Endpoint Admission Control: Architectural Issues and Performance

Lee Breslau, AT&T Labs
Edward W. Knightly*, Rice
Scott Shenker, ICSI
Ion Stoica†, CMU
Hui Zhang†, CMU

ABSTRACT

The traditional approach to implementing admission control, as exemplified by the Integrated Services proposal in the IETF, uses a signalling protocol to establish reservations at all routers along the path. While providing excellent quality-of-service, this approach has limited scalability because it requires routers to keep per-flow state and to process per-flow reservation messages. In an attempt to implement admission control without these scalability problems, several recent papers have proposed various forms of *endpoint admission control*. In these designs, the hosts (the endpoints) probe the network to detect the level of congestion; the host admits the flow only if the detected level of congestion is sufficiently low. This paper is devoted to the study of endpoint admission control. We first consider several architectural issues that guide (and constrain) the design of such systems. We then use simulations to evaluate the performance of endpoint admission control in various settings. The modest performance degradation between traditional router-based admission control and endpoint admission control suggests that a real-time service based on endpoint probing may be viable.

1. INTRODUCTION

In the last decade a large body of work has been devoted to providing quality of service to individual *real-time* flows. Admission control is the common element of these Integrated Services (IntServ) architectures; that is, flows must request service from the network and are accepted (or rejected) depending on the level of available resources. Typically this involves a signalling mechanism such as RSVP [24] to carry the reservation request to all the routers along the path. While such architectures provide excellent quality-of-service, they have significant scalability problems. Routers must process per-flow reservation requests, and must keep per-flow state to ensure that they deliver the desired quality of service to the appropriate flows; in the limit of many flows, this will place an unbearable burden on routers. While there are attempts to make such designs more scalable through aggregation and hierarchy (see [18, 1] for two such efforts) the scalability of these IntServ architectures remains an open question.

Differentiated Services (DiffServ) is another approach to providing quality of service (see, for example, [3, 17]). DiffServ requires no per-flow admission control or signalling, and routers do not maintain any per-flow state. Routers merely implement a suite of priority-like scheduling and buffering mechanisms and apply them based on the DS field in the packet headers. The lack of admission control means that, upon overload in a given service class, all flows in that class suffer a degradation of service. Providing quality of service for individual realtime flows is not the primary purpose of DiffServ, but the combination of provisioning, service-level-agreements and DiffServ router mechanisms may prove sufficient for this task.

In an attempt to combine DiffServ's superior scalability with IntServ's superior quality-of-service, several recent papers [2, 5, 7, 13, 15] have proposed the quite novel approach of using *endpoint admission control*. In these designs, the end host [1] probes the network by sending probe packets at the data rate it would like to reserve and recording the resulting level of packet losses (or ECN congestion marks [19]). The host then admits the flow only if the loss (or marking) percentage is below some threshold value. Because these designs rely on necessarily imprecise network measurements to guide their admission control decisions, endpoint admission control is primarily intended to support a soft real-time service similar to Controlled-Load [22] in which the aggregate load is kept at reasonable levels but no hard or precise service guarantees are made to individual flows. Given that the queueing delays are likely to be quite small, the quality of service is measured strictly in terms of packet loss; the goal is to make this loss rate small but not to give any precise assurances of how small. [2]

Endpoint admission control is much like the traditional IntServ ap-

*Ed Knightly is sponsored by NSF CAREER Award ANI-9733610, NSF Grant ANI-9730104, and Texas Instruments.

†Ion Stoica and Hui Zhang are sponsored by DARPA under contract numbers N66001-96-C-8528, F30602-99-1-0518, and E30602-97-2-0287, and by NSF under grant numbers Career Award NCR-9624979 ANI-9730105, and ANI-9814929. Additional support was provided by Intel, Lucent, and Ericsson. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, NSF, Intel, Lucent, Ericsson or the U.S. government.

SIGCOMM '00, Stockholm, Sweden.

[1] The endpoint can either be a host or an edge router (as it is in [5]), but in this paper we will focus primarily on host endpoints.

[2] There are some more stringent (but still soft) real-time services that attempt to provide an upper bound on the loss rate. Most of the basic architectural points made in Section 2 apply equally well to this *statistical* service; in Section 4 we briefly discuss the possibility of achieving this more stringent statistical service.

proach in that admission control is used and flows are admitted only if resources are available.[3] However, endpoint admission control requires no explicit support from the routers; routers keep no per-flow state and do not process reservation requests, and routers drop or mark packets in a normal manner (perhaps using the various priority mechanisms supplied by DiffServ). Thus, endpoint admission control designs do not have the scalability problems associated with IntServ but their goal is to provide a quality of service similar to IntServ's. Endpoint admission control is an attempt to use the regular *best effort* infrastructure (with its DiffServ extensions) and, by adding control algorithms at the endpoints, deliver a real-time service. If successful, this would represent a dramatic shift in the way real-time services are supported. The crucial question, and the one we focus on here, is whether (and how) such endpoint admission control designs are indeed able to adequately support a soft real-time service like Controlled-Load.

The specific endpoint admission control proposals in the literature all share similar architectures but differ significantly in detail. The scheme described in [13, 15] is part of a more general proposal to base pricing on ECN congestion marks. All packets are treated identically – data and probe packets are indistinguishable, as are best-effort and real-time packets – and packets are marked upon congestion. Flows can send as much traffic as they wish, but must pay for those packets that are marked. In this setting, admission control is a service offered by third-parties (or the network itself) to provide a guaranteed price for the flow (as opposed to, in traditional IntServ, guaranteeing the level of service). A quite different design is described in [7]; this design uses packet drops, rather than congestion marks, to indicate congestion and sends the probe packets in a separate (lower) priority class. A very similar design based on packet drops and probe packets in a lower priority class is also considered in [2]. In [5] the *endpoint* is not the host but is the edge router. While hosts must necessarily probe to detect congestion, edge routers can passively monitor paths to ascertain the current load levels. Passive probing may provide more accurate estimates of the current network load, and it has the added advantage that flows need not endure the probing delay before sending.

Each of these previous papers proposed a specific design, and evaluated that design's basic functionality. While these proposals each have their own merit, what is missing from the literature is a broader exploration of the fundamental architectural and performance issues inherent in endpoint admission control. This paper is our attempt to elucidate some of these issues.

This paper is organized as follows. In Section 2 we use a very simple model to discuss several basic architectural issues. These architectural issues suggest a set of designs that might be capable of supporting a soft real-time service. We are not claiming novelty for these designs – in fact they are rather similar to the designs in the literature discussed above – we claim only that they enable us to investigate the range of design options available in endpoint admission control. We bound this range of options by confining ourselves to designs that are plausibly deployable in the near future. By this we mean that the design must not require any router functionality beyond the priority and marking mechanisms that may be available as part of DiffServ and ECN. This deployability condition eliminates several otherwise attractive possibilities. First, we do not consider designs that require routers to process reservation messages no matter how lightweight that processing is. The lack of reservation messages distinguishes the class of endpoint admission control designs from lightweight signalling proposals such as [8] and [21].[4] Schemes based on Dynamic Packet State [20], which eliminate per-flow state in core routers but still require per-flow signalling and admission control, also violate the deployability condition. Second, the deployability condition implies that we focus only on using hosts as the endpoints in our algorithms, rather than edge routers; we thus cannot avail ourselves of the possible advantages of passive monitoring by edge routers and must rely on hosts actively probing to detect congestion. In particular, probing inherently involves a significant set-up delay, on the order of seconds; not all real-time applications will easily tolerate such sizable set-up delays. Finally, as part of our deployability requirement we do not consider designs that involve changes to the current best-effort service or its pricing structure; this eliminates proposals like [13, 15] that, while quite attractive and intriguing, would face significant deployment hurdles.

Armed with the set of possible designs from Section 2, we then turn to evaluating these designs through simulation. We seek to understand the extent to which these various designs can support a soft real-time service similar to Controlled-Load. For reference, we compare these designs to a conventional measurement-based admission control algorithm. In Section 3 we define in more detail the algorithms to be tested and then describe our simulation methodology. We then present our simulation results in Section 4 and address several performance issues. We conclude in Section 5 with a brief summary of our results.

2. ARCHITECTURAL ISSUES

In this section we discuss some basic architectural issues, and use these considerations to motivate the set of design options that will be simulated in Section 4. The proposals in the literature use either packet drops or congestion marks as an indication of congestion. In what follows we will refer to packet dropping as the default choice (unless otherwise specified) but our points apply equally well to both. Similarly, some proposals have the probe packets sent at the same level of priority as the data packets, and others proposals have them in a separate (lower) priority class; in what follows we will assume the former as the default case unless specified otherwise.

