Adaptive Packet Marking for Providing Differentiated Services in the Internet

Wu-chang Feng† Dilip D. Kandlur‡ Debanjan Saha‡ Kang G. Shin†

†Department of EECS
University of Michigan
Ann Arbor, MI 63130
Phone: (313) 763-5363 Fax: (313) 763-4617
{*wuchang,kgshin*}*@eecs.umich.edu*

‡Network Systems Department
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
Phone: (914) 784-7194 Fax: (914) 784-6205
{*kandlur,debanjan*}*@watson.ibm.com*

Abstract

This paper examines the use of adaptable priority marking for providing soft bandwidth guarantees to individual connections or connection groups over the Internet. The proposed scheme does not require resource reservation for individual connections and can be supported with minimal changes to the network infrastructure. The scheme uses modest support from the network in the form of priority handling for appropriately marked packets and relies on intelligent transmission control mechanisms at the edges of the network to achieve the desired throughput levels. The paper describes the control mechanisms and evaluates their behavior in various network environments. The mechanisms described have the flexibility to adapt to different network environments, thereby making them suitable for deployment in an evolving Internet.

Keywords: Quality-of-service, TCP, Integrated services, Internet

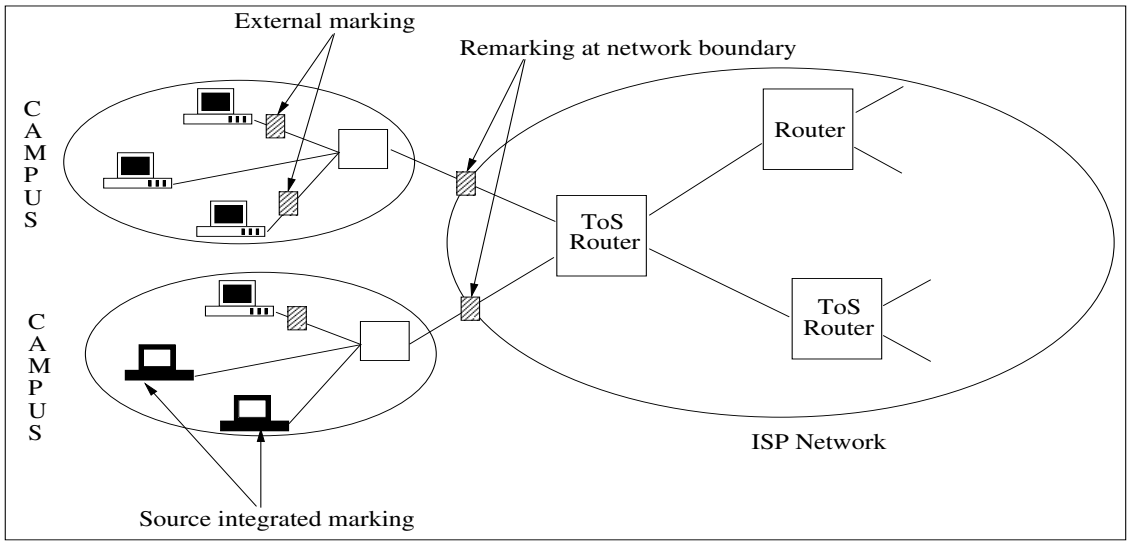


Figure 1: Packet Marking Scenarios

1 Introduction

It is a common perception in the Internet community that the best-effort service model of today's Internet is not rich enough for the diverse set of users and applications it intends to support. In an attempt to enrich this service model, the Internet Engineering Task Force is considering a number of extensions that permit the allocation of different levels of service to different users. One of the outcomes of this effort is RSVP [4, 21], a signaling protocol for resource reservation. RSVP and its associated suite of service classes [19, 20] can be used to offer service discrimination for delay sensitive applications by explicit allocation of resources in the network.

While the deployment of an RSVP-based quality of service infrastructure is underway, a more evolutionary approach to provide service differentiation in the Internet is also under consideration [5, 13]. The crux of this approach is to create service classes with different priorities using either the type-of-service (ToS) feature of IPv4 [1, 17, 18] or the priority bits of IPv6 [7, 8]. Depending on how packets with different priorities are handled by the network, a priority scheme usually translates into higher achieved throughput for higher priority classes and can be a useful building block for explicit service differentiation. One of the major attractions of priority schemes is their simplicity. When coupled with appropriate end-to-end control mechanisms, they can potentially offer the desired service levels without the overhead of explicit signaling and per-flow state management. The main drawback of priority schemes is that they do not provide users with a precise guarantee of network behavior.

In this paper, we explore how network support for type-of-service can be used in conjunction with control mechanisms at the edges of the network to achieve per-flow quality of service. In particular, we target applications which require a minimum rate of service in order to function properly. We consider a service model that is a modest enhancement to the best-effort services provided by today's Internet. More specifically, we assume that the network supports two classes of service: (1) priority service, and (2) best-effort service, which are represented using a single priority bit in the IP header. As with any type of differential service mechanism, we assume that the network provides incentives that would prevent users from continually requesting the highest

level of service. Usage-based pricing is an example of one such incentive mechanism. Many Internet service providers, such as UUNet, PSINet, MCI, already provide services wherein users are charged based on link utilization measured over five minute intervals. It is quite simple to extend this pricing model to two levels of priorities with higher prices for the high priority traffic. Network connections would then use priorities judiciously based on application requirements and usage policies.

