A scalable service architecture for providing strong service guarantees

Nicolas Christin and Jörg Liebeherr

Department of Computer Science, University of Virginia

ABSTRACT

For the past decade, a lot of Internet research has been devoted to providing different levels of service to applications. Initial proposals for service differentiation provided strong service guarantees, with strict bounds on delays, loss rates, and throughput, but required high overhead in terms of computational complexity and memory, both of which raise scalability concerns. Recently, the interest has shifted to service architectures with low overhead. However, these newer service architectures only provide weak service guarantees, which do not always address the needs of applications. In this paper, we describe a service architecture that supports strong service guarantees, can be implemented with low computational complexity, and only requires to maintain little state information. A key mechanism of the proposed service architecture is that it addresses scheduling and buffer management in a single algorithm. The presented architecture offers no solution for controlling the amount of traffic that enters the network. Instead, we plan on exploiting feedback mechanisms of TCP congestion control algorithms for the purpose of regulating the traffic entering the network.

Keywords: Service Architectures, Differentiated Services, Scheduling, Buffer Management

1. INTRODUCTION AND RELATED WORK

Since its creation in the early 1970s, the Internet has adopted a "best-effort" service, which relies on the following three principles: (1) No traffic is denied admission to the network, (2) all traffic is treated in the same manner, and (3) the only guarantee given by the network is that traffic will be transmitted in the best possible manner given the available resources, that is, no artificial delays will be generated, and no unnecessary losses will occur. The best-effort service is adequate as long as the applications using the network are not sensitive to variations in losses and delays (e.g., electronic mail), the load on the network is small, and if pricing by network providers is not service-based. These conditions held in the early days of the Internet, but do not hold anymore due to the increasing number of different applications using the Internet, and to the fact that the Internet has become a source of commercial profit since the mid-1990s, creating a need for revenue maximization.[1]

The issues for providing different levels of services in the network are referred to as providing *Quality-of-Service* (QoS) guarantees. One could think that QoS issues can be resolved by increasing the capacity of the network. Unfortunately, the QoS received by an end-to-end traffic flow is bounded by the QoS received at the link with the smallest capacity (i.e., bottleneck) on the end-to-end path. Thus, augmenting the capacity of some links only moves the bottleneck to another part of the network, and consequently, only changes the location of the problem. For instance, even when the core of the network is overprovisionned, and is therefore under-utilized,[2] bottlenecks, and hence degradation of the QoS, can be observed at the edges of the network.

In fact, the recent explosion of link capacity in the network, instead of solving the problem of providing service guarantees, has put more stringent requirements on QoS architectures. Routers in the core, and even at the edges, of a packet network now have to serve millions of concurrent flows at gigabit per second rates, which induces two scalability requirements. First, the state information kept in the routers for providing QoS should be small. Second, the processing time for classifying and scheduling packets according to their QoS guarantees should be small as well, even with the advent of faster hardware.

A number of service architectures for packet networks have been devised in the last decade to try to provide a solution to the service differentiation problem, while addressing these scalability requirements. Building on the initial work of D. Ferrari and the Tenet group at UC Berkeley,[3] the IETF proposed the Integrated Services (*IntServ*) architecture[4] as a QoS architecture for IP networks. IntServ, developed in the early and mid-1990s, provides the ability to give individual flows *absolute* QoS guarantees on packet delays, in terms of upper bound on delays, and packet losses (no loss), as long as the traffic of each flow conforms to a pre-specified set of parameters. Solutions devised for implementing IntServ require per-flow classification at each router, meaning that, at each router, each incoming packet has to be inspected to determine which service guarantees it must receive. Each router therefore has to keep per-flow state information, which raises concerns regarding the scalability of the IntServ architecture. Additionally, IntServ implementations require complex packet scheduling primitives, which run a dynamic priority sorting algorithm. This can cause scalability problems if the proposed algorithms have to process a large number of packets within a short period of time. Due to the unresolved issues regarding its scalability, IntServ has not gained wide acceptance.

In the second half of the 1990s, the interest in QoS shifted to architectures that make a distinction between operations performed in the the network core, and operations performed at the edges of the network. The basic idea is that the amount of traffic in the network core does not permit complex QoS mechanisms, and that most of the QoS mechanisms should be executed at the network edge, where the volume of traffic is smaller.

These recent efforts resulted in the Differentiated Services (*DiffServ*) architecture[5] which bundles flows with similar QoS requirements in *classes* of traffic. The mapping from individual flows to classes of traffic is determined at the edges of the network, where computational resources are less scarce than in the core. In the core of the network, scheduling primitives only work with a few classes of traffic, and can thus remain relatively simple. DiffServ currently offers two different types of service in addition to Best-Effort: an Assured Forwarding (AF) service[6] and an Expedited Forwarding (EF) service.[7] The Assured Forwarding service provides *qualitative* differentiation among the so-called AF classes. For instance, for two AF classes of traffic denoted by class A and B, Assured Forwarding can specify that class B traffic must be dropped more aggressively than class A in times of congestion. Such a qualitative service architecture can be implemented with simple algorithms and does not require per-flow classification or signaling. However, since the AF service only provides qualitative relative guarantees, the service assurance is limited, compared to the IntServ architecture for instance. Conversely, the Expedited Forwarding service offers absolute service guarantees on delay variations[7] for

classes of traffic. In essence, providing the EF service to a flow is equivalent to providing a virtual leased-line to this flow, which leads to low resource utilization. Thus, it is envisioned that the EF service can only apply to a very limited number of flows.[8] More recently, some research studies have aimed at strengthening the service assurance provided by qualitative relative service models such as the AF service. For instance, the *Proportional Differentiated Services* model[9] defines a service model with no admission control where the ratios of loss ratios and packet delays between successive priority classes remain roughly constant (*proportional* service guarantees).
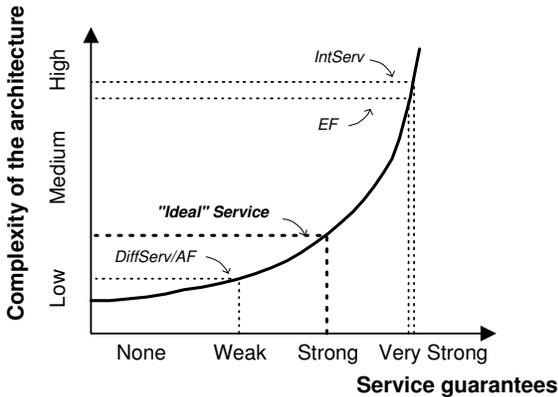

Figure 1. The trade-off between strength of service guarantees and complexity of the implementation. IntServ provides very strong service guarantees at the price of a very high complexity, while DiffServ only provides limited service assurance, but has low complexity. The "ideal" service should be able to provide strong service differentiation with a low complexity. Note that the picture is qualitative.

