, U.S.A.

Abstract— **The Assured Forwarding (AF) service of the IETF DiffServ architecture provides a qualitative service differentiation between classes of traffic, in the sense that a low-priority class experiences higher loss rates and higher delays than a high-priority class. However, the AF service does not quantify the difference in the service given to classes. In an effort to strengthen the service guarantees of the AF service, we propose a *Quantitative Assured Forwarding* service with absolute and proportional differentiation of loss, service rates, and packet delays. We present a feedback-based algorithm which enforces the desired class-level differentiation on a per-hop basis, without the need for admission control or signaling. Measurement results from a testbed of FreeBSD PC-routers on a 100 Mbps Ethernet network show the effectiveness of the proposed service, and indicate that our implementation is suitable for networks with high data rates.**

I. Introduction

The Assured Forwarding (AF, [1]) service of the Differentiated Services (*DiffServ*, [2]) architecture is an attempt to provide a scalable solution to the problem of service differentiation in the Internet. In the AF service, flows with similar QoS requirements are grouped into classes, using the DiffServ CodePoint field (DSCP, [3]) in the IP header. An attractive feature of the AF service is that it does not require admission control or per-flow classification, and is therefore scalable on both the control and data paths. However, the AF service only provides qualitative differentiation between classes, in the sense that some classes receive lower delays and a lower loss rate than others, but the differentiation is not quantified, and no absolute service bounds are offered.

Recently, research efforts have tried to strengthen the guarantees that can be provided within the context of the AF service without sacrificing its scalability and its simplicity, either by trying to quantify the difference in the level of service received by different classes, or by offering absolute bounds on service parameters, e.g., delays, to a specific set of classes. For instance, the *proportional service differentiation* model [4], [5] quantifies the difference in the service by making the ratios of delays or loss rates of different classes roughly constant. This type of service can be implemented through scheduling algorithms [4], [5], [6], [7], [8], [9], [10], [11] and/or buffer management algorithms [5], [12]. Recent works have tried to combine the scheduling and dropping decisions in a single algorithm [13], [14]. Most scheduling and/or buffer management algorithms aim at proportional differentiation, but do not support absolute service guarantees.

In a different approach to strengthening the AF service, the Alternative Best-Effort (ABE) service considers two traffic classes. The first class obtains absolute delay guarantees, and the second class has no delay guarantees, but is given a better loss rate than the first class. Scheduling and buffer management algorithms for the ABE service are presented in [15]. The service model in [16] also supports absolute delay bounds, and qualitative loss and throughput differentiation, but no proportional differentiation.

These recent efforts to strengthen the AF service raise questions on the best possible class-based service model that can be achieved by entirely relying on scheduling and dropping algorithms at routers, and without admission control, traffic policing, or signaling. In an attempt to explore the limits of such a class-based service, we define in this paper a "Quantitative Assured Forwarding" [1] service that offers, on a per-hop basis, both absolute and proportional guarantees to classes. Each node enforces any mix of absolute and proportional guarantees. Absolute guarantees apply to loss rates, delays, or throughput, and define a lower bound on the service received by each class. Proportional guarantees apply to loss rates and queueing delays. As an example of the guarantees in the Quantitative Assured Forwarding service for three classes of traffic, one could specify service guarantees of the form "Class-1 Delay $\leq$ 2 ms," meaning that no Class-1 packet should experience a queueing delay greater than two milliseconds, "Class-2 Delay $\approx$ 4·Class-1 Delay," meaning that Class-2 packets should experience queueing delays

This work is supported in part by the National Science Foundation through grants NCR-9624106 (CAREER), ANI-9730103, and ANI-0085955.

[1] The name "quantitative differentiated service" was recently used in [16].

roughly twice as large as Class-1 packets, "Class-2 Loss Rate $\leq$ 1%," meaning that the loss rate of Class 2 should never exceed 1%, "Class-3 Loss Rate $\approx$ 2·Class-2 Loss Rate," and "Class-3 Service Rate $\geq$ 1 Mbps," meaning that the aggregate of flows belonging to Class 3 should get a throughput of at least 1 Mbps. The Quantitative Assured Forwarding service supports any mix of such service guarantees, and the QoS parameters (e.g., delay bound of 1 ms) are configurable by the network operator. Clearly, without admission control, it is not feasible to satisfy all absolute guarantees at all times. Thus, when absolute constraints cannot be satisfied, we allow that some service guarantees can be temporarily relaxed according to a specified order.