To facilitate the architectural discussion, we employ an oversimplified fluid-flow model of a single congested link of capacity C. Each flow[5] i sends at some fixed rate r_i, and the resulting dropping rate is $\sum_i r_i - C$ and the dropping fraction is $\frac{\sum_i r_i - C}{\sum_i r_i}$. To gain admission, a flow sends its probe packets at rate r_i for some period of time and measures the resulting loss fraction; the flow is admitted only if the probe loss fraction is below some threshold ϵ_i. For most

[3] We should also point out that endpoint admission control resembles the current congestion control paradigm in that neither require router support, and both use host probing to detect the current level of congestion; the key difference between the two is that in congestion control hosts continually adjust their current transmission rate in an attempt to share the available bandwidth fairly (for some definition of fair) whereas in endpoint admission control, as in the traditional IntServ approach, an initial binary decision of whether the flow is admitted or not is made, and there are no subsequent adjustments to the flow's bandwidth (and no subsequent probing).

[4] Our lack of consideration of these proposals should not be taken as a condemnation of lightweight signalling protocols. We are not attempting to weigh the respective merits of the two approaches (endpoint admission control and lightweight signalling) in this paper, merely to provide more insight into the properties of endpoint admission control designs.

[5] For convenience, we equate hosts with flows in the discussion below.

of this section we will assume that probing is perfect; i.e., the loss fraction measured by the flow is exactly $\frac{\sum_i r_i - C}{\sum_i r_i}$ (where the sum includes the r_i of the probing flow). While this model is obviously quite unrealistic, we use it only to illustrate the various architectural issues we address below. We divide the architectural issues into two categories: those that are relevant to the router scheduling mechanisms and those that are relevant to endpoint probing algorithms.

2.1 Router Scheduling Mechanisms

In this section we discuss issues that help identify which scheduling mechanisms are compatible with endpoint admission control algorithms. Our point here is not to design complicated new scheduling algorithms; in fact, we are constrained to use those mechanisms currently in use or proposed for traditional best effort and DiffServ and we are merely investigating which of those mechanisms best support endpoint admission control. For the purposes of this section we assume that $\epsilon_i = 0$ for all i. The insights do not change when we consider nonzero ϵ_i but the algebra becomes unnecessarily cumbersome.

2.1.1 FIFO or Fair Queueing: the issue of stolen bandwidth

Two obvious possibilities for packet scheduling algorithms are FIFO and Fair Queueing. We now consider the possible impacts on these two scheduling algorithms on endpoint admission control. A successful probe – one that detects a dropping percentage less than or equal to ϵ – indicates that the flow would receive an adequate level of service under current conditions.[6] Can the flow count on this level of service to continue, or could its bandwidth be *stolen* from it by subsequent arrivals? We use our simple model of a single congested link and consider two groups of flows: those with rate r_1 and those with rate r_2 (we assume $r_2 > r_1$). Let n_1 and n_2 denote the number of flows currently resident (or probing) in each group.

If the router uses FIFO packet scheduling, then no flow will ever be admitted if the load while probing (given by $n_1 r_1 + n_2 r_2$, where the n's include the probing flow) is greater than the capacity C; this standard applies to all flows, so no flow will be admitted if any flow would be experience a significant loss rate (although, as we will see in Section 2.2.3, probing flows may produce significant loss even if the number of accepted flows is small).

If the router uses Fair Queueing scheduling (or some other variant that enforces max-min fair bandwidth allocations) then the situation is quite different. A probing flow in the first group is admitted (and existing flows in that group continue to receive acceptable service) as long as $r_1(n_1 + n_2) < C$; a flow in the second group is admitted (and existing flows in that group continue to receive acceptable service) as long as $n_1 r_1 + n_2 r_2 < C$. If at some given time we have $n_1 r_1 + n_2 r_2 < C$ flows of the second type will admitted; if a short time later several flows of the first type arrive, these newly arriving flows will be admitted until $r_1(n_1 + n_2) = C$. At that point, the loss fraction for the first group of flows remains zero, while the loss fraction for the second group of flows is $\frac{r_2 - r_1}{r_2}$. If we take $r_2 = 2r_1$ then this loss fraction is $\frac{1}{2}$ which is clearly unacceptable. In such situations the larger flows (those in the second group) will experience substantial losses even though, when they initially probed, the network was completely uncongested. This shows that Fair Queueing's ability to isolate flows from each other – to give each flow its fair share regardless of the overall load – is not suited to endpoint admission control; such isolation leads to situations where smaller flows are admitted even when their acceptance impairs the service being delivered to already accepted larger flows. Thus, in designing endpoint admission control architectures one should not use Fair Queueing or its variants to service admission-controlled traffic.[7]

2.1.2 Coexisting with Best-Effort Traffic

One of our design constraints is that the endpoint admission control coexist with current best-effort traffic. There is widespread agreement that, in the current infrastructure, all applications sharing bandwidth with TCP should deploy TCP-friendly congestion control. Moreover, there are efforts to deploy various forms of *penalty boxes* that would punish flows for not being TCP friendly [10]. The admission-controlled flows we have been discussing are not TCP-friendly, and thus would not share fairly with existing TCP applications and may very well be punished by penalty boxes. Thus, it is necessary to provide isolation between the TCP traffic and the admission-controlled traffic. The discussion above suggests that this isolation cannot be achieved by using a Fair Queueing or CBQ scheduler to give best-effort and admission-controlled each a share of the bandwidth. One needs a mechanism that does not allow the admission-controlled traffic to ever *borrow* bandwidth from best-effort (because that bandwidth might fool a probe into thinking extra bandwidth was available) and does not allow the best-effort traffic to pre-empt admission-controlled traffic. The admission-controlled traffic thus needs a strict upper bound (so it never borrows) and lower bound (so it is never pre-empted) on its available bandwidth.

The easiest way to accomplish this is to serve the admission controlled traffic in a higher priority class, but to strictly limit its share of bandwidth to some fraction of the link bandwidth (the amount of the allocated share is a local administrative decision and need not be uniform across routers). Simple priority queues with a rate limiter (as described in [23]) could easily be included in DiffServ equipped routers. Note that such queueing mechanisms are not work conserving; if there is no best-effort traffic present when the admission-controlled traffic exceeds its limit, the scheduler temporarily leaves the link idle rather than sending the admission-controlled traffic.

As a side note, we observe that one could use other forms of rate-limited schedulers, such as rate-limited Fair Queueing or Round Robin, to share between best-effort and the admission-controlled traffic. In each of these, the admission-controlled traffic needs a strict rate limit, but the best-effort traffic does not. All that is required of the scheduling algorithm (beyond the rate-limit) is that when there is admission-controlled traffic present it gets its allocated share of the bandwidth. We focus on the priority service only because it is the simplest way to achieve this, and it is likely that the admission-controlled traffic is more delay-sensitive than typical best-effort traffic.

[6] We do not mean to imply that flows require the dropping percentage to be below some precise target in order for service to be acceptable. However, we do assume that if the loss percentage is below ϵ then service is acceptable and that if the loss percentage is quite high then the service is not acceptable.

[7] Nothing in this section has any bearing on whether Fair Queueing should be used to service best effort traffic. Our conclusion is only that it should not be used for the admission-controlled traffic.

2.1.3 Multiple Levels of Service

Once we are using priority queueing mechanisms, it seems natural to offer several levels of admission-controlled service with these levels of service receiving different priorities.[8] However, if the probes are sent at the same priority as the latter data packets, then offering several levels of priority introduces the same stealing problem we saw in Section 2.1.1. To see this more clearly, imagine all flows have the same rate r. Let n_1 and n_2 denote the number of flows using priority levels 1 and 2 (with class 1 being higher priority). Flows will continue to be admitted to (and continue to receive acceptable service in) level 1 as long as $n_1 r < C$, while flows will be admitted to (and continue to receive acceptable service in) level 2 as long as $(n_1 + n_2)r < C$. As in Section 2.1.1, any flows admitted to level 2 may have their bandwidth *stolen* by later admissions to level 1. To be concrete; if there are currently flows in level 2 and $(n_1 + n_2)r < C$, but new flows arrive in level 1 making $n_1 r = C$ then all packets in level 2 are dropped and the flows in that class are completely deprived of service even though they detected no congestion when they initially probed.

The above argument shows that one cannot have multiple levels of priority for the probes. However, if one uses a different DS field in the probe packets than in the data packets so that all probe packets go into the same priority class then one can still have the data packets sent at different levels of priority as long as the probe traffic is at the same, or lower, priority than all other admission-controlled traffic. In such a design, all admission-controlled flows would compete on an equal basis for admission, but would receive different levels of service once admitted.

In summary, the problem of *stealing* bandwidth leaves us with very little design leeway; one cannot use scheduling algorithms that allow the admission-controlled traffic to borrow from other traffic. In particular, one should not use traditional Fair Queueing (or other variants) on a per-flow basis (to separate admission-controlled flows from each other) or on a per-class basis (to separate admission-controlled traffic from best effort traffic). One must use scheduling mechanisms with strict rate limits on the admission-controlled flows. A natural mechanism is priority scheduling with a strict bandwidth limit to separate admission-controlled flows from best-effort flows, with FIFO service within the admission-controlled traffic itself. In addition, one can offer multiple levels of admission-controlled service but only if all probing traffic uses the same level of priority.