From these previous research efforts, it appears that a trade-off between simplicity of the implementation and strength of service guarantees has to be made, as shown in Figure 1. On the one hand, IntServ and Expedited Forwarding are too complex to be realized on a large-scale network. On the other hand, the AF service of the DiffServ model only support weak QoS guarantees. As shown in Figure 1, an "ideal" service architecture, that is, a service architecture which can deployed and used on a large network, should provide strong service guarantees with limited complexity. To this date, the most significant advance in devising such an ideal service is probably the SCORE architecture, proposed by Stoica and Zhang,[8] and architectures derived from it.[10,11] SCORE tries to reconcile the strength of the IntServ guarantees with the simplicity of the DiffServ architecture, by moving the state information needed to provide IntServ-like service guarantees from network routers to IP packets. SCORE uses this approach to provide end-to-end delay guarantees to flows without requiring per-flow state information at network routers. The basic idea for meeting end-to-end delay requirements is to keep track of the delays experienced by packets along the path from the source to the destination, by storing the values of the experienced delays in the packet headers. The stored information is used for adjusting the priority of packets so that end-to-end requirements are met. However, this approach has two drawbacks. It does not alleviate the need for packet classification, which can be computationally expensive, and it requires a change in the format of the IP header, which raises questions on the efforts needed to deploy the service.

In this paper, we discuss an alternative design architecture that provides strong guarantees and low complexity. We therefore present a new point in the design space described in Figure 1, with low computational complexity, and strong service guarantees. The service we propose is a per-hop, per-class service, which does not immediately translates into end-to-end, per-flow service guarantees. However, we point out that a per-hop, per-class service architecture such as described in this paper can be used to build end-to-end service guarantees, for instance if the end applications are in charge of dynamically selecting which class of traffic they require.[12]

To achieve the goal of providing strong service guarantees with a low complexity, we believe that one needs to revisit the current tenets of Internet QoS. We note that, until very recently, scheduling and buffer management were treated as orthogonal concepts. However, scheduling and buffer management both address the issue of managing a transmission queue at a given router. The only difference between the two concepts lies in the fact that scheduling manages the head of the transmission queue, deciding which packet will leave next, while buffer management manages the end of the transmission queue, deciding if new packets can be admitted to the queue. Thus, we will show that considering buffer management and scheduling as the same problem enables us to provide strong service guarantees within a simple and scalable architecture. More precisely, we have worked on mechanisms that conjointly address packet dropping and service rate allocation, from which packet scheduling immediately follows.

Additionally, mechanisms for providing QoS guarantees have to work in concert with end-to-end mechanisms, such as TCP feedback mechanisms for congestion control and avoidance.[13] However, to the best of our knowledge, with the exception of RIO,[14] which builds on the Random Early Detection algorithm, RED,[15] in an effort to reduce packet drops,

none of the proposed service architectures takes into account the feedback capabilities of TCP traffic. Traffic regulation is generally realized by admission control mechanisms or traffic policers, which are separate from the scheduling and dropping algorithms. Here, we make the hypothesis that exploiting TCP feedback mechanisms, coupled with Explicit Congestion Notification, ECN,[16, 17] to regulate the traffic arrivals by marking packets "smartly" will alleviate the need for admission control, and will allow us to design a simple, yet effective at providing strong guarantees, service architecture.

The remainder of this paper is organized as follows. In Section 2, we describe the service architecture we envision for providing a scalable service with strong guarantees. In Section 3, we illustrate the potential of the proposed approach with the Joint Buffer Management and Scheduling (JoBS) framework. Last, we draw brief conclusions and outline our current research plan in Section 4.

2. SERVICE ARCHITECTURE

In this section, we sketch our proposed solution to the problem of providing strong service guarantees in a scalable manner. We introduce our approach by identifying three key components of a class-based service architecture. The first component is the definition of the set of QoS guarantees offered by the service architecture. The second component addresses the issue of managing the transmission queue of a router in order to enforce the desired service guarantees. The third component is the regulation of traffic arrivals. If the first two components are common to all QoS architectures, the third component in a class-based architecture significantly differs from its counterpart in per-flow architectures. In per-flow QoS, each flow is described by a set of parameters (e.g., maximum burst size, peak sending rate), which requires that a lot of state information be maintained if a large number of flows are present in the network. This set of parameters, or flow profile, serves as the basis for traffic regulation. Two approaches are possible: Either flows that are not conforming to their profile are policed (by dropping traffic) to satisfy to the specified profile, or, for each flow, a binary decision is performed at the ingress node (i.e., where the flow enters the network) to determine if, according to its profile, the flow can be admitted to the network and meet its service guarantees. Note that the two approaches are in fact complementary. In general, each flow which is granted admission to the network is policed if it exceeds its specified profile. As stated above, using per-flow descriptors imposes limit on the scalability of per-flow architectures, due to the amount of state information that has to be stored. Additionally, the computational complexity of checking whether each flow conforms to its profile, and possibly policing it, can be rather high if a large number of flows are present in the network. In a class-based model such as the one we propose, we do not rely on per-flow descriptions, in order to alleviate the aforementioned scalability and complexity concerns. In fact, we take a radically different approach by not making any assumption on traffic arrivals. Instead, we use the capabilities of the network (i.e., TCP feedback mechanisms, ECN) to dynamically regulate traffic arrivals. We next describe separately the approach we propose for each of the three components in more details.