We present a formal description of the Quantitative Assured Forwarding service, and we devise an algorithm that enforces guarantees on 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 the algorithm, and, to this effect, make assumptions which approximate the non-linearities in the system of study, similar to [17], [18], [19].

This paper is organized as follows. In Section II, we define the Quantitative Assured Forwarding service. In Sections III and IV, we describe the algorithms which provide the Quantitative Assured Forwarding service. In Section V, we present an implementation of these algorithms in FreeBSD PC-routers. We evaluate the algorithms using the implementation in Section VI, and present brief conclusions in Section VII.

II. The Quantitative Assured Forwarding Service

In this section, we describe the Quantitative Assured Forwarding Service, and outline a solution for an algorithm that realizes this service.

A. Formal Description

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 system, where events are traffic arrivals. We use $t(n)$ to denote the time of the n-th event in the current busy period[2], 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

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

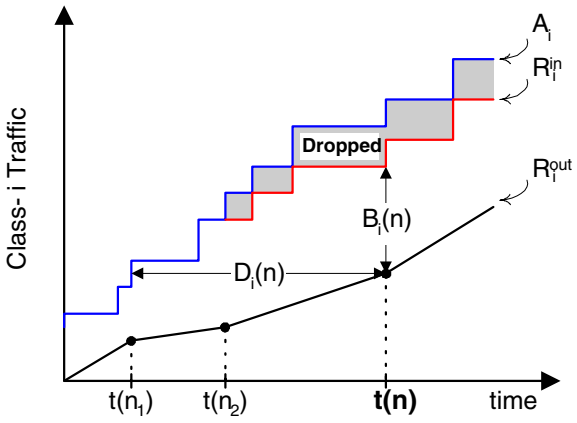


Fig. 1. Delay and backlog at the transmission queue of an output link. A_i is the arrival curve, R_i^{in} is the input curve and R_i^{out} is the output curve.

to class-i at the time of the n-th event. The service rate of a class i is a fraction of the output link capacity, which can vary over time, and is set to zero if there is no backlog of class-i traffic in the transmission queue. For the time being, we assume bursty arrivals with 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 V, we discuss how the fluid-flow interpretation is realized in a packet network.

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

The following presentation specifies the service differentiation independently for each busy period. Let $t(0)$ define the beginning of the busy period. The arrival curve of 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 en-

tered 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,

$$R_i^{out}(n) = \sum_{k=0}^{n-1} r_i(k)\Delta t(k) \ . \tag{3}$$

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

$$\begin{aligned} D_i(n) = \quad & t(n) \\ & -t\big(\max\{k < n \mid R_i^{out}(n) \geq R_i^{in}(k)\}\big) \ . \end{aligned} \tag{4}$$

Eqn. (4) 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

$$p_i(n) = \frac{A_i(n) - R_i^{in}(n)}{A_i(n)} \ . \tag{5}$$

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.

With these metrics, we can express the service guarantees of a Quantitative Assured Forwarding service. An absolute delay guarantee on class i is specified as

$$\forall n : D_i(n) \leq d_i \ , \tag{6}$$

where d_i is the delay bound of class i. Similarly, an absolute loss rate bound for class i is defined by

$$\forall n : p_i(n) \leq L_i \ . \tag{7}$$

An absolute rate guarantee for class i is specified as

$$\forall n : B_i(n) > 0 \ , r_i(n) \geq \mu_i \ . \tag{8}$$

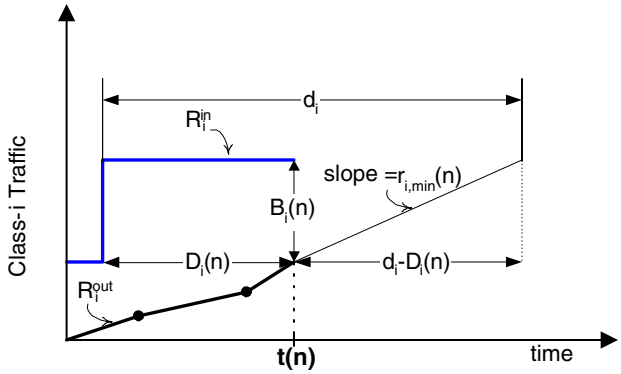


Fig. 2. Determining service rates for delay guarantees.