As shown in Figure 1, packets can be marked to reflect their service priorities at different points in the network. The objective of packet marking engines located at the host-network boundaries is to help users achieve a desired level of service using an optimal mix of priority and best-effort packets. The marking engines placed at the network-network boundaries are mainly responsible for enforcing service contracts between different networks. In this paper, our objective is to develop marking algorithms and study their interaction with the transmission control mechanisms used at the source. More specifically, we examine how TCP sources can be used in conjunction with adaptive packet marking engines to maintain target service levels for individual connections as well as connection groups.

In our model, the user specifies a desired minimum service rate for a connection or a connection group. At any point of time, the sender, in cooperation with the packet marking engine, tries to achieve and potentially exceed the requested minimum service rate without using the high priority service. If however, it fails to achieve the minimum target rate, the packet marking engine starts prioritizing packets until the observed service rate reaches the desired target rate. Once the target is reached, it strives to reduce the number of priority packets without falling below the target. When the number of priority packets comes down to zero, the source tries to increase its share of the best-effort bandwidth.

We consider marking mechanisms of two distinct flavors: (1) where packet marking is completely independent of and transparent to the traffic source, and (2) where the marking module is integrated with the transmission control mechanisms at the source. Sections 3 and 4 examine these two approaches in detail. To address the robustness of the algorithms, we evaluate their performance under a variety of situations. Section 5 investigates the performance of the priority marking schemes when the network is over-subscribed and when the network contains non-responsive flows. Section 6 addresses the pragmatics of deploying the proposed mechanism in the Internet. In particular, we consider scenarios where only some parts of the network support service differentiation and propose mechanisms for detecting and reacting to such situations at the source. Finally, we discuss issues in interoperating ToS-based mechanisms with alternative mechanisms [3, 6, 12] for supporting service differentiation in the Internet.

2 Service Model

In this section, we present a detailed description of our service model and architecture. As mentioned earlier, our objective is to provide service discrimination in the network without using explicit reservation of resources. We assume a network infrastructure where the routers and gateways provide only modest functionality to support service discrimination, namely appropriate handling of multi-priority traffic. There is no guaranteed service level associated with a priority class, although a higher priority generally translates into a better quality of service. In this paper, we assume that there are only two levels of service: (1) priority service, and (2) best-effort service. The two service classes differ only in the loss rates they experience in the network. We also assume that the traffic types are carried in the type-of-service (ToS) bits in the IP header. For reasons of simplicity, we refer to the ToS bits in the headers of best-effort IP traffic as the default and denote them as unmarked packets. Consequently, we refer to the priority traffic as marked traffic.

One way of providing different services to marked and unmarked packets is to maintain separate queues for each class and to serve them according to their scheduling priority. We propose, however, to use a common queue for both marked and unmarked traffic and to serve them in FIFO order. A common FIFO queue not only simplifies the scheduling mechanism at the router, it also helps maintain packet ordering. Although maintaining packet ordering is not a requirement at the IP layer, failure to do so may have serious performance impact on transport protocols such as TCP. Our approach to service differentiation between marked and unmarked packets relies on a selective packet discard mechanism. We use an enhanced version of the RED (Random Early Detection) algorithm [2, 11] for this purpose. In classical RED routers, a single FIFO queue is maintained for all packets. Packets are dropped randomly with a given probability when the queue length exceeds a certain threshold. The drop probability itself depends on the queue length and the time elapsed since the last packet was dropped. Enhanced Random Early Detection (ERED) is a minor modification to the original RED algorithm. In ERED, the drop probabilities of marked packets are considerably lower than that of unmarked packets.

Given this service model, we examine the use of various packet marking schemes which can be deployed at the host-network or network-network boundaries and which will allow an individual connection or a group of connections to achieve a specified service level by the appropriate use of priority marking. For example, a user may request a specific target rate for a particular connection or an aggregate rate for a group of connections. The objective of a packet marking engine is to monitor the throughput received by the connection or connection group and appropriately adjust the packet marking so that the target rate is maintained. Due to the particular nature of the service model, at times it may not be possible to sustain the requested target rate because of the over-commitment of resources. Such lapses may also be caused by partial deployment of IP type-of-service or heterogeneity in network services. A significant part of our effort goes into detecting such cases and taking appropriate actions whenever required.

It is possible that packet marking be performed at multiple points in the network to enforce different policies. Consider the scenario described in in Figure 1. Marking engines at host-network boundaries may mark packets at certain rates in order to achieve target throughputs for specific connections or connection groups. Packets may then be remarked at the network-network boundaries to enforce the service agreements different networks. In this paper, we consider scenarios where packets are marked only once. The impact of packet remarking is under investigation and will be addressed in future work.