2.1. Service Guarantees

The first component of a class-based service architecture is the set of proposed service guarantees. Our goal is to provide a set of service guarantees that can encompass all of AF, Proportional Differentiated Services, and other class-based services (e.g., ABE[18]). More generally, we want to be able to enforce *any* mix of absolute and proportional guarantees at each participating node (i.e., router). We refer to this service as "Quantitative Assured Forwarding" service.[19] 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", "Class-2 Delay $\approx$ 4·Class-1 Delay", "Class-2 Loss Rate $\leq$ 1%", "Class-3 Loss Rate $\approx$ 2·Class-2 Loss Rate", and "Class-3 Service Rate $\geq$ 1 Mbps". We require that the algorithms used to implement the service be independent of the choice of specific bounds or proportional coefficients for each class. Our proposed service can thus be viewed as a generalization of all previous research efforts on class-based services, and should therefore be represented by a vertical line in the "Strong" domain in Figure 1.

2.2. Scheduling and Dropping

The second component is the scheduling and dropping algorithms required at each router to enforce the desired service guarantees. To ensure scalability and high utilization, these algorithms should not rely on admission control, signaling or traffic policing. We will express the provisioning of per-class QoS within a formal framework inspired by Cruz's

service curves.[20] Using this approach, we will present a reference algorithm called JoBS (Joint Buffer Management and Scheduling), combining scheduling and dropping, which is capable of supporting a wide range of relative, as well as absolute, per-class guarantees for loss and delay, without assuming admission control or traffic policing. The reference algorithm operates as follows. After each arrival, the algorithm predicts delays of the currently backlogged traffic, and then adjusts the service rate allocation to classes and the amount of traffic to be dropped so as to meet the specified service guarantees. A unique feature of the algorithm is that it considers scheduling and buffer management (dropping) together in a single step. The algorithm allocates service rates and drops packets according to the solution to a non-linear optimization problem. While the performance of the reference algorithm with respect to satisfying the service guarantees is excellent, its computational complexity prohibits its implementation at high data rates. To address this issue, we believe that application of linear feedback control theory can yield an efficient heuristic approximation of the reference algorithm that can be implemented at line speeds.

2.3. Regulating Traffic Arrivals

The third component is the end-to-end mechanism in charge of the regulation of traffic arrivals. Clearly, without admission control, it is not feasible to satisfy all absolute guarantees at all times without regulating the traffic arrivals. Admission control makes traffic arrivals conform to a set of parameters that ensure that the specified service guarantees are feasible. One of the main drawback of admission control (or traffic policing) is that the set of parameters is generally statically configured, thereby resulting in an inefficient utilization of the network resources. *

Different from existing approaches which either use admission control to regulate traffic arrivals, or perform traffic policing by means of shapers such as leaky-buckets, we will explore the possibility of considering the violation of QoS guarantees as a *congestion control* problem rather than an *admission control* problem. Congestion control has been extensively studied in the context of TCP traffic,[13, 15–17] which accounts for more than 90% of the total traffic on the Internet.[23] TCP provides feedback mechanisms that permit to resolve congestion problems, by reducing the sending rate of a source when an acknowledgment packet is marked with the ECN bit, or when a packet is dropped. However, to the best of our knowledge, with the exception of RIO, which only provides qualitative loss differentiation, feedback capabilities of TCP traffic and correlation between ECN-marked packets (or packet losses if ECN is not available) and sending rates have not been exploited in existing service architectures. We believe that using TCP feedback mechanisms and ECN for dynamically regulating traffic arrivals can an efficient alternative to admission control or traffic policing.

3. JOINT BUFFER MANAGEMENT AND SCHEDULING

We briefly present in this section the Joint Buffer Management and Scheduling (JoBS) algorithm,[19, 24] which is the core of our service architecture. The work on JoBS, and the discussion in this section, do not address the use of the feedback capabilities of TCP traffic, however, the results that we are presenting in this section should emphasize that the approach of combining buffer management and scheduling in a single algorithm is a possible solution to the problem of providing strong service guarantees in a scalable manner. For the discussion in this section, we assume that no traffic regulation is performed, and that at times when not all service guarantees can be met, some service guarantees are temporarily relaxed. We defer the discussion about traffic regulation to Section 4.

JoBS provides a framework for reasoning on per-class service guarantees, and formulates the realization of these service guarantees as an optimization problem. The discussion in this section elaborates on the first two components of a scalable service architecture, specified in the previous section. We will then turn to a brief description of a heuristic approximation of the optimization problem, that can be implemented in high-speed routers.

3.1. A Framework for Scalable Service Guarantees

We consider an individual router in the network, and propose to provide per-hop, per-class guarantees. We assume that all traffic that arrives to the transmission queue of the output link of this 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, whose beginning is

---

*This problem of under-utilization can be addressed by statistical[21] or measurement-based[22] admission control but neither of these techniques alleviates the problem of storing state information regarding which flows are/are not admitted to the network.

defined as the last time the transmission queue at the output link was empty, 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 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 presentation of this framework, 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. On the other hand, scheduling algorithms that emulate fluid-flow schedulers with rate-guarantees are readily available.[25]

Figure 2. Delay and backlog at the transmission queue of an output link. A_i is the arrival curve, R_i^{in} is the input curve, R_i^{out} is the output curve, $D_i(n)$ is the delay, and $B_i(n)$ is the backlog.

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 at 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 $R_i^{out}(n) = \sum_{k=0}^{n-1} r_i(k)\Delta t(k)$. In Fig. 2, we illustrate the concepts of arrival curve, A_i, input curve, R_i^{in} and output curve, R_i^{out} 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. 2, 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 $D_i(n) = t(n) - t\big(\max\{k < n \mid R_i^{out}(n) \geq R_i^{in}(k)\}\big)$. This last equation 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, $p_i(n) = \frac{A_i(n) - R_i^{in}(n)}{A_i(n)}$. 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.