The proportional guarantees on delay and loss, respectively, are defined, for all n such that $B_i(n) > 0$ and $B_{i+1}(n) > 0$, as

$$\frac{D_{i+1}(n)}{D_i(n)} = k_i \ , \tag{9}$$

and

$$\frac{p_{i+1}(n)}{p_i(n)} = k_i' \ , \tag{10}$$

where k_i and k_i' are constants that quantify the proportional differentiation desired.

B. Rate Allocation and Drop Decisions

We now sketch a solution for providing the service guarantees specified in Eqs. (6)-(10) at the output link of a router with capacity C and buffer size B. We assume per-class buffering of incoming traffic, thus, each class is transmitted in a First-Come-First-Served manner. In the proposed solution, 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. (6)-(10) are met. If not all constraints in Eqs. (6)-(10) can be met at the n-th event, then some service guarantees need to be temporarily relaxed. We assume that the order in which guarantees are relaxed is given.

The absolute delay guarantee 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 if

$$r_i(n) \geq \frac{B_i(n)}{d_i - D_i(n)} \ .$$

This condition can be verified by inspection of Figure 2. If the condition holds for any n, the delay bound d_i is never violated. If class i has, in addition, an absolute rate

guarantee μ_i, the expression for the minimum rate needed by class i at the n-th event, becomes [3]

$$r_{i,min}(n) = \max\left\{\frac{B_i(n)}{d_i - D_i(n)}, \mu_i \cdot \chi_{B_i(n)>0}\right\} . \quad (12)$$

The service rate that can be allocated to class i is upper bounded by the output link capacity minus 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 differentiation.

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

$$r_i(n) = r_i(n-1) + \Delta r_i(n) , \quad (15)$$

where $\Delta r_i(n)$ is selected such that the constraints of proportional delay differentiation are satisfied at event n. From Eqs. (3) and (4), 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 thus determine the deviation from the desired proportional differentiation resulting from 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 meeting all delay guarantees exist 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. The loss guarantees determine which class(es) suffer(s) traffic drops at the n-th event.

To enforce loss guarantees, we rewrite the loss rate, defined by Eqn. (5), as a difference equation

$$p_i(n) = p_i(n-1)\frac{A_i(n-1)}{A_i(n)} + \frac{l_i(n)}{A_i(n)} . \quad (16)$$

From Eqn. (16), 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 violating absolute loss guarantees. In this set, we choose the class whose loss rate differs by the largest amount from the objective of Eqn. (9).

Having expressed the service rate and the loss rate in terms of a recursion, we can characterize the service rate allocation and dropping algorithm as feedback control problems. In the next sections, we will describe two feedback problems: one for delay and absolute rate differentiation ('delay feedback loop'), and one for loss differentiation ('loss feedback loop'). We describe the interaction of the two feedback problems in Section V.

[3] For any expression '$expr$', we define $\chi_{expr} = 1$ if '$expr$' is true and $\chi_{expr} = 0$ otherwise.

III. The Delay Feedback Loop

In this section, we present feedback loops which enforce the desired delay and rate differentiation given by Eqs. (6), (8), and (9). We have one feedback loop for each class with proportional delay guarantees. 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 stability conditions for the linearized control loop.

A. Objective

Let us assume for now that all classes are offered proportional delay guarantees. Later, this assumption will be relaxed. The set of constraints given by Eqn. (9) leads to the following system of equations:

$$\begin{aligned} D_2(n) &= k_1 \cdot D_1(n) , \\ &\vdots \\ D_N(n) &= \left(\textstyle\prod_{j=1}^{N-1} k_j\right) D_1(n) . \end{aligned} \quad (17)$$

Let $m_i = \prod_{j=1}^{i-1} k_j$ 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

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

By multiplying each line of Eqn. (17) with $\prod_{j\neq i} m_j$, we see that the desired proportional delay differentiation is achieved for all classes if

$$\forall i, j\ , \forall n : D_i^*(n) = D_j^*(n) . \quad (19)$$

Eqn. (19) is equivalent to

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

where

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

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 $\overline{D}^*(n)$.

Remark: We view event numbers, n, as sampling times on a virtual time axis in which events are equidistant. Hence,

convergence of the control loop applies to virtual time. However, the relationship between delay and rate is independent of the time axis chosen. By virtue of this independence, and since real-time is monotonically increasing with virtual time, we make the assumption that the skew between virtual-time and real-time can be neglected, and that the convergence condition we present later applies to real-time as well.