2.2 Endpoint Probing Algorithms

In what follows we assume there is only one class of admission-controlled service.

2.2.1 Acceptance Thresholds

Given that admission control decisions are made by the hosts, one might think the hosts should be free to choose a wide variety of acceptance thresholds. To investigate this possibility, consider the following example. There are two groups of flows, one with acceptance threshold ϵ_1 and another with acceptance threshold ϵ_2; all flows send at rate r. Let's assume that $\epsilon_2 > \epsilon_1$ so the flows in the second group have a less stringent admission standard than flows in the first group. Flows in the first group will be admitted if there are fewer than $n_1 = \frac{C}{r}\frac{1}{1-\epsilon_1}$ flows currently accepted or probing; similarly, flows in the second group will be admitted if there are fewer than $n_2 = \frac{C}{r}\frac{1}{1-\epsilon_2}$ flows currently accepted or probing. Note that $n_2 > n_1$; thus, when the number of current flows (either admitted or probing) is between n_1 and n_2 only those flows in the second group are admitted. The relative size of this *window* where only the second group is admitted (compared to the total window of when any flow is accepted) is given by $\frac{n_2 - n_1}{n_2} = \frac{\epsilon_2 - \epsilon_1}{1-\epsilon_1}$. This suggests that if all the acceptance thresholds are small then the relative size of the window is also small. If the probability distribution on flow occupancy – that is, the distribution of the number of flows present at a given time – is reasonably uniform, then the acceptance rates (or blocking probabilities) of the two groups of flows are extremely similar. In this case, flows could indeed have a great deal of freedom in picking their ϵ (as long as they are reasonably small). However, there are two relevant observations.

First, while in this case adopting a more stringent admission standard (adopting a lower ϵ) does not hurt a flow, in that its blocking probability will not be much higher than if it chose a less stringent acceptance threshold, neither does it improve that flow's quality of service. The quality of service experienced by a flow is a function of the total number of flows present. Even if the more stringent flows would only load the link up to a maximum of n_1, the less stringent flows would increase the link's load up to n_2 and all flows would experience the same dropping fraction; the quality of service of all flows depends on the least stringent acceptance threshold of any flow. Thus, flows have little to gain by choosing a more stringent acceptance threshold.

Second, if the incoming load is reasonably heavy, the probability distribution on flow occupancy – that is, the distribution of the number of flows present at a given time – is heavily weighted towards n_2. If this is the case, the blocking probability of the more stringent flows would indeed be significantly higher. We investigate this phenomena through simulation in Section 4.

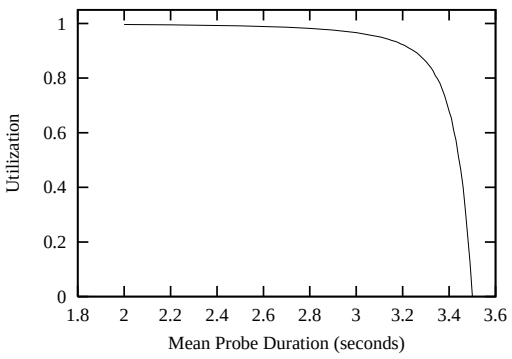
This analysis suggests that endpoint admission control would function best if all flows adopted the same acceptance threshold ϵ. This is reminiscent of the current end-to-end congestion control paradigm, where there is no router support for bandwidth allocation (as there is in Fair Queueing and related approaches) and the hosts are responsible for backing off in the presence of congestion. For congestion control to function properly all flows must adhere to the uniform standard of TCP-friendliness [10] or else some flows might get substantially more than their share of the bandwidth.

2.2.2 Accuracy

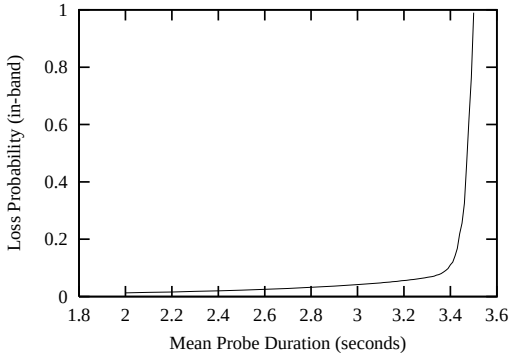
For a host to be sure that the loss (or marking) fraction of a probe is below ϵ the probe must last for many multiples of ϵ^{-1} (measured in packet transmissions, not time). Small ϵ means extremely long probe times, resulting in significant wasted bandwidth and a substantial delay before the host can send data traffic.

If packet loss is used as the congestion signal and probe traffic is sent at the same priority as data traffic, then ϵ is a reasonable predictor of the likely data packet loss levels. There is a choice between long set-up times and small loss fractions, or short set-up times but somewhat higher loss fractions. An alternative is to send the probe packets at a lower priority than the admission-controlled data traffic (but still at a higher priority than best-effort traffic); we call this *out-of-band* probing, in contrast to the default of *in-band* probing. With out-of-band probing the data packet loss fraction is substantially lower than the probe packet loss fraction. Thus, one can have

[8] For the purposes of this discussion we will assume we are talking about priority scheduling, but our comments apply equally well to priority dropping.



(a) Utilization



(b) Loss Probability

Figure 1: Illustration of Thrashing Behavior: The utilization and data packet loss probability for in-band and out-of-band probing. The fluid model has Poisson arrivals, exponential flow times and exponential probe times. The flow inter-arrival time is 3.5 sec, the average flow lifetime is 30 sec, the link bandwidth is 10 Mbps, and the flow bandwidth is 128 kbps. The x-axis is the average probe length. The utilization (shown in (a)) is exactly the same for the in-band and out-of-band models. The data packet loss fraction (shown in (b)) is that of the in-band model; the out-of-band-model has no loss.

a reasonably sized ϵ, with its corresponding reasonable set-up delays, and still achieve low data losses.

Another alternative is to use congestion marks (as in ECN) to indicate congestion. The rate of packet marking will be substantially higher than the rate of packet dropping, so one can again use a larger value of ϵ to reach the same level of loss.

Thus the probing time needed to achieve a given level of loss is much less if one uses either out-of-band probing or uses marking rather than dropping as the congestion signal. However, these two alternative approaches – out-of-band probing and congestion marking – have the disadvantage that it is quite difficult to relate the acceptance threshold to the likely level of loss. In these two cases (as observed in [7] for out-of-band probing) the acceptance threshold serves as a very loose upper bound on the likely level of loss (if the probing is done for a sufficiently long period of time).

We will explore the tradeoffs between probing in-band and out-of-band, and between marking and dropping in Section 4.

2.2.3 Thrashing

When many flows are probing at once – so many that the drop percentage is significantly above ϵ – none of the probing flows will be accepted. This holds true even if the current number of accepted flows is quite low. It is thus possible under high offered loads for the system to enter into a *thrashing* regime where the number of accepted flows is well below what the link could comfortably handle but the cumulative level of probe traffic prevents further admissions.

We can quantify this effect with a simple (but unrealistic) fluid flow model with dynamic arrivals and departures and using packet drops as the congestion signal. The flows arrive in a Poisson process and have exponential lifetimes. The probes are exponential in length and we assume they make perfect measurements (i.e., detect the load level exactly); this is particularly unrealistic, but it is necessary to make the model tractable. For space reasons we omit a more detailed description of this calculation and merely present the numerical results in Figure 1. Figure 1(a) shows the *useful* utilization of the link, which is the fraction of the link utilized by data packets. Figure 1(b) depicts the data packet loss probability. The utilization applies to both the in-band and out-of-band models. The data packet loss probability applies only to the in-band model; the out-of-band model has no data packet loss. These functions are plotted against the average probe length; similar curves would result if we increased the Poisson arrival rate of flows with a fixed average probe time. What these curves reveal is that as the length of probes increases (or as the incoming load increases) the system undergoes a fairly sharp transition. Below the transition, the utilization is quite high and the data packet loss probability quite low. Past the transition the number of probing flows begins to accumulate without bound (because the incoming rate is higher than the outgoing rate); the utilization then collapses to zero and the loss fraction (in the in-band case) approaches one. Note that when probing is in-band the system experiences a *collapse*; thrashing not only denies new admissions, driving utilization to zero, but thrashing also raises the data packet drop percentage. In contrast, if the probing is out-of-band the system experiences *starvation*; the drop percentage remains low but thrashing drives the utilization to zero.

Note that traditional IntServ admission control does not have a similar thrashing problem at a single link.[9] This is because when requests arrive at a router, they are serialized (i.e., they come in order) and admission control decisions can be made sequentially. In endpoint admission control, as in Ethernets and other multiple access problems, there is no central point of serialization. When two flows seek admission at the same time, and there is only room for one, an IntServ router can admit one but not the other; in an endpoint admission control architecture both are rejected.