Service Guarantees 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 $\forall n : D_i(n) \leq d_i$, where d_i is the delay bound of Class i. Similarly, an absolute loss rate bound for Class i is defined by $\forall n : p_i(n) \leq L_i$. An absolute rate guarantee for Class i is specified as $\forall n : B_i(n) > 0, r_i(n) \geq \mu_i$.

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 $\frac{D_{i+1}(n)}{D_i(n)} = k_i$, and $\frac{p_{i+1}(n)}{p_i(n)} = k_i'$, where k_i and k_i' are constants that quantify the proportional differentiation desired.
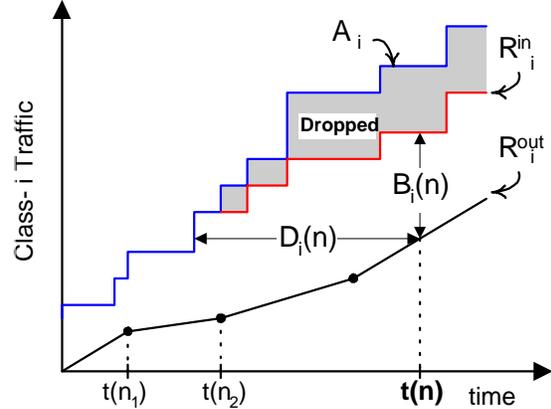
3.2. Service Rate Allocation and Packet Dropping: An Optimization Problem

With the framework just defined, we see that enforcing the desired service guarantees is a matter of selecting the service rate allocation to classes and the packet drops to be performed at each event n. We showed that the service rate allocation and packet drops could be viewed in terms of a non-linear optimization problem,[24] which we summarize here.

Each event n, when an arrival occurs, a new optimization is performed. The optimization variable $\mathbf{x}_n$ is a vector that contains the service rates $r_i(n)$ and the amount of traffic to be dropped $l_i(n)$,

$$\mathbf{x}_n = (r_1(n)\ldots r_N(n)\, l_1(n)\ldots l_N(n))^T \ .$$

The optimization problem has the form

$$\textbf{Minimize} \quad F(\mathbf{x}_n) \quad \textbf{Subject to} \quad \begin{aligned} g_j(\mathbf{x}_n) &= 0, \quad j = 1,\ldots,M \\ h_j(\mathbf{x}_n) &\geq 0, \quad j = M+1,\ldots,Q, \end{aligned}$$

where $F(.)$ is an objective function, and the g_k's and h_k's are constraints, which are discussed next.

Objective Function   Even though the choice of the objective function is a policy decision, we select two specific objectives, which - we believe - have general validity: (1) Avoid dropping traffic, and (2) avoid changes to the current service rate allocation. The first objective ensures that traffic is dropped only if there is no alternative way to satisfy the constraints. The second objective tries to hold on to a feasible service rate allocation as long as possible. We give the first objective priority over the second objective. Provided that the constraints can be satisfied, the chosen objective function F will select a solution for $\mathbf{x}_n$.

Constraints   The constraints of the optimization problem are system constraints, which describe physical limitations and properties of the output link, and QoS constraints, which define the desired service differentiation. The optimization at event n is done with knowledge of the system state before event n, that is, the optimizer knows R_i^{in} and R_i^{out} for all events $m < n$, and A_i for all events $m \leq n$.

The first system constraint states that the total backlog cannot exceed the buffer size B, that is, $\sum_i B_i(n) \leq B$ for all events n. The second system constraint enforces that scheduling at the output link is work-conserving. At a work-conserving link, $\sum_i r_i(n) = C$ holds for all events n where $\sum_i B_i(n) > 0$. This constraint is stronger than the limit given by the link capacity C, i.e., $\sum_i r_i(n) \leq C$. Other system constraints enforce that transmission rates and loss rates are non-negative. Also, the amount of traffic that can be dropped is bounded by the current backlog. Hence, we obtain $r_i(n) \geq 0$ and $0 \leq l_i(n) \leq B_i(n)$ for all events n.

The QoS constraints result from the service guarantees we want to offer, as described in the previous subsection. QoS constraints for classes that are not backlogged are simply ignored. While the absolute loss and throughput guarantees can be directly used to define some constraints of the optimization problem, the delay guarantees need to be adapted. Indeed, the delay metric $D_i(n)$ characterizes the delay of the Class-i traffic that is *leaving* the transmission queue at event n, and thus, there is no direct relationship between $r_i(n)$ and $D_i(n)$. To include the delay guarantees in the optimization problem, the scheduler makes a projection of the delays of all backlogged traffic. We point out that the mapping between the functions g_k's and h_k's and $\mathbf{x}_n$ is not straightforward and we refer the reader to a related paper.[24] If not all constraints can be satisfied at the same time, a precedence order on the "importance" of the constraints is used to temporarily relax some constraints.

This concludes the description of the optimization process in JoBS. The structure of constraints and objective function makes this a *non-linear optimization problem*, which can be solved with available numerical algorithms.[26]

3.3. A Feedback-Based Heuristic Algorithm

The optimization model described above is computationally expensive, and thus, this solution is impractical for an implementation at high speeds. In an effort to provide a scheme that can be implemented at high data rates, we devised feedback control algorithms that approximate the optimization described above.[19, 27] As in the optimization model, 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 in Subsection 3.2 are met. We characterized the service rate allocation and dropping algorithm as two separate feedback control problems, which we summarize in this section.
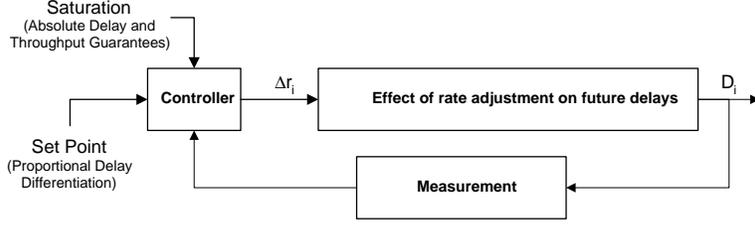
Figure 3. Overview of the feedback-based approach. The figure is an overview of the delay feedback loop for Class i. There is one such loop per class.