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.,

$$e_i(n) = \overline{D}^*(n) - D_i^*(n) \; . \tag{22}$$

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. (22), 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

$$\Delta r_i(n) = K(n) \cdot e_i(n) \; , \tag{24}$$

where $K(n) < 0$, which, in feedback control terminology, is the controller. An advantage of this controller is that it requires a single multiplication, and hence is easily implemented in a real system. Another advantage is that, at any n, we have

$$\sum_i \Delta r_i(n) = K(n) \sum_i e_i(n) = 0 \; . \tag{25}$$

Therefore, the controller produces 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, are undesirable for networks that need to achieve a high resource utilization.

We then express limits on $K(n)$ so that the feedback loops are stable. Let us define $\overline{r_i}(n)$ as the average service rate experienced by the class-i traffic departing at the n-th event over the time this class-i traffic was backlogged. Under the assumption that the backlog $B_i(n)$ does not change significantly during the time a particular traffic arrival is backlogged, we can write $D_i(n) \approx B_i(n)/\overline{r_i}(n)$. Further, if we can assume that changes to the average service rate, defined as $\Delta\overline{r_i}(n) = \overline{r_i}(n) - \overline{r_i}(n-1)$, are small compared to the average service rate, i.e., $\Delta\overline{r_i}(n) \ll \overline{r_i}(n)$, then we can approximate the effects of changes to the rate allocation on the changes to the delay by a linear relationship. We refer to [20] for details of these arguments.

With the above approximations, we can design $K(n)$ so that the feedback loop, composed of the controller and the effects of the service rate adjustment on the delay, is linear and time-invariant. We can then derive a stability condition on the worst-case of the approximate, linearized model.

The stability condition on the linearized approximate model, presented in detail in [20], results in the following condition

$$-2 \cdot \min_i \left\{ \frac{B_i(n)}{\prod_{j \neq i} m_j \cdot D_i^2(n)} \right\} \leq K(n) \leq 0 \; . \tag{26}$$

We emphasize that the assumptions made do not hold in general. Thus, while we cannot make any claim as to the stability of the delay feedback loops resulting from the analysis presented here, the numerical data in Section VI suggests that the loops converge adequately well.

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. (25). Hence, we use the following to compute $K(n)$ that would satisfy the saturation constraint

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

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

$$K(n) \geq \max_i \left(\frac{r_{i,min}(n) - r_i(n-1)}{e_i(n)} \right) \; . \tag{28}$$

If $\max_i \left(\frac{r_{i,min}(n) - r_i(n-1)}{e_i(n)} \right) > 0$, we see that we cannot have $K(n) < 0$. In other words, we cannot satisfy absolute delay and rate guarantees and proportional delay differentiation at the same time. In such a case, we relax either Eqn. (26) or (28) according to the given precedence order on the service guarantees.

Remark: If proportional delay differentiation is requested for some, but not for all classes, constraints as in Eqn. (17) can be defined for each group of classes with contiguous indices. Then, the feedback loops are constructed independently for each group.

IV. The Loss Feedback Loop

We now describe the feedback loop which controls the traffic dropped from a class i to satisfy proportional loss differentiation within the limits imposed by the absolute loss guarantees. As before, we assume that all classes have proportional loss guarantees. The assumption is relaxed similarly as described in the remark at the end of Section III.

Traffic must be dropped at the n-th event either if there is a buffer overflow or if absolute delay guarantees cannot be satisfied given the current backlog. To prevent buffer overflows at the n-th event, the following condition must hold:

$$B \geq \sum_{k=1}^{N} \big(B_k(n-1) + a_k(n) - l_k(n)\big) - \Delta t(n-1)C \; . \tag{29}$$

To provide absolute delay and rate guarantees, the following condition must be satisfied

$$C \geq \sum_{k=1}^{N} \max\left\{ \frac{B_k(n-1) - r_k(n-1)\Delta t(n-1)}{d_k - D_k(n)} + \frac{a_k(n) - l_k(n)}{d_k - D_k(n)}, \mu_k \cdot \chi_{B_k(n)>0} \right\} \; . \tag{30}$$

To choose the amount of traffic to drop from each class so that Eqs. (29) and (30) hold, we define the weighted loss rate to be

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

where $m_i' = \prod_{j=1}^{i-1} k_j'$ for $i > 1$ and $m_1' = 1$. With this definition, Eqn. (10) is equivalent to

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

We choose the following set point for the loss feedback loop

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

and we use the set point to describe an error

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

To reach the set point, the error is decreased 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 $\langle i_1, i_2, \ldots, i_R \rangle$.