We consider two different places for packet marking: (1) external to the host, and (2) within the host. In the first case, the packet marking engine, which we call a packet marking gateway (PMG), is completely independent of the source. In this case, the marking policy and mechanism are entirely decoupled from the flow and congestion control mechanisms used at the source. The PMG is responsible for monitoring the throughput seen by the TCP source and marking packets with higher priority, when required, in order to achieve the user specified minimum service rate. Having the marking module be transparent and external to the source has several deployment benefits. First, it does not require any changes to host protocol stack and can be integrated into the infrastructure without affecting other hosts and routers. Second, it can be used to provide service guarantees not only to individual connections, but also to a group of connections. On the other hand, integrating the packet marking scheme with the host protocol engine can provide a solution that adapts better with the flow and congestion control mechanisms used at the transport layer. One disadvantage of using an external marking module is that at times it may mark more packets than required. When the marking module is integrated with the TCP source, it can control TCP windowing mechanisms for optimal performance.

```
Every update interval:
        scale = | 1 − obw/tbw |
        if (obw < tbw)
                mprob = mprob + scale * increment
        else
                mprob = mprob − scale * increment
```

Figure 2: TCP independent algorithm

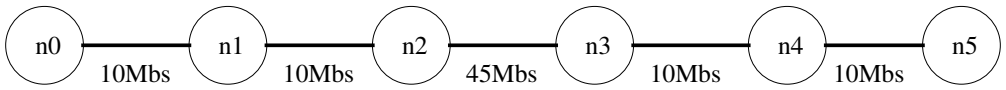


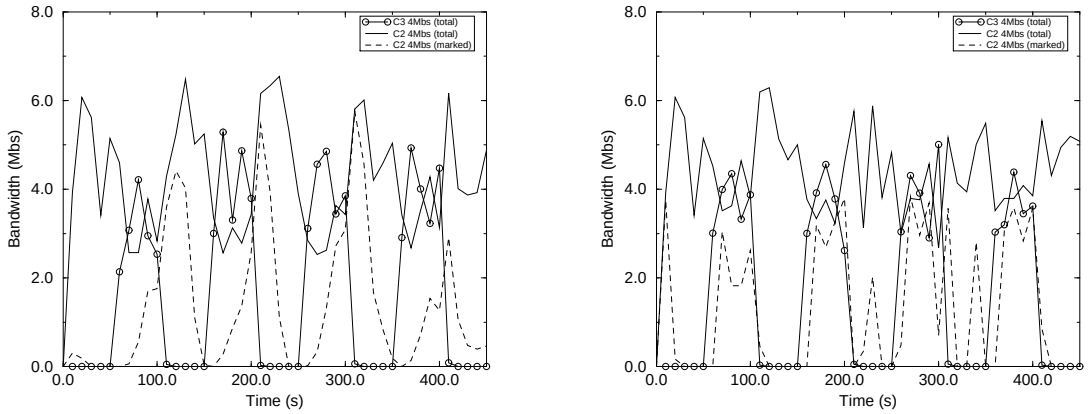
Figure 3: Network topology.

3 Packet Marking Gateway

A PMG snoops on connections passing through it and measures their observed throughputs. If the measured throughput is sufficiently close to the requested target rate, it takes the role of a passive monitor. However, if the observed throughput of a connection is lower than its requested target, the PMG takes a more active role and starts marking packets belonging to the connection or connection group. The fraction of marked packets varies from 0 to 1 depending upon the measured and target throughputs. Selective marking essentially upgrades a fraction of the packets belonging to the connection to the higher priority level. The PMG constantly adjusts the fraction of packets to be marked in order to sustain a bandwidth which is close to the requested target rate, while keeping the number of marked packets as low as possible.

One of the important tasks performed by a PMG is measuring the throughput seen by connections passing through it. This is fed into the packet marking process that has to adapt to the changes in observed throughput caused by variations in network load. While the overall measure of network performance from an application's point of view is goodput, the PMG used in our experiments only measures the local bandwidth that is consumed by a connection. The gateway counts bandwidth against a connection or connection group when it either sends (or receives) a packet from it, even though the packet may be dropped later on in the transit path. One of the reasons for measuring local throughput, instead of end-to-end goodput, is simplicity. The PMG does not have to understand the transport layer protocol semantics in order to determine whether or not the application's data was actually delivered. In some cases, even if the PMG is well aware of the transport layer semantics, it may not have access to the stream of acknowledgments from the receiver to compute goodput. This may be the case when the forward and the return paths of connections are different.