Service Rate Allocation   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. The objective of the feedback loops is to enforce the desired delay and rate differentiation given in Subsection 3.2.

We view the computation of $r_i(n)$ in terms of a recursion $r_i(n) = r_i(n-1) + \Delta r_i(n)$, where $\Delta r_i(n)$ is selected such that the constraints of proportional delay differentiation are satisfied at event n. 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. Thus, using control theory terminology, the function f is the *controller*.

We illustrate the feedback loop in Fig. 3. The set point of the controller is defined by the proportional delay differentiation desired, and the action of the controller is limited by saturation constraints which translate limitations on the rate allocated to Class i due to system constraints, and absolute delay and rate guarantees. The rate adjustment $\Delta r_i(n)$ has then an effect on the delays $D_i(k)$ $(k > n)$. The delays of traffic leaving the system are measured and compared to the set point upon each packet arrival.

In a previous paper,[19] we approximated the effect of $\Delta r_i(n)$ on $D_i(k)$ $(k > n)$ by a set of linear relationships, making the assumptions that the backlog $B_i(n)$ does not change significantly during the time a particular traffic arrival is backlogged, and that the Class-i delays do not vary significantly between two events. We then performed an analysis of the linearized control loop, and showed that the adjustment $\Delta r_i(n) = K(n) \cdot e_i(n)$, where $e_i(n)$ represents the distance between the set point and the delay of Class i, and $K(n)$ is a time-dependent proportional coefficient, produced the desired differentiation on delay and rate, provided that $K(n)$ satisfied to certain constraints.[19]   The adjustment $K(n) \cdot e_i(n)$ is simple enough to be implemented at high speeds, and satisfies the work-conserving property, as long as $\sum_i r_i(0) = C$.

Packet Dropping   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 classes suffer traffic drops at the n-th event. To enforce loss guarantees, we rewrite the loss rate, as a difference equation $p_i(n) = p_i(n-1)\frac{A_i(n-1)}{A_i(n)} + \frac{l_i(n)}{A_i(n)}$. From this difference equation, 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. We define the Class-i error $e'_i(n)$ as the distance between the value the Class-i loss rate should have according to the proportional loss guarantees and its actual value. We then use $\langle i_1, \ldots, i_R \rangle$ as an ordering of the class indices from all backlogged classes, that is, $B_{i_k}(n) > 0$ for $1 \le k \le R$, such that $e'_{i_s}(n) \ge e'_{i_r}(n)$ if $i_s < i_r$. We drop traffic in the order of $\langle i_1, i_2, \ldots, i_R \rangle$ to satisfy to the proportional loss guarantees. For each Class i, we stop dropping traffic when either (1) the loss guarantee L_i is reached, or (2) the buffer overflow is resolved or a feasible rate allocation for absolute guarantees exists and there is no need for dropping traffic anymore.

3.4. Evaluation

In this subsection, we first compare by simulation the performance of the optimization-based algorithm with the performance of the feedback-based algorithm, and show that the performance degradation is negligible. Secondly, we show, using experimental results gathered on a testbed of PC-routers, that the feedback-based algorithm is efficient at providing the desired service guarantees in an efficient manner. Thirdly, we summarize measurements of the implementation overhead of the feedback-based algorithm.
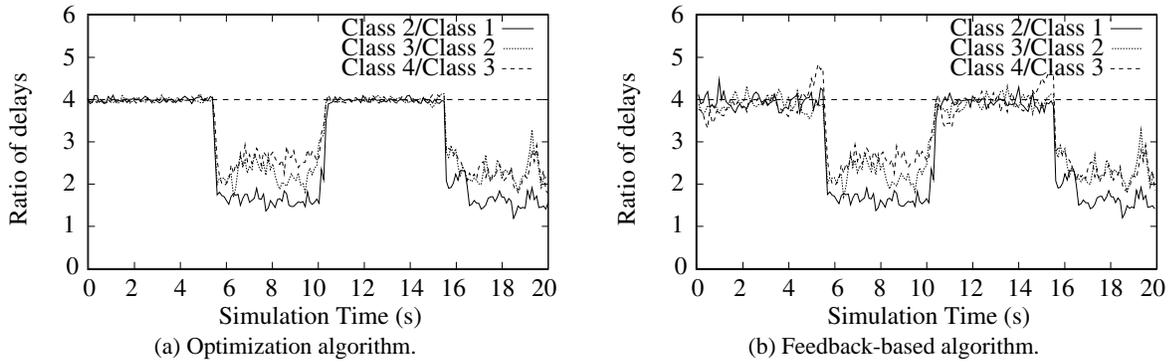
8

(a) Optimization algorithm.   (b) Feedback-based algorithm.

Figure 4. Experiment 1: Relative Delay Differentiation. The graphs show the ratios of the delays for successive classes. The target value is $k = 4$.

(a) Optimization algorithm.   (b) Feedback-based algorithm.
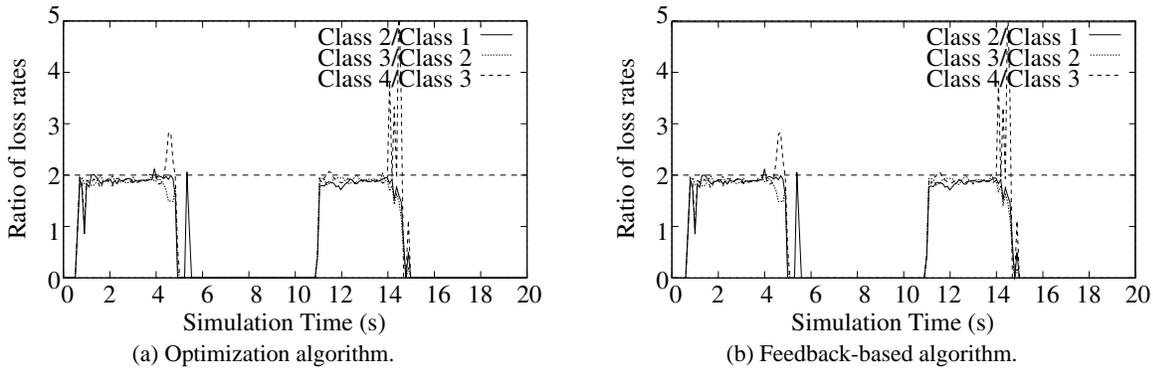
Figure 5. Experiment 1: Relative Loss Differentiation. The graphs show the ratios of loss rates for successive classes. The target value is $k' = 2$.