Since we expect that there will be occasional periods when the offered load is much higher than the link capacity, the probing algorithm should be designed to minimize thrashing. We suggest using *slow-start* probing, in which the probing rate slowly ramps up, to detect congestion without unnecessarily creating it. One example of such a scheme is to have the rate of the probe traffic start out small for some period of time; if the observed loss (or marking) percentage is below ϵ then the host doubles the probing rate for the

[9] The IntServ approach can have a thrashing problem in a multi-link scenario when resources are scarce at more than one router along a path [16].

next period of time. This process repeats until the desired transmission rate is reached. This allows the flow to receive an early signal of congestion before overloading the link. We are not able to include slow-start probing into the model above (the state space explodes), but we test it through simulation in Section 4.[10]

3. SIMULATION PRELIMINARIES

We first define the designs to be tested and then describe the simulation methodology.

3.1 Prototype Designs

The discussion of router scheduling mechanisms greatly limited the design options. It appears that the most sensible design is to use a strict priority packet scheduler, with a bandwidth (and buffer) limit on the admission-controlled traffic. The bandwidth limit determines the share of bandwidth allocated to admission-controlled traffic. Determining this allocation is a local administrative decision. We expect that best-effort traffic will dominate admission-controlled traffic (in terms of bandwidth usage) so that admission-controlled traffic's allocated share of the bandwidth will likely be 50% or less of the total link.

Within the admission-controlled traffic we also consider another level of priority so that probe traffic can be sent at a lower priority level than data traffic (but still higher than best-effort traffic); there is no bandwidth cap limiting the amount of bandwidth that must be left over for the probe traffic. That is, there is a bandwidth limit that applies to the sum of the admission-controlled data and probe traffic, but there is not a separate bandwidth limit for the admission-controlled data traffic. When we have this additional level of priority for the probe packets, incoming data packets push out resident probe packets if the buffer is full.

We test both packet drops and congestion marks as the signal of congestion. The dropping behavior of these admission-controlled queues can be either drop-tail or RED [11]; we used drop-tail for ease of simulation but we don't think this affected the results.[11] For the marking algorithm we use a *virtual queue* similar in spirit to what is discussed in [6] and [13]. The router simulates the behavior of a queue with 90% of the real bandwidth (but same size buffer) and marks packets that would have been dropped in the virtual queue. This can be implemented efficiently, as it only requires one counter for each priority level and an update of that counter on packet arrivals.

Hosts must characterize their flows as conforming to an (r, b) token bucket (as in [22]). In the simplest form of probing, the host sends probe packets at a rate r for some fixed duration; in our simulations, that duration is 5 seconds.[12] At the end of the probing interval, the loss percentage is computed and the admission decision is made; the receiving host records the losses and communicates the acceptance/rejection decision to the sending host. If, before the end of the probing period, the number of packet losses has already reached the point where the total loss percentage will be over threshold, then the probing is stopped and the flow rejected. For instance, if the probe rate is 1000 packets per second, and the acceptance threshold is 1%, then once 51 packets are dropped the probing is halted and the flow is rejected.

As discussed in Section 2.2.3, it may be necessary to slowly ramp up the probe rate in order to prevent starvation or collapse. For this version of *slow-start* probing, we first probe at rate $\frac{r}{16}$ for 1 second; if the loss (or mark) percentage is below threshold we probe at rate $\frac{r}{8}$ for a second and again the loss percentage is checked. This process is continued for 5 seconds at which time the host, if the flow has not been rejected, has probed for a second at rate r.

An intermediate version of probing, which we call *Early Reject* probes at the rate r for at most five seconds (as in the simple probing algorithm) but if in any second-long interval the loss (or mark) percentage is above threshold then the flow is rejected (and the probing stopped). This algorithm allows us to determine if the performance differences between slow-start probing and the simple probing algorithms are due to the early rejection or the incremental increases in bandwidth.

The probing algorithms described above do not take the bucket size b value into account when they probe. We could easily modify them to put the probe packets into bursts of size b followed by a quiescent period of time $\frac{b}{r}$. Alternatively, one could probe at some rate r' that is a function of the r and b. For instance, some measurement-based admission control (MBAC) algorithms use an *effective peak rate* that is a function of r and b (see [9]); this value could be used as the probing rate.

As we argued in Section 2, the admission control threshold should probably be a uniform standard.[13] For most of our simulations we assume that all flows use the same threshold ϵ. However, we do run one test where the ϵ's are different to evaluate the impact of threshold heterogeneity.

Aside from the options in probing (slow-start, early reject, or simple) the main design options we explore through simulations are whether to probe in-band or out-of-band and whether to signal congestion with packet drops or congestion marks. Thus, we have four basic design choices: dropping in-band, dropping out-of-band, marking in-band, and marking out-of-band. Dropping in-band is the simplest scheme, and requires only a rate-limited priority scheduler to separate admission-controlled traffic from best-effort traffic. The in-band marking scheme is very similar to that proposed in [13] and requires simulating a virtual queue and the use of ECN-like bits. The out-of-band dropping scheme is similar to that proposed in [7] and requires three levels of priority (one for admission-controlled data, one for probes, and one for best-effort). The out-of-band marking scheme is a hybrid of these latter two approaches.[14]

[10] In addition, as in regular IntServ, rejected flows should use exponential back-off before retrying (see [16]), but we do not explore the issue of retrying flows here.

[11] While RED has substantial advantages for TCP flows, it is not clear that it provides substantial benefits for traffic that is not adjusting its transmission rate (once it has been accepted).

[12] Five seconds is chosen to balance the delay that a user might tolerate at the start of a real-time session with the need to achieve an accurate sample of network performance though probing. In Section 4 we use simulation to further understand this tradeoff.

[13] However, it is appropriate to set ϵ differently for different algorithms, since the relationship between ϵ and actual performance varies for in-band and out-of-band probing. Once the decision is made to adopt a certain design, the uniform value of ϵ can be set.

[14] The real difference between the out-of-band marking scheme and the out-of-band dropping scheme is *not* the use of congestion bits but instead is the use of a virtual queue. One could use the virtual queue idea to decide when to do something about the probe packets, but instead of marking them merely drop them. This *virtual dropping* design would remove the need for ECN bits while still giving early congestion signals. This is possible with out-of-band marking, but not in-band marking, because having a separate queue

Source	Burst Rate	On Time	Off Time	Avg. Rate	β
EXP1	256k	500ms	500ms	128k	–
EXP2	1024k	125ms	875ms	128k	–
EXP3	512k	500ms	500ms	256k	–
EXP4	256k	5000ms	5000ms	128k	–
POO1	256k	500ms	500ms	128k	1.2

Table 1: Traffic Sources: The burst rates and average rates are in units of bits per second.

We will be testing these four variations in various scenarios. The Measured Sum algorithm, a traditional IntServ per-hop measurement-based admission control (MBAC) algorithm described in [14], will serve as a benchmark for these endpoint admission control designs.

Our design does not address the issue of multicast. One could use the simple algorithm that when a receiver decides that a flow has been rejected, it merely leaves the multicast group. The large leave-latencies of current multicast implementations may hinder this approach, but otherwise it would result in the desired state: the admission-controlled traffic would only travel paths along which the probes passed successfully. We do not test the multicast design in this paper.

3.2 Simulation Methodology

The admission-controlled traffic is given strict priority over best-effort traffic, but there is a bandwidth limit. In our simulations[15] we merely simulated the admission-controlled traffic as being serviced by a queue running at the speed of its bandwidth limit. This is not precisely the behavior of a rate-limited priority queue, but it is rather close. This simplification frees us from simulating the best-effort traffic running at strictly lower priority. Correspondingly, when we report utilization figures in the simulation results, these refer to the amount of the allocated share, not the link bandwidth, that is consumed by admission-controlled data packets. We do not include probe traffic in our utilization figures (because they do not reflect useful transmissions).

In the simulations that follow, the admission-controlled traffic is modeled by a Poisson arrival process with average interarrival time τ. Flows that are rejected do not retry; this is obviously unrealistic, but our simulations are simplified if we consider retrying flows as part of the incoming Poisson process (i.e., retrying flows would merely make λ effectively larger). All flows have exponential lifetimes with an average lifetime of 300 seconds.

We use six different traffic sources. Five of them are on-off traffic sources, with four of them having exponential on and off times. The fifth on-off traffic source has Pareto on and off times (described by a shape parameter, β). This source produces LRD traffic in the aggregate. All packets from these on-off sources are 125 bytes in length. Table 1 contains the parameter values for these five on-off sources; each of them conform to a token bucket with $b = 125$ bytes with the rate r given by the burst rate. Exponential and Pareto on/off sources are denoted with the labels EXP and POO, respectively. The last source is a trace from the Star Wars movie [12] which uses 200 byte packets; we reshape (by dropping) it to conform to a token bucket with $r = 800$kbps and $b = 200$kb.

allows the router to use the virtual queue to drop probe packets and not data packets.

[15] We used the *ns* simulator, extending it to support the functionality required for our experiments.