The local throughput seen by a connection at the PMG can be measured in several ways. One simple technique is to measure the amount of data transferred with a sliding window and to use the average bandwidth received over this window as a measure of the observed bandwidth. If



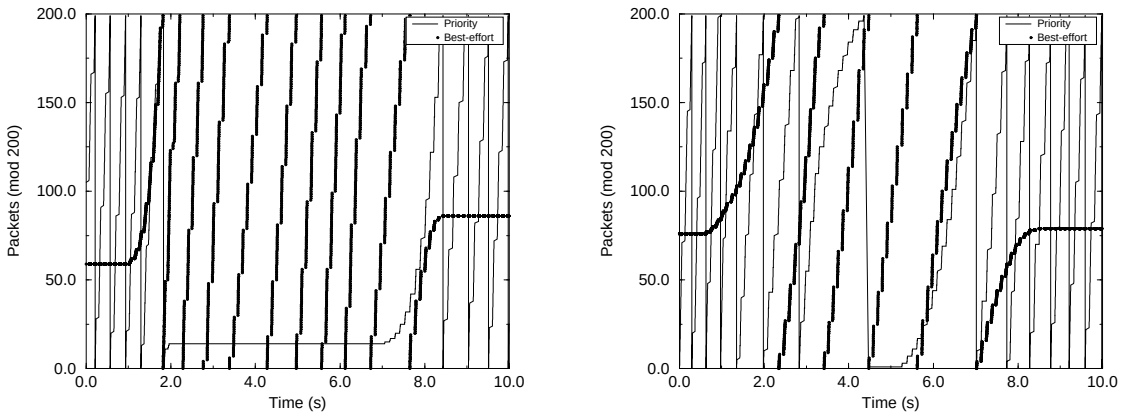
(a) Marking probability increment = 0.01 (b) Marking probability increment = 1.00

Figure 4: Effect of external packet marking.

the window is small, the measured throughput is biased towards the more recent observations. If window is large, the computed throughput converges to the long-term average bandwidth seen by the connection. While this is a fairly accurate and tunable measure of the observed throughput, it requires a window's worth of information stored for each connection. For the experiments reported in this study, we use a lightweight alternative mechanism. We measure throughput seen by a connection over a small time window. We then compute the observed bandwidth as a weighted average of this measured throughput and the current value of observed bandwidth.

The most important task of a PMG is to adaptively adjust the packet marking rate based on the measured throughput. In this paper, we consider a probabilistic marking scheme where the packets are marked randomly as they pass through the PMG . The marking probability ($mprob$) is periodically updated depending on the observed bandwidth (obw) and the corresponding target bandwidth (tbw). Figure 2 shows a simple algorithm designed for this purpose. As can be seen from the algorithm, when the observed bandwidth is less than the target bandwidth, the packet marking probability is incremented in steps. Similarly, the marking probability is decremented in steps if the observed throughput exceeds the target rate. Note that both increments and decrements in marking probability are scaled by the difference between observed and target throughputs. That is, the changes in the marking probability get smaller as the observed bandwidth nears the target bandwidth. This scaling damps the amplitude of oscillations of the marking probability.

In order to understand the effect of packet marking, we simulated a simple scenario using the `ns` [16] network simulator. As shown in Figure 3, the simulated network consists of six nodes, *n0* through *n5*, and five links connecting them. Each link is labeled with its respective link bandwidth and transmission delay. The queues in the routers are ERED queues with min_{th} of 10 packets, max_{th} of 80 packets, and an initial drop probability of 0.05 for unmarked packets. Marked packets have a drop probability two orders of magnitude less than unmarked packets, but use the same threshold values. Additional experiments using ERED queues with separate threshholds for priority and best-effort traffic were also performed and showed similar results. We simulate three connections between nodes *n0* and *n5*: an infinite best-effort TCP connection (*C1*), a second infinite TCP connection (*C2*) with a $4Mbs$ target bandwidth, and a third TCP connection (*C3*) that toggles between on and off



(a) Marking probability increment = 1.00 (b) Marking with TCP-like algorithm

Figure 5: Burstiness observed using packet marking schemes

states every 50 seconds, but has a throughput requirement of $4Mbs$ when it is on. We assume that the observed throughputs and marking probabilities are updated every $100ms$.

In this network configuration, when only *C1* and *C2* are active, the bottle link bandwidth of $10Mbs$ is shared evenly between them and thus, no packet marking is required for *C2* to achieve its target of $4Mbs$. However, when *C3* is active, an even share of the bottleneck bandwidth ($3.33Mbs$) does not satisfy the target throughput requested by *C2* and *C3*. The PMG has to mark packets belonging to *C2* and *C3* in order for them to obtain the higher throughput.