Experiment 1: Comparison of the Optimization-Based and Feedback-Based Algorithms   We compared the performance of the optimization-based approach and of the feedback-based algorithm by performing a single-node simulation with highly variable offered load, represented in Fig. 6, oscillating between 70% and 140% of the output link capacity. The major difference between the optimization-based algorithm and the feedback-based algorithm lies in the way proportional service differentiation is handled, since the saturation constraints of the feedback-based algorithm and the absolute constraints of the optimization-based approach are equivalent. Therefore, we focus our comparison on proportional service differentiation. Additional simulation results, including a comparison with other algorithms proposed for proportional service differentiation, can be found in a related paper.[24]
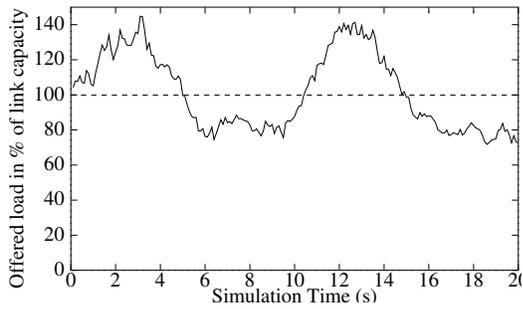

Figure 6. Offered Load. The plot represents the offered load we consider in our evaluation of JoBS.

In Fig. 4, we present ratios of delays, averaged over a sliding window of size 0.1 seconds for both algorithms. There are four classes of traffic, with $k_1 = k_2 = k_3 = 4$, and no absolute delay bounds. We see that the performance of both algorithms with respect to realizing the desired service guarantees is comparable. At high loads, the desired differentiation is achieved, and at low loads, neither of the algorithms can provide the target ratio, due to the work-conserving constraint. However, both algorithms still manage to provide some differentiation between classes, even at low loads. In Fig. 5, we compare ratios of loss rates. The target ratio is $k'_1 = k'_2 = k'_3 = 2$, and there are no absolute loss rate bounds. We see that both plots are almost identical, and thus, the performance of both algorithms with respect to proportional loss differentiation is even closer than in the case of proportional delay differentiation.
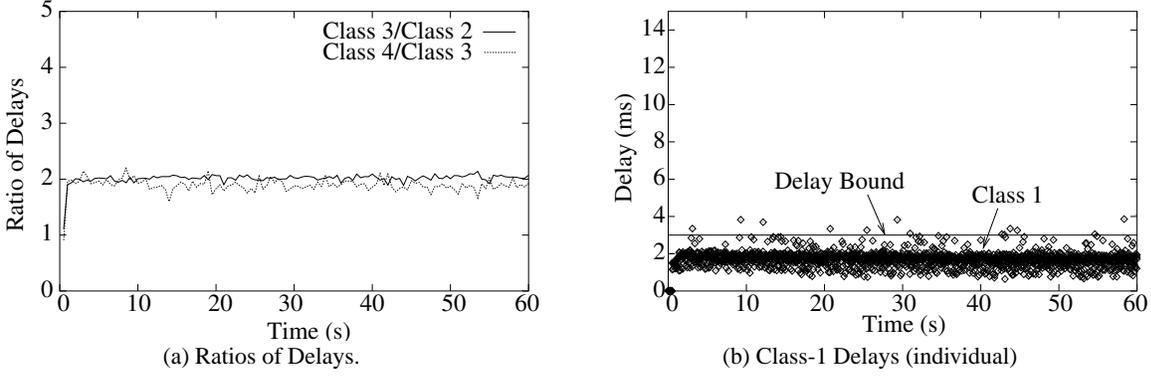
9

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

Figure 8. Experiment 2: Delay Differentiation. The graphs show the ratios of delays averaged over a moving window of size 0.1 s, and the individual delays encountered by all Class-1 packets.

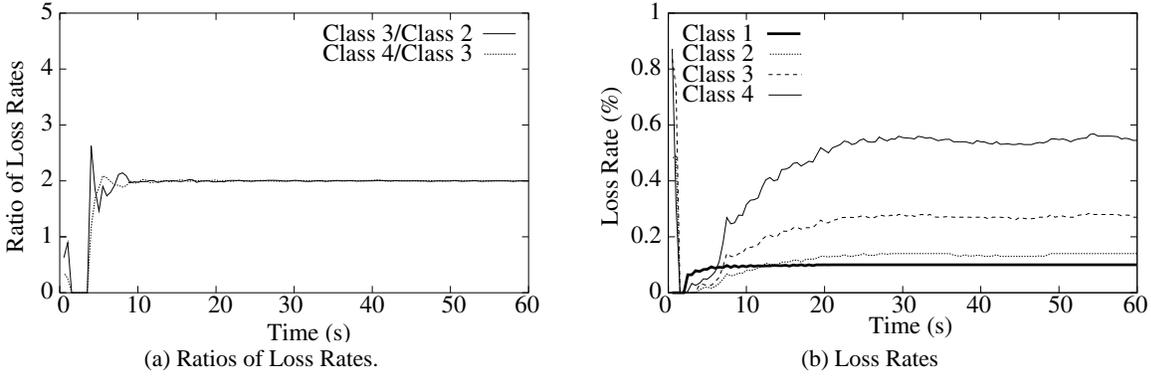(a) Ratios of Loss Rates.
(b) Loss Rates

Figure 9. Experiment 2: Loss Differentiation. The graphs show the ratios of loss rates and the loss rates encountered by each class. Results are averaged over a moving window of size 0.1 s.