Figure	Source(s)	τ (sec)	Description
2	EXP1	3.5	Basic scenario
3	EXP1	3.5	Longer probing
4- 7	EXP1	1.0	Higher load
8(a)	EXP2	3.5	Four times burst rate, same average
8(b)	EXP3	7.0	Twice burst and average
8(c)	POO1	3.5	Long-tailed on/off times
8(d)	Star Wars Trace	8.0	Real trace data
8(e)	EXP1, EXP2, EXP4, POO1	3.5	Heterogeneous traffic sources
8(f)	EXP1	35	Low multiplexing
Table 3	EXP1	3.5	Heterogeneous ϵ
Tables 5-6	EXP1	-	Multi-link topology
11	EXP1	3.5	Coexistence with TCP

Table 2: Simulation Scenarios

All but one of our simulations uses a simple topology with many sources sharing a single congested link. The bandwidth of this link, unless otherwise specified, is 10Mbps and it has a propagation delay of 20msec. There is enough buffering for 200 packets. We also use a 12-node topology to assess the impact of flows traversing different numbers of congested links.

We test in-band marking and in-band dropping with $\epsilon = 0, .01, .02, .03, .04$ and $.05$. We test out-of-band marking and dropping with $\epsilon = 0, .05, .10, .15$ and $.20$. All simulations are run for 14,000 simulation seconds, and data for the first 2000 seconds are discarded.

For reference, Table 2 lists all of the simulation scenarios that we describe in the following section.

4. PERFORMANCE ISSUES

We now use simulation to evaluate the performance of the endpoint admission control designs.

4.1 Basic Scenario

We start off by considering a rather basic scenario consisting of our single link topology and EXP1 sources. Figure 2 shows the data packet loss probability versus utilization achieved for the 5 tested algorithms (4 endpoint admission control algorithms and the reference MBAC.) Slow-start probing is used for endpoint admission control. Offered load is such that the blocking rates in these experiments are approximately 20%. For a given endpoint admission control algorithm each point shown reflects the loss probability and utilization produced by a different ϵ value, averaged over 7 simulation runs with different random seeds; following [4] we call the curve described by these points the *loss-load* curve of the algorithm. There are two aspects to these curves. First, one cares about the loss value for a given level of utilization (or, equivalently, the utilization at a given loss value); all other things being equal, algorithms that produce lower loss at the same utilization level are clearly superior. Second, one cares about the range of utilization (or loss rates) that can be achieved by varying the ϵ parameter for a given probing length. We say that two loss-load curves have the same frontier if (by extrapolating loosely) it appears that the losses at the same level of utilization would be similar. We say the curves have different ranges if the levels of utilization (at the end points) are quite different.

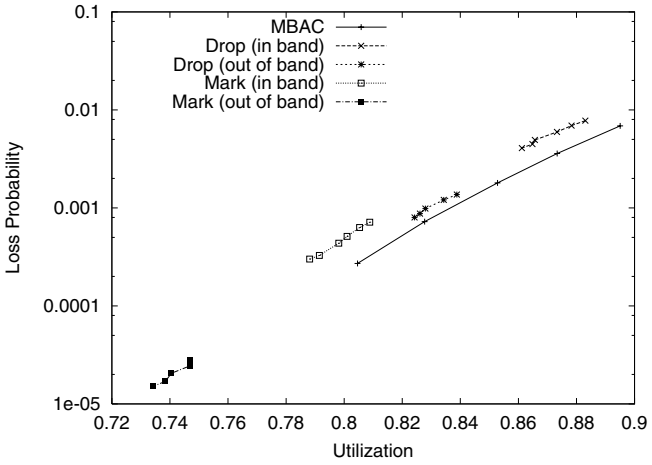


Figure 2: Basic Scenario – EXP1 traffic source, $\tau = 3.5s$

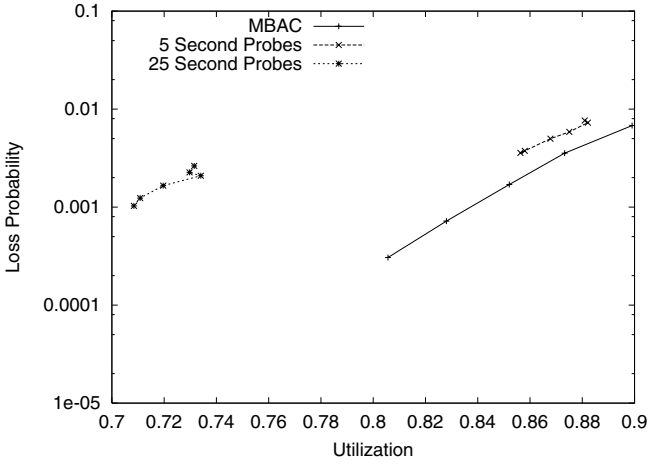


Figure 3: Basic Scenario with Long Probing

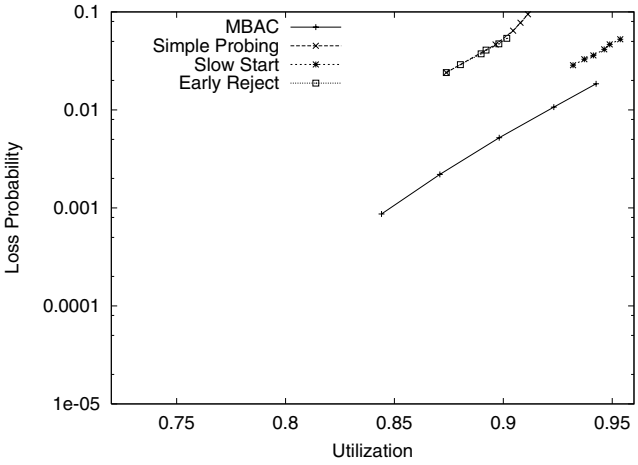


Figure 4: High Load: in-band dropping

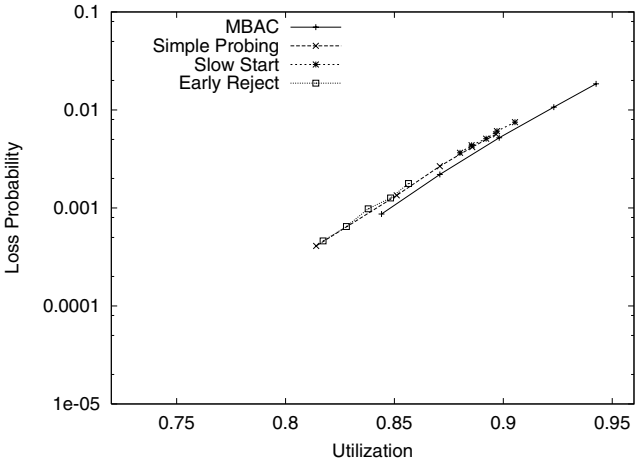


Figure 5: High Load: out-of-band dropping

With this in mind, we now turn to Figure 2. The frontiers of the various loss-load curves seem to be fairly close to the benchmark MBAC curve, and even closer to each other. Thus, for a given level of utilization the loss rates achieved by endpoint admission control are quite competitive with the loss rates produced by the MBAC (consistently within a factor of two, and often closer). The most striking aspect of the results is the dramatically different ranges of these loss-load curves. The out-of-band marking is able to achieve dropping rates on the order of 10^{-5} while only probing for 5 seconds. In contrast, the in-band dropping algorithm's minimal drop rate exceeds 10^{-3}. The in-band marking and out-of-band dropping reach intermediate levels of loss.

Note that even when $\epsilon = 0$ the in-band dropping algorithm has significant losses (0.4%). To see why this is so, let's consider a simple model (using simple probing, not slow-start probing) and assume the link has a fixed drop percentage δ. Note that if one fixes a probe time T and one's probe rate is r (with packet size P), then even if one sets $\epsilon = 0$ flows will be admitted with probability $(1-\delta)^{\frac{rT}{P}}$. A flow will be admitted with 50% probability when $\delta = 1 - 2^{-\frac{P}{rT}}$. This value of δ is a rule-of-thumb estimate of how low a drop rate in-band dropping can achieve for a given probing interval. For the scenario considered here, this results in a rule-of-thumb drop rate of 0.13% which is about a third of what is observed.

One could presumably achieve lower loss rates by probing for longer (if one were willing to incur the longer set-up delays). In Figure 3 we compare the loss-load curves achieved for in-band dropping with our usual 5 second slow-start probing algorithm to one which probes for 25 seconds (5 seconds at each probing rate as it doubles towards r). We find that while longer probing does lead to decreased drop rates, utilization has also decreased substantially because more bandwidth is consumed by probe packets. Probing for such a long time runs the risk of inducing thrashing, which is the subject of the next section.