Absolute loss guarantees 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. (7) and (16) as

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

If the conditions in Eqs. (29) and (30) 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. (29) and (30) 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 allow absolute delay and rate conditions to be violated. In other words, condition (30) 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, 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.

V. Implementation

We implemented the algorithms presented in Sections III and IV on PC-routers running the FreeBSD v4.3 [21] operating system, using the ALTQ v3.0 package [22]. ALTQ allows programmers to modify the operations of the transmission queue in the IP layer of the FreeBSD kernel. Our implementation is available to the public at `http://qosbox.cs.virginia.edu/software.html`. For a detailed discussion of the implementation issues, we refer the reader to [23]. In this paper, we will only discuss the operations performed in our implementation when a packet is entered into the transmission queue of an IP router (packet enqueueing) and when a packet is selected for transmission (packet dequeueing).

We use the DSCP field 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 testbed implementation, this is the first router traversed by a packet.

In our implementation, we chose the following precedence order for relaxing constraints. Absolute loss guarantees have higher precedence than absolute delay and

rate guarantees, which have in turn higher precedence than proportional guarantees.

A. Packet Enqueueing

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 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 IV to determine if and how much traffic needs to be dropped from each class. In our implementation, the algorithm of Section IV is run twice. The first time, buffer overflows are resolved by ignoring condition (30); The second time, violations of absolute delay and rate guarantees are resolved by ignoring condition (29).

Next, the `enqueue` procedure computes new values for $r_{i,min}(n)$ from Eqn. (12), and determines new service rates, using Eqs. (15) and (24), with the constraints on $K(n)$ given in Eqs. (26) and (28). If no feasible value for $K(n)$ exists, Eqn. (26) is ignored, thereby giving absolute delay guarantees precedence over proportional delay guarantees.

B. Packet Dequeueing

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. In our implementation, we use a modified Deficit Round Robin (DRR, [24]) 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 which is the most behind its theoretical output curve.

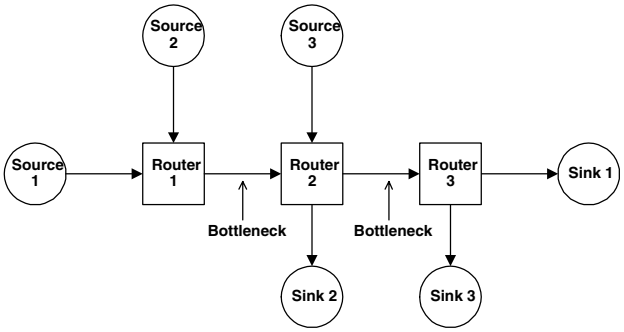


Fig. 3. Network Topology. All links have a capacity of 100 Mbps. We measure the service provided by Routers 1 and 2 at the indicated bottleneck links.

Class	Service Guarantees				
	d_i	L_i	μ_i	k_i	k'_i
1	8 ms	1 %	–	–	–
2	–	–	35 Mbps	2	2
3	–	–	–	2	2
4	–	–	–	N/A	N/A