Experiment 2: Evaluation by Implementation We next evaluate the effectiveness of JoBS in providing absolute service guarantees. We implemented the feedback-based algorithm in 1 GHz Pentium-III PC routers running FreeBSD 4.3[28] and ALTQ 3.0.[29] Comprehensive details about the implementation issues, and complementary measurements can be found in an associated technical report.[27] We point out that this implementation is now available as part of the standard ALTQ 3.1 distribution.

We ran a single-node experiment with four classes of traffic. The buffer size is set to $B = 200$ packets, and the PC-routers we use in this experiment are on a FastEthernet network, thus have a link capacity $C = 100$ Mbps. Class-1 traffic consists of 6 UDP flows, following an on-off Pareto traffic pattern consisting of bursts of 25 packets on average during an on-period, and of an off-period of 150 ms on average, while traffic for the three other classes consists of 10 greedy TCP flows for each class. The offered load is almost constant and equal to 100 % of the link capacity. The following absolute guarantees apply to Class 1: $d_1 = 3$ ms, $L_1 = 0.1$ %. Class 2 is provided a throughput guarantee of $\mu_2 = 35$ Mbps, and Classes 2, 3 and 4 are provided proportional service guarantees with $k_2 = k_3 = k'_2 = k'_3 = 2$. We

Figure 7. Experiment 2: Throughput differentiation. Class 2 is provided with a guarantee $\mu_2 = 35$ Mbps.

present our results in Figs. 7, 8, and 9. Fig. 7 shows that the throughput guarantee on Class 2 is always respected, and that the feedback-based algorithm is efficient enough to transmit traffic at the link speed. We plot the ratios of delays, and individual packet delays in Fig. 8, and see that, except for a small number of packets (< 1 %) which present deadline violations, JoBS is able to satisfy all delay constraints at the same time. In Fig. 9, we graph the ratios of loss rates, and the loss rates for all classes, and see that all loss guarantees are respected. When meeting all service guarantees at the
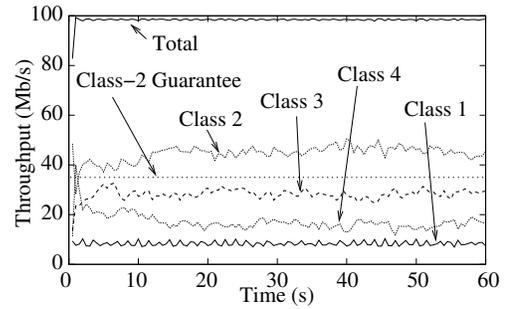
10

same time is not feasible, JoBS relaxes the delay guarantees in favor of the loss guarantees, which explains why a very small fraction of packets miss their deadline. We refer to Christin, Liebeherr and Abdelzaher[19] and to a related technical report[27] for multiple node experiments with near-constant and highly variable offered loads.

We showed by simulation and measurements on a testbed of PC-routers that the feedback-based algorithm was an efficient approximation of the optimization model, and that JoBS was able to provide both proportional and absolute service guarantees. When the set of constraints resulting from the service guarantees is infeasible, JoBS temporarily relaxes some constraints. We note that this study does not make any assumption on traffic arrivals, and does not use the feedback capabilities of TCP traffic. Including the feedback capabilities of TCP can only improve the performance of the algorithm with respect to the service guarantees provided, and avoids situations where the set of constraints is infeasible.

Implementation Overhead   We next present a brief evaluation of the overhead of the feedback-based algorithm, using our PC-router implementation. We have measured the number of CPU cycles consumed by the `enqueue` and `dequeue` procedures, which implement the feedback algorithms and the translation of service rates into packet forwarding decisions, respectively, 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 datagram transmissions on a heavily loaded link. We compare measurements for a set of four classes with four different sets of guarantees. Set 1 is the set of service guarantees we used in Experiment 2. Set 2 is the same as Set 1, without the absolute service guarantees. Set 3 is the same as Set 1, without the proportional service guarantees. Finally, Set 4 does not include any service guarantees, and only serves as a reference to infer the overhead linked to service guarantees. The measurements of the number of cycles are shown in the Table 1. The table shows that the overhead for the `enqueue` operation 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 datagram on the Internet is $\overline{P} = 451$ bytes,[23] this results in a maximum throughput of at least 186 Mbps. We thus showed that our approach was viable at high speeds.

4. CONCLUSIONS AND FUTURE WORK

We sketched the design of a scalable service architecture for providing strong service guarantees. We formally defined our notion of strong service guarantees, by presenting the Quantitative Assured Forwarding service, and then discussed mechanisms to implement the service. Based on a reference algorithm that dynamically solves a non-linear optimization, we devised a heuristic relying on feedback control theory that can be implemented in PC-routers. The implementation of the feedback-based heuristic is now distributed as part of the new ALTQ 3.1 package, and is also available at `http://qosbox.cs.virginia.edu`. Experimental results showed the viability of the approach.

The last piece of work we are currently investigating regards the regulation of traffic arrivals. Indeed, with the architecture of Section 3, it may be impossible to satisfy a set of absolute service guarantees at a given time, in which case JoBS selectively relaxes some service guarantees. This situation occurs in the case an absolute loss bound and an absolute delay bound are conflicting, that is, when traffic must be dropped to satisfy the delay bound, but cannot be dropped without violating a loss rate bound. The current direction of our work focuses on using TCP congestion control algorithms in conjunction with ECN to regulate traffic arrivals, instead of using admission control or traffic policing. The solution we envision for avoiding conflicting service guarantees is to proactively mark packets with the ECN bit before the conflict between the service guarantees occur. More specifically, we suggest to separate TCP and non-TCP traffic into different classes. Non-TCP traffic cannot be regulated by packet marking, and therefore we allow relaxation of