4.2 High Loads

As discussed in Section 2.2.3, under high loads the system can enter a thrashing regime where few flows are admitted. We consider a load model similar to the one considered above, except that the arrival rate is now 1 flow per second, 3.5 times what it was before, representing a total load of approximately 400% of the link capacity (yielding blocking rates around 75%.) We use this scenario to investigate the impact of thrashing in a real environment, and so evaluate the effectiveness of slow-start probing in allevi-

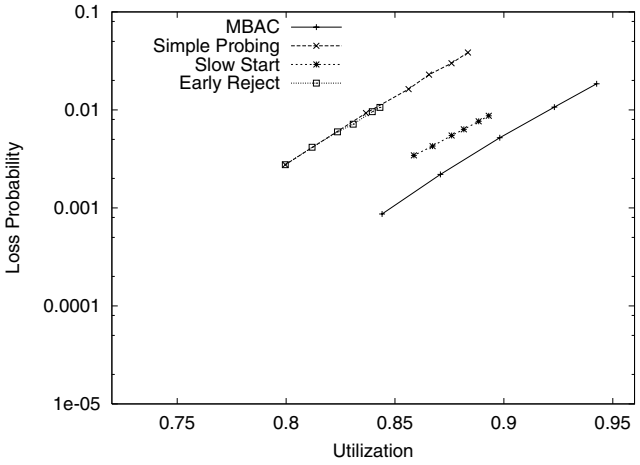


Figure 6: High Load: in-band marking

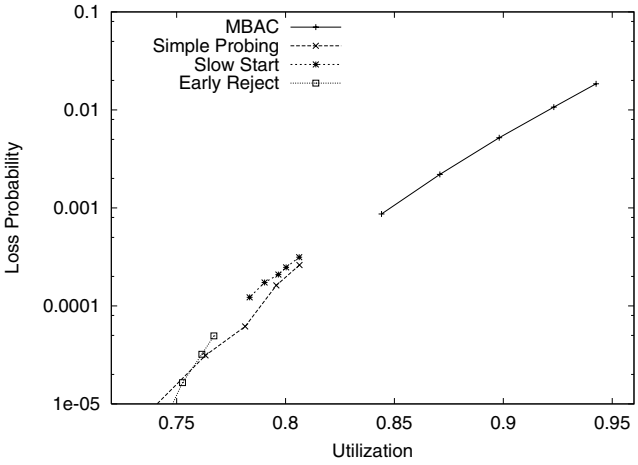


Figure 7: High Load: out-of-band marking

ating it. Figures 4-5 show the loss-load curves of the two prototype designs that use dropping, each with three different probing algorithms: slow-start, simple probing, and early reject. Loss-load curves for the reference MBAC are also shown.

Consider first Figure 4 which shows these three probing algorithms for the in-band dropping design. Note that simple probing and early reject have very similar frontiers, with early reject having somewhat lower loss rates for the same values of ϵ. However, both of these frontiers are substantially worse than that of the benchmark MBAC algorithm; the loss rates are roughly ten times larger at equivalent levels of utilization. The slow-start frontier is much closer to the MBAC, only a factor of two worse. Note that the loss rates achieved by the probing algorithms are all rather similar, the only difference is that slow-start keeps the utilization level higher for the same value of loss. In this sense, slow-start has achieved its goal of minimizing thrashing by not allowing the incoming probe traffic to prevent admissions.

Figure 5 shows the results with out-of-band dropping. The three probing algorithms have extremely similar loss-load frontiers, and they are quite close to that of the MBAC. Slow-start continues to achieve higher utilizations than the other two probing algorithms, but here it comes at the expense of higher loss rates. This is consistent with the theory we described in Section 2.2.3; with out-of-band probing there was no loss due to thrashing, only starvation under heavy load. Here with a more varied and realistic load model, we see that the loss versus utilization curves are unaffected by which probing algorithm is used (reflecting that thrashing itself causes no additional loss) but that slow-start is capable of minimizing starvation. When comparing in-band marking to out-of-band marking in Figures 6 and 7, one sees a similar difference between thrashing collapse and thrashing starvation.

In all of the simulations that follow we use the slow-start form of probing because of its apparent benefits.

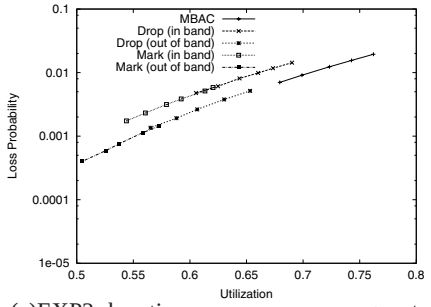
4.3 Robustness

We now subject our designs to a much wider set of load patterns. We use the additional source models described in Table 1. These include loads with burstier sources, bigger sources, LRD traffic, trace driven traffic, heterogeneous traffic, and low degree of multiplexing (where the link is only 1 Mbps). The loss-load curves are shown in Figure 8. We don't discuss each graph in detail, but will briefly summarize the results. In each graph the endpoint admission designs produce loss-load frontiers that are reasonably close to the MBAC benchmark. The in-band dropping design consistently has the highest dropping rates, but in each case it is able, for $\epsilon = 0$ to achieve a dropping rate of roughly 2% or less. The out-of-band marking design always produces the lowest dropping rates. Typically in-band marking had a lower dropping range than out-of-band dropping, but the magnitude of this gap varied widely between the different scenarios. Figure 8(a) depicts a somewhat exceptional case, where the in-band frontiers were substantially worse than the two out-of-band designs. This source model has a higher token rate, r. Therefore, it probes the network at a higher rate, resulting in a higher fraction of network bandwidth devoted to these probe packets.

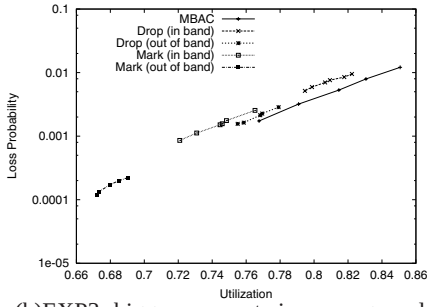
As we discussed in Section 2, and as we will elaborate on below, the ϵ values will likely have to be a uniform standard. That means that endpoints cannot adjust their ϵ to achieve the desired loss rate in a particular scenario. To what extent can the loss rate be predicted, or at least bounded, by the choice of ϵ? This depends on how widely the loss values vary for a fixed ϵ.[16] Figure 9 shows the resulting loss rates for each algorithm for a fixed ϵ; $\epsilon = 0.01$ for the in-band designs and $\epsilon = 0.05$ for the out-of-band designs. The point here is not to compare across designs (since these losses occur at different utilizations) but to note the variation in loss rates within a design. The loss rates show significant variation, at least an order of magnitude in every case. This suggests that it will be hard to provide any meaningful *a priori* predictions about likely loss rates.

In all but one case the maximal loss occurred with low multiplexing, which is consistent with the intuition that effective probing is aided by having smooth aggregate traffic. The maximal loss was greater than ϵ for in-band probing for the low multiplexing and high load scenarios. For the other three designs the maximal loss rate was less than ϵ. Thus, for these designs ϵ serves as a crude upper bound on the likely loss rate if one needed to characterize the maximal loss rates for a statistical real-time service (if the probe times are long compared to ϵ^{-1}). As observed in [7], these bounds tend

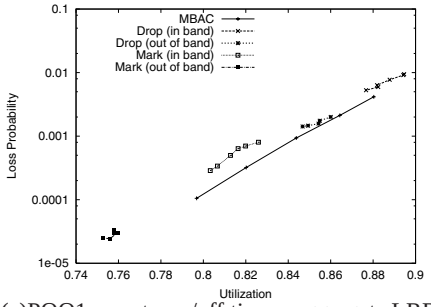
[16] Note we are not asking about utilization variation, since what an individual flow cares about is its own quality of service.



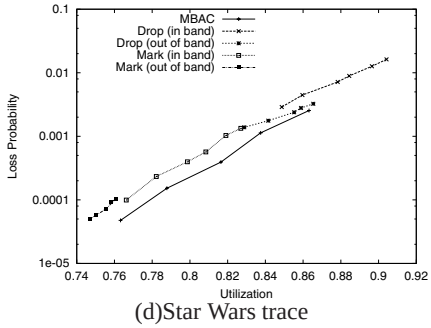
(a)EXP2: burstier source, same average rate as basic scenario



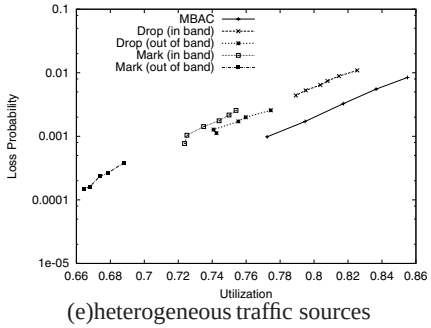
(b)EXP3: bigger source, twice average and burst of basic scenario



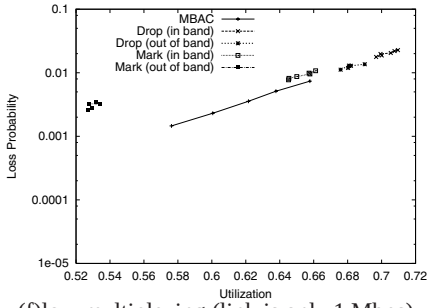
(c)POO1: pareto on/off times, aggregate LRD traffic



(d)Star Wars trace



(e)heterogeneous traffic sources



(f)low multiplexing (link is only 1 Mbps)

Figure 8: Robustness Experiments

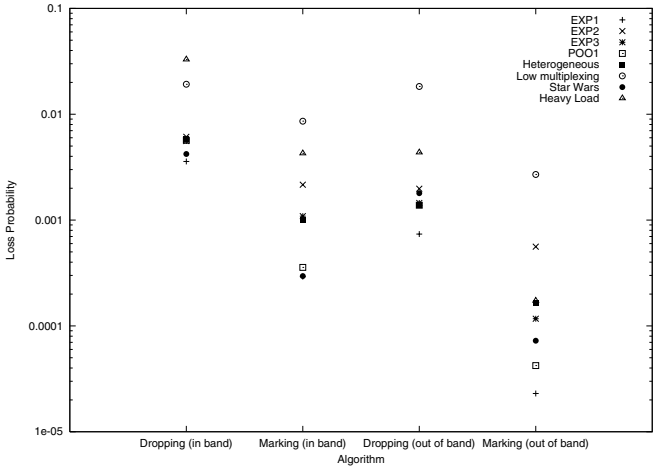


Figure 9: Loss percentages for many scenarios with fixed ϵ

to be quite conservative for the out-of-band designs.