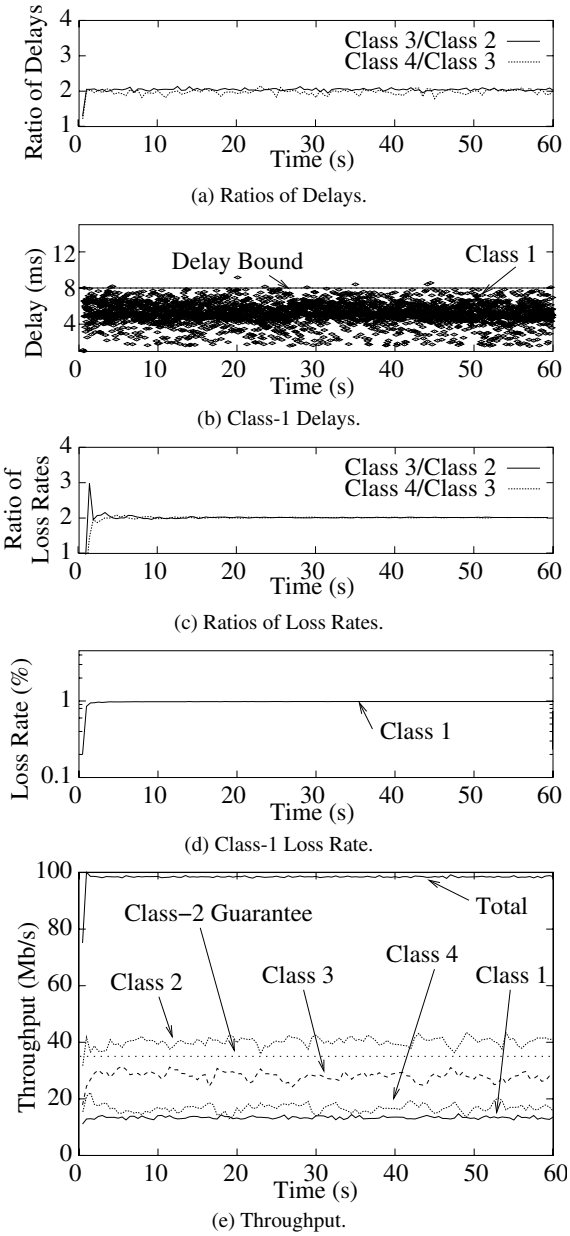
TABLE I
SERVICE GUARANTEES. THE GUARANTEES ARE IDENTICAL AT EACH ROUTER.

VI. EVALUATION

We present experimental measurements of our implementation of the Quantitative Assured Forwarding service on a testbed of PC routers. 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.

In our experiments we determine if and how well our algorithm provides the desired service differentiation on a per-node basis. In addition, we want to observe the stability of the feedback loops.

We use a local network topology using point-to-point Ethernet links as shown in Figure 3. All links are full-duplex and have a capacity of $C = 100$ Mbps. Three PCs are set up as routers, indicated in Figure 3 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.



(a) Ratios of Delays.

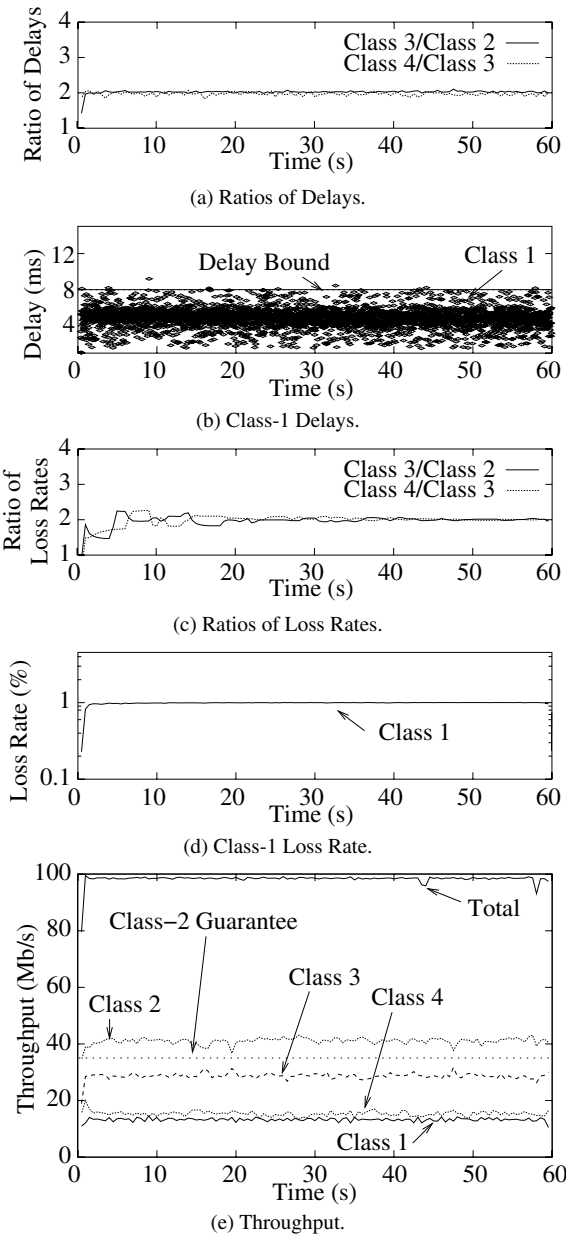
(b) Class-1 Delays.

(c) Ratios of Loss Rates.

(d) Class-1 Loss Rate.

(e) Throughput.

Fig. 4. 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.

(c) Ratios of Loss Rates.