Set	Enqueue		Dequeue		Pred.
					λ_{pred}
	$\overline{X}$	s	$\overline{X}$	s	(Mbps)
1	15347	2603	4053	912	186
2	11849	2798	3580	970	234
3	2671	1101	3811	826	557
4	2415	837	3810	858	580

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

service guarantees for non-TCP traffic classes. On the other hand, for a class solely containing TCP traffic, the input curves can be *projected* in the future, by tracking the congestion window sizes, round-trip times and maximum segment sizes of all backlogged TCP flows. Using these projections, we propose to mark (or drop, if ECN is not available) as many

packets as needed to limit traffic arrivals so that conflicts between service guarantees are avoided ahead of time. We note that the proposed method requires to maintain per-flow information, which defeats our scalability criteria. However, it can be pointed out that a very small number of flows, or "heavy-hitters", contribute to the majority of traffic.[30] For instance, short-lived HTTP connections do not significantly contribute to the arrivals in a given class, due to the fact that their TCP congestion window does not have enough time to reach a significant value. Thus, we only need to monitor long-lived TCP flows. To that effect, sampling techniques can be used and drastically reduce the amount of state information to be maintained.[31] We note that the proposed technique resembles proactive marking/dropping algorithms such as RED.[15] Indeed, it has been shown[16] that using proactive packet marking limited the total number of packet drops, and therefore, increased the aggregate throughput of all TCP flows. We point out that even when ECN is not available, using proactive packet drops has been shown to reduce the long-term loss rates.[15] The major difference with our proposed approach lies in the fact that RED uses probabilistic arguments to avoid maintaining state information, which has led to questioning its configurability.[32] Conversely, our proposed mechanism maintains limited state information, since only long-lived TCP flows are tracked and represent an small fraction of the total number of flows. The algorithm can then use this state information to have an accurate representation of the input curves over a short time interval, and make deterministic marking decisions.

REFERENCES

1. S. Shenker, D. Clark, D. Estrin, and S. Herzog, "Pricing in computer networks: reshaping the research agenda," *ACM Computer Communication Review* **26**, pp. 19–43, Apr. 1996.

2. N. Taft, S. Bhattacharyya, J. Jetcheva, and C. Diot, "Understanding traffic dynamics at a backbone POP," in *Proceedings of SPIE ITCOM Workshop on Scalability and Traffic Control in IP Networks*, (Denver, CO), Aug. 2001.

3. D. Ferrari and D. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications* **8**, pp. 368–379, Apr. 1990.

4. R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview." IETF RFC 1633, July 1994.

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

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

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

8. I. Stoica and H. Zhang, "Providing guaranteed services without per-flow management," in *Proceedings of ACM SIGCOMM '99*, pp. 81–94, (Boston, MA), Aug. 1999.

9. C. Dovrolis and P. Ramanathan, "A Case for Relative Differentiated Services and the Proportional Differentiation Model," *IEEE Networks* **13**, pp. 26–34, Sept. 1999.

10. J. Kaur and H. M. Vin, "Core-stateless guaranteed rate scheduling algorithms," in *Proceedings of INFOCOM 2001*, **3**, pp. 1484–1492, (Anchorage, AK), Apr. 2001.

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

12. C. Dovrolis and P. Ramanathan, "Dynamic class selection: From relative differentiation to absolute QoS," in *Proceedings of ICNP 2001*, pp. 120–128, (Riverside, CA), Nov. 2001.

13. M. Allman, V. Paxson, and W. Stevens, "TCP congestion control." IETF RFC 2581, April 1999.

14. D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Transactions on Networking* **6**, pp. 362–373, Aug. 1998.

15. S. Floyd and V. Jacobson, "Random early detection for congestion avoidance," *IEEE/ACM Transactions on Networking* **1**, pp. 397–413, July 1993.

16. S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review* **24**, pp. 10–23, Oct. 1994.

17. K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP." IETF RFC 3168, September 2001.

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

19. N. Christin, J. Liebeherr, and T. F. Abdelzaher, "A quantitative assured forwarding service," in *Proceedings of IEEE INFOCOM 2002*, (New York, NY), June 2002. To appear.

20. R. Cruz, H. Sariowan, and G. Polyzos, "Scheduling for quality of service guarantees via service curves," in *Proceedings of IEEE ICCCN '95*, pp. 512–520, (Las Vegas, NV), Sept. 1995.

21. C. S. Chang, *Performance Guarantees in Communication Networks*, Springer Verlag, London, UK, 1999.

22. S. Jamin, P. Danzig, S. Shenker, and L. Zhang, "A measurement-based admission control algorithm for integrated service packet networks," *IEEE/ACM Transactions on Networking* **5**, pp. 56–70, Feb. 1997.

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

24. J. Liebeherr and N. Christin, "Rate allocation and buffer management for differentiated services," *Computer Networks* **40**, Aug. 2002. To appear.

25. L. Zhang, "Virtual clock: A new traffic control algorithm for packet switched networks," *ACM Trans. Comput. Syst.* **9**, pp. 101–125, May 1991.

26. K. Schittkowski, "NLPQL: A FORTRAN subroutine solving constrained nonlinear programming problems," *Annals of Operations Research* **5**, pp. 485–500, 1986. Edited by Clyde L. Monma.

27. 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.

28. "The FreeBSD project." `http://www.freebsd.org`.

29. K. Cho, "A framework for alternate queueing: towards traffic management by PC-UNIX based routers," in *Proceedings of USENIX '98 Annual Technical Conference*, pp. 247–258, (New Orleans, LA), June 1998.

30. W. Fang and L. Peterson, "Inter-AS traffic patterns and their implications," in *Proceedings of IEEE GLOBECOM '99*, pp. 1859–1868, (Rio de Janeiro, Brazil), Dec. 1999.

31. C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *Proceedings of the 2001 ACM SIGCOMM Internet Measurement Workshop*, pp. 75–80, (San Francisco, CA), Nov. 2001.

32. C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "A control-theoretic analysis of RED," in *Proceedings of IEEE INFOCOM 2001*, **3**, pp. 1510–1519, (Anchorage, AK), Apr. 2001.