So far we have presented a fairly broad set of simulations. We now address a few more specific performance issues.

4.4 Heterogeneous Thresholds

The discussion in Section 2 indicated that if the ϵ values were uniform, and thus static, it would be hard to control the level of service in a predictable manner. We now test the assumption that the thresholds should be uniform. We considered the same load model as in the basic scenario but with two classes of flows, those with $\epsilon = 0$ and those with larger ϵ ($\epsilon = 0.05$ for the in-band designs and $\epsilon = 0.20$ for the out-of-band designs). Table 3 shows the resulting blocking probabilities of the two classes of flows; their packet drop rates are the same, since once the flows are admitted they share the same service class. Simulations at higher offered loads resulted in very similar ratios of the blocking rates. This shows that indeed that lowering the ϵ in an attempt to increase one's quality of service merely leads to a higher blocking rate. Conversely, if all other flows were using some threshold ϵ then another flow could use a value of 2ϵ to lower its blocking probability without sacrificing its quality of service. However, when all flows followed suit and raised their thresholds the resulting quality of service would degrade. This is exactly in parallel with the tragedy-of-the-commons we find with the current congestion control paradigm.

Design	Low ϵ	High ϵ
In-band dropping	.238	.134
In-band marking	.332	.206
Out-of-band dropping	.315	.192
Out-of-band marking	.362	.284

Table 3: Blocking probabilities for low and high ϵ's: For each design, the low value is $\epsilon = 0$. For the in-band designs the high value is $\epsilon = 0.05$, and for the out-of-band designs the high value is $\epsilon = 0.20$.

4.5 Heterogeneous Traffic

The load model used in Figure 8(e) is a combination of four traffic sources. Three of them are described by the same token bucket rate r, and so their admission control probing rates are the same (and hence they have the same blocking rate). The fourth flow has a token bucket rate that is 4 times that of the other flows. When it probes, it is presumably more likely to incur losses and will be less likely to gain admission. As noted in [4] traditional MBAC admission control has a similar tendency to discriminate against bigger flows. Table 4 presents the blocking probabilities of the large and small flows in this scenario. The MBAC level of discrimina-

Design	Small Flows	Large Flows
In-band dropping	.200	.457
In-band marking	.329	.429
Out-of-band dropping	.278	.524
Out-of-band marking	.321	.490
MBAC	.156	.624

Table 4: Blocking probabilities for the large and small flows with heterogeneous traffic.

Design	Short Flows	Long Flows
In-band dropping	0.0039	.011
In-band marking	0.0003	0.0007
Out-of-band dropping	0.0006	0.0021
Out-of-band marking	0.000015	0.000044
MBAC	0.0013	0.0045

Table 5: Loss probability for the long and short flows: The loss probabilities of the short flows at each of the three congested links are averaged together. The data is for $\epsilon = 0$. The relevant comparisons are between the loss probabilities for the short and long flows for a given algorithm; the parameter values are not tuned to give the various designs equivalent loss rates.

tion against big flows is significantly higher than that of any of the endpoint admission control algorithms. This derives from the less precise nature of the endpoint algorithms. The MBAC sees a much larger sample (actually all) of the packets traversing the link, and therefore has a much more accurate estimate of the current load on the link. Thus, it is aware of how much spare capacity exists, and under high load it will often only have capacity to admit a small, and not a large, flow. Edge admission control algorithms, on the other hand, make decisions based on a smaller sample of packets. Since their decisions can not be as accurate, they will be more likely to admit a large flow, even if capacity does not exist, and reject a small flow when there is spare capacity. Thus, edge admission control actually alleviates to a degree the problem of discrimination present in traditional admission control algorithms.

4.6 Multi-hop

All of our simulations so far have been on a single link topology. What happens on multiple links? We use the topology depicted in Figure 10 to answer this question. Some flows take the three hop route along the linear backbone, while others follow the paths that traverse the backbone only for one hop. Only the three links on the backbone are congested. Thus, some flows must fight for admission across a path with multiply congested links, while others – the cross traffic – only have to contend with a single congested link. We address two issues here.

First, it is reasonable to ask if the endpoint probing paradigm is viable over multiple hops. One might imagine that the probing signal is degraded by traversing multiple congested links, leading to improper admission decisions. Our simulations do not reveal any sign of this. Table 5 shows the loss probabilities of the short and long flows with $\epsilon = 0$. The loss probabilities of the long flows are closely approximated as three times the loss percentage of the short flows. Thus, while long flows inherently will experience higher drop rates because they are traversing multiple hops, it appears the longer path does not impair the accuracy of the admission decision.

Second, as described in [4], traditional MBAC admission control

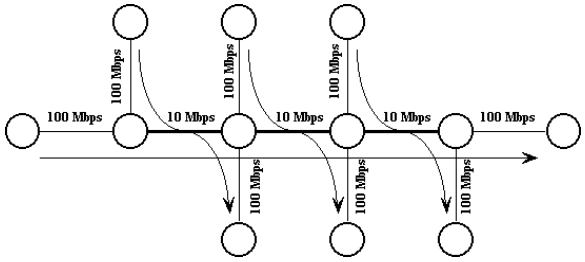


Figure 10: Simulation topology for multiple link scenario

discriminates against the multi-hop flows. If admission control decisions were purely uncorrelated and if the probability of acceptance at each hop was a then cross traffic flows would be accepted with probability a while the multi-hop flows would be accepted with probability a^3. The question we have is whether endpoint admission control experiences more severe discrimination against multi-hop flows. Table 6 shows the blocking probabilities for the two classes of flows. The MBAC blocking probability is well modeled by the product approximation. The marking designs are also well modeled by this approximation, but the dropping designs discriminate more severely against the long flows. This may not be a serious issue in practice, given that it will be rare to be traversing many congested links and, if it happens, the resulting discrimination is not drastically worse than that of the MBAC.

4.7 Incremental Deployment

The previous simulations are all based on the universal deployment of, at the very least, a separate DiffServ class for this endpoint admission-controlled traffic. If indeed endpoint admission control is deployed, its deployment will be incremental and there will be cases where admission-controlled traffic traverses legacy routers that do not have a separate DiffServ class set aside for this admission controlled traffic. At these routers, the admission-controlled traffic will share the queue with best effort traffic. In this last scenario, we see how bandwidth is shared between admission-controlled traffic and TCP traffic at such a legacy router. Figure 11 shows the aggregate bandwidth over 10 second intervals used by the TCP flows, for several values of the parameter ϵ. 20 TCP flows are started at time zero, and admission-controlled traffic begins 50 seconds later. These simulations use TCP Reno and drop tail routers, which are both currently widely deployed. For small ϵ the loss induced by the TCP flows prevents the admission-controlled flows from being admitted to the network and using any significant level of bandwidth. For higher values of ϵ the bandwidth is shared fairly equally between the two classes. Similar results were obtained when we reversed the starting order between the TCPs and the admission-controlled flows. Additional runs with varying numbers of TCP flows shows that there is typically critical value for ϵ such that above that value both kinds of traffic receive a significant share of the bandwidth, and below that value the TCP traffic dominates. As the number of TCP flows increases, the higher the critical value.