(d) Class-1 Loss Rate.

(e) Throughput.

Fig. 5. Router 2. The graphs show the service obtained by each class at the output link of Router 2.

As mentioned earlier, the buffer size at the output link of each router is set to $B = 200$ packets.

We consider four traffic classes with service guarantees as summarized in Table I. Recall that all service guarantees are per-node guarantees. They are enforced independently at each router.

Sources 1, 2 and 3 send traffic to Sinks 1, 2 and 3, respectively. Each source transmits traffic from all four classes. 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 II. Each source transmits 6 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* v2.1pl3 tool [25]. The network load is initially zero and quickly ramps up to generate an overload at the bottleneck links of Figure 3. Congestion control at the TCP sources maintains the total load at a level of about 99% of the link capacity [20].

We measure the delay, the loss rate, and the through-

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 II

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 MS OFF-PERIOD. ALL TCP SOURCES ARE GREEDY, I.E., THEY ALWAYS HAVE DATA TO TRANSMIT, AND RUN THE *NewReno* CONGESTION CONTROL ALGORITHM.

put of each traffic class at the output links of Routers 1 and 2 (the links which go to the bottleneck links). Delays are measured as the waiting time of a packet in the transmission queue, i.e., as the difference of the times read of the machine clock when the packet enters and departs the transmission queue. Throughput and loss rates are obtained from reports generated every 0.5 sec by the OS kernel. In the plots, which summarize our measurements, we depict delay measurements of individual packets. Measurement of delay ratios, loss rates, ratios of loss rates and throughput are shown as averages over a sliding window of size 0.5 sec.

In Figures 4 and 5, we present our measurements of the service received at the bottleneck links of Routers 1 and 2, respectively. Figs. 4(a) and 5(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 I) is achieved. The plots indicate that the delay feedback loops appear to be stable, despite the simplified model we used for determining $K(n)$ in Section III.

In Figs. 4(b) and 5(b) we show the delay of Class-1 packets at Router 1 and Router 2. The delay bound of $d_1 = 8$ ms is satisfied, with few ($< 1.5\%$) exceptions at times when it is not possible to satisfy simultaneously absolute loss and delay guarantees; as discussed in Section V, such a conflict is resolved by giving precedence to the loss guarantee. Note that even if delay bounds are violated, no class-1 packet experiences a delay which exceeds 10 ms at either Router 1 or 2. Delay values, averaged over sliding windows of size 0.5 s, of other classes (not shown) are in the range 12-50 ms.

In Figs. 4(c) and (d), and Figs. 5(c) and (d), we show the measurements of the loss rates. Figs. 4(c) and 5(c) depict the ratios of loss rates for Classes 4 and 3, and for Classes 3 and 2. The desired ratios of $k_2' = k_3' = 2$ are maintained most of the time. Since the buffers at the routers are empty at the beginning of the experiment, there are no losses initially. As Figs. 4(d) and 5(d) indicate, the bound on the loss rates for Class 1 of $L_1 = 1\ \%$ is always kept (Recall that we give highest precedence to absolute loss guarantees.) We note that the maximum loss rate of classes 2–4 (not shown) is below 2% over the entire experiment.

Finally, in Figs. 4(e) and 5(e) we include the throughput measurements of all classes. We observe that the rate guarantee for Class 2 of $\mu_2 = 35$ Mbps is maintained. The total throughput of all classes, labeled in Figs. 4(e) and 5(e) as 'Total', is close to the link capacity of 100 Mbps at each router.

In summary, the measurement experiments of an overloaded network with multiple bottlenecks show that our feedback algorithms achieve the desired service differentiation, and utilize the entire available bandwidth, while maintaining stability throughout.

We present a brief evaluation of the overhead of the feedback-based algorithms. We have measured the number of CPU cycles consumed by the `enqueue` and `dequeue` procedures, by reading the timestamp counter register of the Pentium processor. We measured the average and standard deviation of the number of cycles over 500,000 packet transmissions on a heavily loaded link, using the topology and traffic pattern described in Figure 3 and Table II. We compare measurements for a set of four classes with constraints given in Table I, to a system of four classes without any guarantees. The measurements of the number of cycles, collected for Router 1, are shown in the following table.