Figure 4(a) shows the throughputs received by *C2* and *C3* over time. In this experiment, the marking probability is adjusted in steps of 0.01. As the figure shows, *C2* is slow in reacting to changes in the network. When all of the sources are on, it is consistently below its $4Mbs$ target bandwidth. It takes a significant amount of time to build up the marking probability in response to the changes in the network load. Figure 4(a) also shows the marking rate for connection *C2*. As expected, the marking rate lags behind the changes in the network load, slowly rising in response to an increased traffic load and slowly falling in response to a decreased traffic load. To experiment with the other end of the spectrum, we repeated the experiment allowing the marking probability to be updated in steps of 1.0. That is, when more bandwidth is needed, all packets are marked. Otherwise, packet marking is turned off. Figure 4(b) shows the results from this experiment. As expected, in this experiment, the packet marking probability adapts very quickly to the changes in the network load, thus allowing *C2* to achieve its target rate even during the periods of increased traffic load. This rapid response also allows the PMG to turn off the packet marking quickly when it detects that the available bandwidth is sufficient to satisfy the target rate. While adapting quickly to changes in network conditions has its benefits, it can also cause significant burstiness in both marked and unmarked packet streams. For example, if packet marking is turned on for a connection with a relatively high target throughput, it may cause large spikes in the number of marked packets in the network. Similarly, when packet marking is turned off, a spike of unmarked packets may be injected into the the network. Figure 5(a) shows a sample packet trace of a connection using this algorithm. The figure plots the number of marked and unmarked packets sent. As the figure shows, as soon as the connection reaches its target, the PMG quickly cuts down the number of

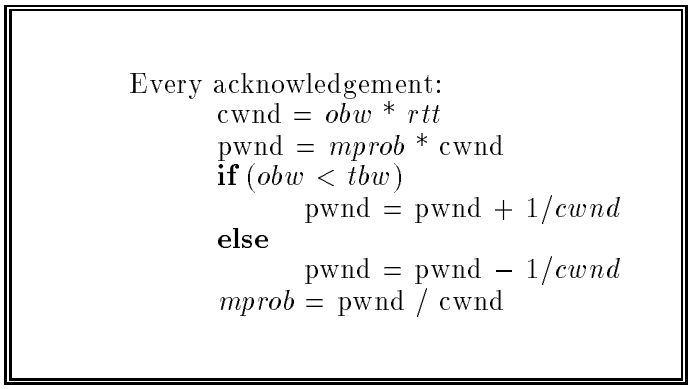
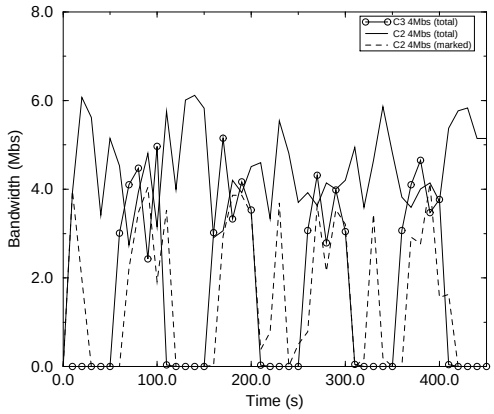


Figure 6: TCP-like algorithm for changing marking probabilities

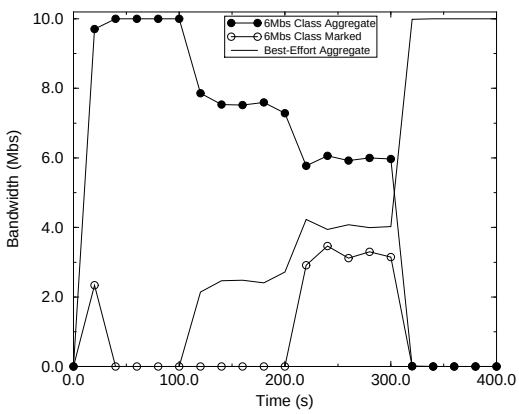
marked packets sent and starts sending a large amount of unmarked packets. In the simulations performed, we do not observe any significant impact from the bursts of marked and unmarked packets. This is due to the fact that the TCP congestion control algorithm controls the combined stream of marked and unmarked packets in a very network-friendly fashion. The use of a common queue for marked and unmarked packets also adds to stability. When the PMG changes the marking probability in large steps, the overall impact is a mere replacement of marked packets by an equal number of unmarked packets or vice versa. However, in situations where not all of the sources use TCP or where not all queues are ERED queues, large swings in the number of marked and unmarked packets can potentially lead to network instability.

In order to minimize the chances of triggering such instability in the network, the PMG should update marking probabilities in a manner that is more network-friendly, while maintaining the ability to react to the changes in network load. To address the potential shortcoming of the algorithm presented in Figure 2, we experimented with an algorithm (shown in Figure 6) that updates the marking probability in a more network-friendly manner. It draws on the windowing mechanisms used in TCP and tries to ensure that the number of marked (or unmarked) packets in the network increases by no more than 1 per round-trip time. This is in some sense similar to the linear increase algorithm for congestion avoidance used by TCP [14]. As shown in Figure 6, we compute an estimated number of marked packets in flight (pwnd) by taking the estimated congestion window given as the product of the observed bandwidth and the estimated round-trip time (rtt) and multiplying it by the marking probability. At every update epoch, if the observed bandwidth is less than the target rate, pwnd is incremented linearly ($1/cwnd$). This ensures that the number of marked packets increases by no more than one in every round-trip time. Similarly, when the observed bandwidth is higher than the target rate, the decrease in the number of marked packets (and hence increase in the number of unmarked packets) is limited to one every round-trip time. Figure 5(b) shows the packet trace of the modified scheme. Unlike the previous trace, this time the connection slowly increases and decreases the number of marked and unmarked packets sent.