In none of our tests did the admission-controlled traffic take (on average) substantially more than 50% of the link; admission-controlled traffic was either rejected due to the background loss induced by TCP or it shared the bandwidth reasonably with TCP. However, we did not test this under a wide range of conditions. More extensive testing would be needed to ensure that the conclusions reached here are indeed more generally valid. Of particular interest are the

Design	Short Flows I	Short Flows II	Short Flows III	Long Flows	Product
In-band dropping	.202	.210	.210	.601	.508
In-band marking	.262	.267	.247	.611	.593
Out-of-band dropping	.317	.286	.275	.717	.646
Out-of-band marking	.331	.333	.359	.732	.711
MBAC	.307	.259	.286	.646	.633

Table 6: Blocking probabilities for the long (i.e., multi-hop) and short (i.e., single hop) flows: The blocking probabilities of the short flows at each of the three congested links are listed separately. The last column indicates the blocking probabilities that would result from assuming that the acceptance probability for a long flow is the product of the acceptance probabilities at each hop. The data is for $\epsilon = 0$. The relevant comparisons are between the product approximation and the actual blocking probability of the long flows; comparing the absolute blocking probabilities between the MBAC and the endpoint designs is misleading because the parameter values are not tuned to give equivalent loss rates.

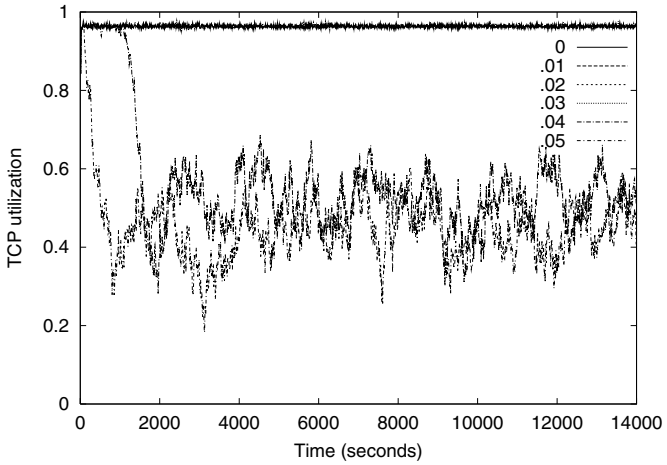


Figure 11: TCP utilization in the presence of admission-controlled traffic for different values of ϵ. The two lower curves are for $\epsilon = 0.04$ and $\epsilon = 0.05$.

resulting bandwidth shares when flows have different round trips times and when conditions are not stationary. Verifying that these results hold more generally would be necessary (but not sufficient) to determine whether it might be safe to deploy endpoint admission control algorithms in hosts even before the routing infrastructure universally supports a separate DiffServ class for that traffic.

5. SUMMARY

Several papers have proposed endpoint admission control as a possible alternative to IntServ for supporting soft real-time services such as Controlled-Load. Rather than using complicated and non-scalable signalling protocols, endpoint admission control combines sophisticated host algorithms with a rather traditional best-effort infrastructure modified only to give two additional priority levels. Our goal here was to first understand some of the basic architectural issues involved, and then to evaluate the performance of various design options for endpoint admission control. We want to stress that the credit for inventing this approach lies elsewhere (with [5, 7, 13, 15]) and that our purpose was to do a more thorough architectural and performance analysis.

We divide the rest of our concluding comments into *which* and *whether*.

Which: Our architectural discussion suggested that there were only two basic design decisions: should we mark or drop to indicate congestion, and should we probe in-band or out-of-band. The simulation results indicate that the marking and probing out-of-band loss-load frontiers are not always superior to that of in-band dropping. However, the key advantage of both marking and probing out-of-band, and particularly their combination, is that they consistently achieve lower loss rates for a given length of probing. Note that we are concerned primarily with loss, not utilization, here. That is because the utilization figures only indicate the fraction of the allocated share that the design achieves. If at a particular router the level of utilization is too low (and in none of our experiments was the achieved utilization less than 50%), one could always just increase the allocated share to increase the level of admission-controlled traffic. In addition, the leftover bandwidth is not going to waste, it is being used by the best-effort traffic.

Choosing between the designs presents us with the typical complexity versus performance tradeoff: marking[17] and out-of-band probing both entail additional mechanism, but they also allow us to achieve lower loss rates. One of the fundamental quandaries in choosing between the various designs is deciding what loss rates the Internet should attempt to support for admission-controlled traffic. This question is well outside the scope of our paper, but it holds the key to whether the less complex in-band dropping design would ever be deemed acceptable.

However, before deciding *which* endpoint admission control design to adopt, one should first ask *whether* to adopt one at all.

Whether: Endpoint admission control certainly has its flaws. The set-up delay is substantial, on the order of seconds, which may limit its appeal for certain applications. The utilization and loss rate can degrade somewhat under sufficiently high loads even with slow-start probing. The quality of service is not predictable across settings. While these performance problems are not insignificant, we suspect there are two far greater barriers to adoption.

First, as of yet we have no proposed mechanism to enforce the uniformity of the admission thresholds, or even to enforce the use of admission control at all in this service class. That is, users could send packets with the appropriate admission control DS field without using admission control.[18] A similar problem is faced by our current best effort congestion control paradigm, where users can

[17] Recall our discussion in Section 3.1 where we contended that the real complexity for out-of-band marking was the virtual queue, as one could easily achieve exactly the same results doing out-of-band *virtual* dropping instead of out-of-band marking.

[18] This is equivalent to using a threshold of $\epsilon = 1$, which is why we related it to the problem of setting the thresholds uniformly.

currently send best-effort traffic without using any congestion control. However, there are at least proposed solutions to the best-effort problem (e.g., penalty boxes, Fair Queueing like scheduling algorithms) whereas for admission-controlled traffic this is a completely open problem.

Second, we must continue to explore how one could deploy endpoint admission control incrementally. The simulations in Section 4.7 showed that in the limited scenarios tested endpoint admission control does the right thing at legacy routers by either equitably sharing the bandwidth with TCP flows or surrendering gracefully. However, more extensive simulations are needed to see whether this holds under a wider set of reasonable operating conditions.

Thus, there are significant design and deployment challenges that remain to be addressed. We urge further research in this arena, since the stakes are quite high. Endpoint admission control represents a radical and welcome departure from the complexities of the decade-long IntServ effort, and offers a much more scalable and deployable approach to support soft real-time services.

6. REFERENCES

[1] S. Berson and S. Vincent. Aggregation of internet integrated services state, 1997. Internet Draft, draft-berson-classy-approach-01.txt.

[2] G. Bianchi, A. Capone, and C. Petrioli. Throughput analysis of end-to-end measurement-based admission control in IP. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000.

[3] S. Blake et al. An architecture for differentiated services, 1998. Internet RFC 2475.

[4] L. Breslau, S. Jamin, and S. Shenker. Comments on the performance of measurement-based admission control algorithms. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000.

[5] C. Cetinkaya and E. Knightly. Egress admission control. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000.

[6] C. Courcoubetis, G. Kesidis, A. Ridder, and J. Walrand. Admission control and routing in ATM networks using inferences from measured buffer occupany. *IEEE Transactions on Communications*, 43(2):1778–1784, 1995.

[7] V. Elek, G. Karlsson, and R. Ronngren. Admission control based on end-to-end measurements. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000.

[8] T. Ferrari, W. Almesberger, and J. L. Boudec. SRP: a scalable resource reservation protocol for the Internet. In *Proceedings of IWQoS '98*, pages 107–116, Napa, CA, May 1998.

[9] S. Floyd. Comments on measurement-based admissions control for controlled-load services, July 1996. Lawrence Berkeley Laboratory Technical Report.

[10] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, Aug. 1999.

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

[12] M. Garret and W. Willinger. Analysis, modeling and generation of self-similar VBR video traffic. In *Proceedings of ACM SIGCOMM '94*, pages 269–280, London, UK, Aug. 1994.

[13] R. Gibbens and F. Kelly. Distributed connection acceptance control for a connectionless network. In *Proceedings of ITC '99*, Edinburgh, UK, June 1999.

[14] S. Jamin, S. Shenker, and P. Danzig. Comparison of measurement-based admission control algorithms for Controlled-Load Service. In *Proceedings of IEEE INFOCOM 1997*, Apr. 1997.

[15] F. Kelly, P. Key, and S. Zachary. Distributed admission control. To appear in *IEEE Journal on Selected Areas in Communications*, 2000.

[16] D. Mitzel, D. Estrin, S. Shenker, and L. Zhang. A study of reservation dynamics in integrated services packet networks. In *Proceedings of IEEE INFOCOM '96*, San Francisco, CA, Mar. 1996.

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

[18] P. Pan, E. Hahne, and H. Schulzrinne. BGRP: A framework for scalable resource reservation, 2000. Internet Draft, draft-pan-bgrp-framework-00.txt.

[19] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ECN) to IP, 1999. Internet RFC 2481.

[20] I. Stoica and H. Zhang. Providing guaranteed services without per flow management. In *Proceedings of ACM SIGCOMM '99*, Cambridge, Massachusetts, Sept. 1999.

[21] L. Westberg, Z. Turanyi, and D. Partain. Load control of real-time traffic, 2000. Internet Draft, draft-westberg-loadcntr-03.txt.

[22] J. Wroclawski. Specification of the controlled-load network element service, 1997. Internet RFC 2211.

[23] H. Zhang and D. Ferrari. Rate-controlled service disciplines. *Journal of High Speed Networks*, 3(4):389–412, 1994.

[24] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 7(5):8–18, Sept. 1993.