Guarantees	Enqueue		Dequeue	
	Avg.	Std. Dev.	Avg.	Std. Dev.
with	15347	2603	4053	912
without	2415	837	3810	858

The table shows that the overhead for the `enqueue` operation, which implements the feedback algorithms, is significant. At the same time, the numbers indicate that a 1 GHz PC can enqueue and dequeue more than 50,000 packets per second. Considering that the average size of an IP packet on the Internet is $\overline{P} = 451.11$ bytes [26], this results in a maximum throughput of 186 Mbps.

VII. CONCLUSIONS

We presented the Quantitative Assured Forwarding service, which provides proportional differentiation on loss and delay and absolute service guarantees on loss, throughput and delay for classes of traffic. We proposed a feedback based algorithm for realizing the Quantitative Assured Forwarding service at a router. The algorithm does not require prior knowledge of traffic arrivals, and does not rely on signaling. At times when not all

absolute service guarantees can be satisfied simultaneously, the algorithm relaxes some of the guarantees by using a priority order. The algorithm has been implemented in FreeBSD PC-routers and the implementation is available at `http://qosbox.cs.virginia.edu/software.html`. Through experiments in a network of PC-routers, we showed that the proposed algorithm could fully utilize the available capacity of 100 Mbps. The measurements showed that the service guarantees of the Quantitative Assured Forwarding service are enforced. In ongoing work, we are conducting experiments for an empirical evaluation of the robustness of the proposed feedback algorithms, where we vary the network topology, the service guarantees, and the network load.

REFERENCES

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

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

[3] 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.

[4] C. Dovrolis, *Proportional differentiated services for the Internet*, Ph.D. thesis, University of Wisconsin-Madison, Dec. 2000.

[5] C. Dovrolis and P. Ramanathan, "Proportional differentiated services, part II: Loss rate differentiation and packet dropping," in *Proceedings of IWQoS 2000*, Pittsburgh, PA, June 2000, pp. 52–61.

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

[7] 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)*, Belek, Turkey, Oct. 1998, pp. 17–24.

[8] T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Barghavan, "Delay differentiation and adaptation in core stateless networks," in *Proceedings of IEEE INFOCOM 2000*, Tel-Aviv, Israel, Apr. 2000, pp. 421–430.

[9] S. Bodamer, "A scheduling algorithm for relative delay differentiation," in *Proceedings of the IEEE Conference on High Performance Switching and Routing (ATM 2000)*, Heidelberg, Germany, June 2000, pp. 357–364.

[10] L. Essafi, G. Bolch, and H. de Meer, "Dynamic priority scheduling for proportional delay differentiated services," Tech. Rep. TR-I4-01-03, University of Erlangen, Mar. 2001.

[11] H. Saito, C. Lukovszki, and I. Moldován, "Local optimal proportional differentiated services scheduler for relative differentiated services," in *Proceedings of Ninth IEEE International Conference on Computer Communications and Netowrks (ICCCN 2000)*, Las Vegas, NV, Oct. 2000, pp. 554–550.

[12] U. Bodin, A. Jonsson, and O. Schelen, "On creating proportional loss differentiation: predictability and performance," in *Proceedings of IWQoS 2001*, Karlsruhe, Germany, June 2001, pp. 372–386.

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

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

[15] 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`.

[16] R. R.-F. Liao and A. T. Campbell, "Dynamic core provisioning for quantitative differentiated service," in *Proceedings of IWQoS 2001*, Karlsruhe, Germany, June 2001, pp. 9–26.

[17] 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 2001*, Anchorage, AK, Apr. 2001, vol. 3, pp. 1726–1734.

[18] 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*, Mar. 2002, Special Issue on Control-Theoretical Approaches to Real-Time Computing. In press.

[19] 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*, Phoenix, AZ, Apr. 2001, pp. 615–624.

[20] N. Christin, J. Liebeherr, and T. F. Abdelzaher, "A quantitative assured forwarding service," Tech. Rep. CS-2001-21, University of Virginia, Aug. 2001, `ftp://ftp.cs.virginia.edu/pub/techreports/CS-2001-21.pdf`.

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

[22] K. Cho, "A framework for alternate queueing: towards traffic management by PC-UNIX based routers," in *Proceedings of USENIX '98 Annual Technical Conference*, New Orleans, LA, June 1998.

[23] N. Christin and J. Liebeherr, "The QoSbox: A PC-router for quantitative service differentiation in IP networks," Tech. Rep. CS-2001-28, University of Virginia, Nov. 2001, `ftp://ftp.cs.virginia.edu/pub/techreports/CS-2001-28.pdf`.

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

[25] 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`.

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