Figure 7(a) shows the result from the same experiment with the PMG implementing the packet marking algorithm presented in Figure 6. As seen from the graph, the marking algorithm is very reactive to changes in the network load and hence observed throughput. Consequently, connection *C2* maintains an average throughput at or above its $4Mbs$ target most of the time. However, it changes the marking probability in a more network-friendly fashion and reduces the risk of network



(a) Transient Experiment



(b) Aggregation Experiment

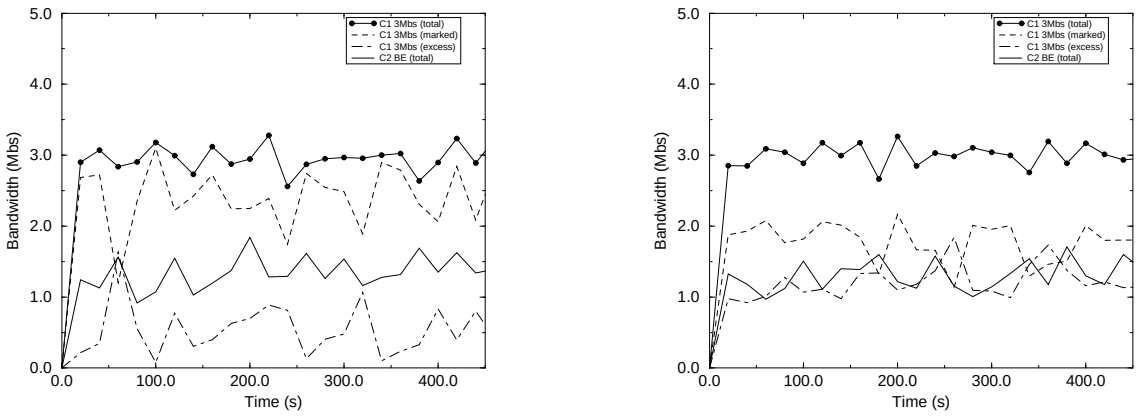
Figure 7: Performance of TCP-like algorithm.

instability.

While these experiments show how per-connection target throughputs can be achieved, the PMG can also be used to meet the throughput target of an aggregation of connections. As in the case of individual connections, it simply monitors the throughput of the connection group and adjusts the marking rate based on the observed throughput and requested target. Figure 7(b) shows the results of an experiment where a PMG is used to control two sets of connections sharing a $10Mbs$ bottleneck link. The first set of connections requires at least $6Mbs$ of bandwidth at all times while the other set is simply treated as best-effort. In this simulation, there are 3 identical connections in the first set and 4 identical connections in the second set. Initially, only the three connections of the first set are active. Thus, the aggregate bandwidth seen is the entire link bandwidth with each source receiving a third of the bandwidth. Note that the marking rate for the connection group is zero as there is enough bandwidth available to meet the target service level. At $t = 100s$, one best-effort connection is started. Since an even split of the bandwidth gives each connection approximately $2.5Mbs$, the three connections in the first set get a total of $7.5Mbs$ without any packet marking. At $t = 200s$, the other three best-effort connections are started. In this case, an even split of the bandwidth across all connections is not sufficient to sustain the target rate of $6Mbs$ for the first set. Thus, the PMG begins to mark packets in order to sustain the target rate of $6Mbs$. As the figure shows, the marking increases to a level sufficient to maintain the target rate. The best-effort connections then get an equal share of the leftover $4Mbs$. Finally, at $t = 300s$, all connections of the first set are terminated. As the figure shows, the best-effort connections get the entire $10Mbs$ with each getting a fair share of it.

4 TCP Modifications

One of the problems with having the marking module external to the source is that it has little control on the flow and congestion control mechanisms excercised by the source. This lack of control can have detrimental impact on performance. For example, while the packet marking gateway is fairly effective in maintaining the observered throughput close to the target bandwidth, it often



(a) Packet marking gateway with TCP-like algorithm (b) TCP with windowing modifications.

Figure 8: Bandwidth sharing example

marks more packets than required. In an ideal scenario, a connection that stripes its packets accross two priorities should receive a fair share of the best-effort bandwidth in addition to the bandwidth received due to priority packets. A TCP source oblivious of the packet marking module fails to compete fairly with best-effort connections for its share of best-effort bandwidth. Consequently, the marking module marks more packets than it should have, had the connection received its fair share of the best-effort bandwidth.

In Figure 8(a) we present results from an experiment that substantiates this conjecture. In this experiment, we spawn connection *C1* with a target bandwidth of $3Mbs$, and 5 best-effort connections (*C2*, *C3*, *C4*, *C5*, *C6*) between nodes *n0* and *n5*. Figure 8 shows the marking rate, the best-effort bandwidth, and the total bandwidth received by *C1* along with the total bandwidth received by *C2*, one of the 5 identical best-effort connections. As shown in the figure, *C1* gets a much smaller share of the best-effeort bandwidth than *C2*. Thus, it must mark a larger portion of its packets than it should in order to maintain the desired level of performance.

This phenomenon can be easily explained if we take into account the windowing mechanism used by TCP. The congestion window of *C1* essentailly consists of two parts: (1) a priority window consisting of priority packets, and (2) a best-effort window consisting of best-effort packets. Assume that the size of the priority window is w_p and that of the best-effort window is w_b. The best-effort connections should also have congestion windows of size w_b. When congestion is detected in the network, ideally, *C1* should cut its best-effort window in half resulting in a window size of $w_p + \frac{w_b}{2}$. However, since the source has no knowledge of the priority marking and service differentiation, the window is cut to $\frac{w_p+w_b}{2}$ instead. In other words, by marking packets the PMG puts *C1* at a disadvantage in competing for the best-effort bandwidth. A similar bias is also seen when TCP opens its congestion window.

In order to address this problem, we experimented with a packet marking scheme that is integrated with the TCP sender. Figures 9 and 10 show the new algorithm. In this scheme, the congestion window (*cwnd*) maintained by a TCP source is split into two parts: (1) a priority window (pwnd) which is a measure of the number of marked packets that are in the network, and (2) a best-effort window (bwnd) that reflects the number of unmarked packets that are outstanding.

```
After every acknowledgment (opencwnd)
        pwnd = mprob *cwnd
        bwnd = (1-mprob )*cwnd
        if (obw < tbw)
                if (pwnd < pssthresh ) pwnd = pwnd + pwnd/cwnd
                else pwnd = pwnd + 1/cwnd
                if (bwnd < bssthresh ) bwnd = bwnd + bwnd/cwnd
                else bwnd = bwnd + 1/cwnd
        else
                if (pwnd > 0)
                        if (bwnd < bssthresh ) pwnd = pwnd - bwnd/cwnd
                        else pwnd = pwnd - 1/cwnd
                else
                        if (bwnd < bssthresh ) bwnd = bwnd + bwnd/cwnd
                        else bwnd = bwnd + 1/cwnd
        if (pwnd < 0) pwnd = 0
        cwnd = pwnd + bwnd
        mprob = pwnd/cwnd
```

Figure 9: Customized TCP congestion window opening.

```
After every segment loss from dupack (closecwnd)
        pwnd = mprob *cwnd
        bwnd = (1-mprob )*cwnd
        if (priority loss)
                cwnd = cwnd /2
                pssthresh =mprob *cwnd
                bssthresh =(1-mprob )*cwnd
        else
                bwnd = bwnd/2
                bssthresh = bwnd
                cwnd = pwnd + bwnd
                mprob = pwnd/cwnd
```

Figure 10: Customized TCP congestion window closing.

Upon a loss, the sender determines whether the lost packet was sent as a marked or an unmarked packet. The loss of a marked packet is an indication of severe congestion in the network. Consequently, both the priority and best-effort windows are reduced. However, the loss of an unmarked packet is an indication of congestion potentially only in the best-effort service class and hence only the best-effort window is backed off. The procedure for opening the congestion window is also

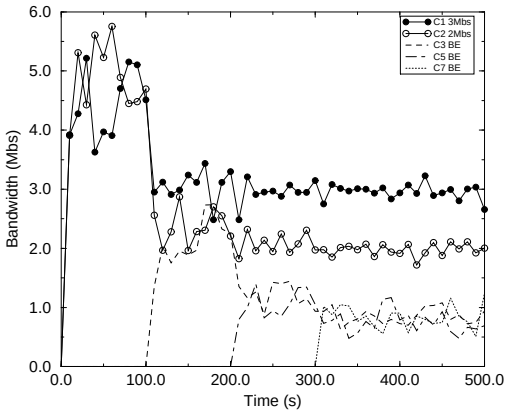
modified. The connection keeps track of two additional thresholds values, namely *pssthresh* and *bssthresh* which are updated whenever the connection experiences a priority and a best-effort loss, respectively. When a connection is below its target bandwidth, it opens up both the priority and best-effort windows. If either one of the windows is below its respective threshhold (*pssthresh* and *bssthresh*), it is in the slow start mode. Note that the increases are scaled so that the overall congestion window does not grow any faster than that in an unmodified TCP. Scaling these increases is slightly conservative, since it temporarily hinders the source from growing its best-effort window as quickly as other best-effort sources. However, the conservative behavior aids in avoiding congestion collapse scenarios. When either window is above its threshhold, it increases linearly (i.e. one segment per round-trip time). Note that while *cwnd* grows by two segments every round-trip time, the best-effort part of the window (bwnd) only grows as quickly as the *cwnd* of a best-effort connection. While this modified windowing algorithm is essential in obtaining a fair share of the best-effort bandwidth in a network that supports service differentiation, it essentially behaves like two fairly independent connections. In a network that does not support end-to-end service differentiation, a TCP source modified in this manner may receive twice as much bandwidth as compared to unmodified TCP sources. We discuss additional modifications to address this aspect in Section 6. Figure 8(b) shows results from the experiment presented in Figure 8(a) using the algorithm described above. In contrast to Figure 8(a), the amount of best-effort bandwidth received by the $3Mbs$ source closely matches the bandwidth received by the best-effort sources.

To further examine the issue of fair bandwidth sharing, we took a closer look at the packet marking rate and its deviation from the theoretically computed optimal marking rate. The computation of ideal marking rates is quite straightforward. For example, suppose we have a network with a bottleneck link of bandwidth B. Assume that n connections with target rates of R_i, $i = 1, 2, \ldots, n$, are passing through it. Let r_i be the optimal marking rate of the connection with a target rate of R_i, and let b be share of best-effort bandwidth received by all connections. A connection j with $R_j < b$, is essentially a best-effort connection with $r_j = 0$. The following set of equations capture the system constraints.

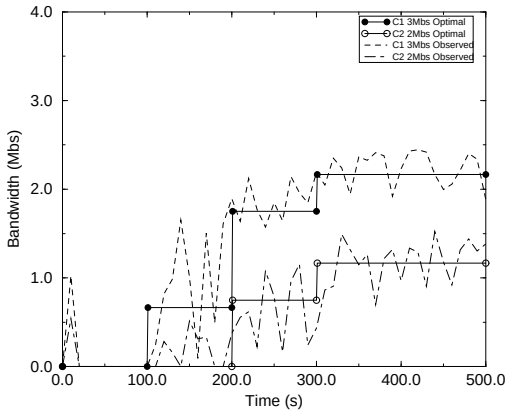
$$\begin{aligned} r_i + b &= R_i \\ \textstyle\sum_{i=1}^{n} r_i + nb &= B \end{aligned}$$

Figure 11 shows the results of an experiment with two connections *C1* and *C2* with target rates of $3Mbs$ and $2Mbs$, respectively, and six best-effort connections sharing a bottleneck link of $10Mbs$. The connections *C1* and *C2* start at time $t = 0s$, followed by two best-effort connections at $t = 100s$, another two at $t = 200s$, and the last two at $t = 300s$. Figure 11(a) shows the bandwidth received by *C1* and *C2* and three of the best-effort connections. Figure 11(b) shows the marking rate of both *C1* and *C2*, as well as their calculated ideal marking rates. At time $t = 0s$, when only two connections are on-line, a fair split of the bandwidth satisfies target rates of both *C1* and *C2*. Thus, neither source marks any of their packets and each gets approximately half of the bottleneck bandwidth. At $t = 100s$, two best-effort connections are added. At this point, *C1* needs to mark at a $0.67Mbs$ rate and each of the sources should get $2.33Mbs$ of the excess best-effort bandwidth. Since *C2*'s share of best-effort bandwidth is more than its target rate, it need not mark any of its packets. As Figure 11 shows, the marking rate and total bandwidth graphs reflect the change.

At $t = 200s$, two more best-effort connections are added. Now, *C1* has to mark at a rate of $1.75Mbs$ while *C2* needs to mark at at a rate of $0.75Mbs$. This leaves each source $1.25Mbs$ of the excess bandwidth. As the total bandwidth graph shows, the best-effort connections get about $1.25Mbs$ while *C1* and *C2* get their respective target bandwidths. The marking rates of *C1* and *C2* also adapt to this change, increasing to the optimal marking rates. Finally, at $t = 300s$, the

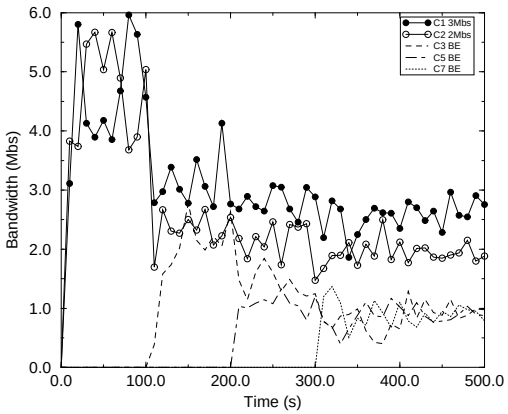


(a) Total Bandwidths

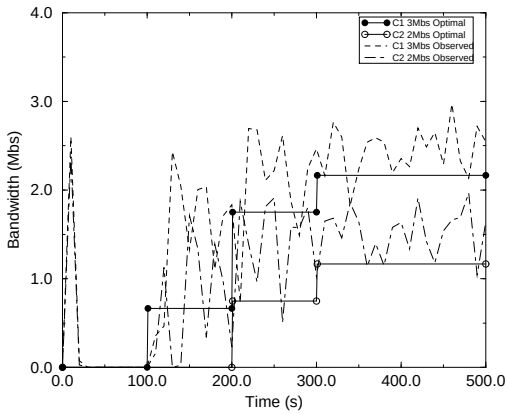


(b) Marking Rates

Figure 11: TCP with modified windowing



(a) Total Bandwidths

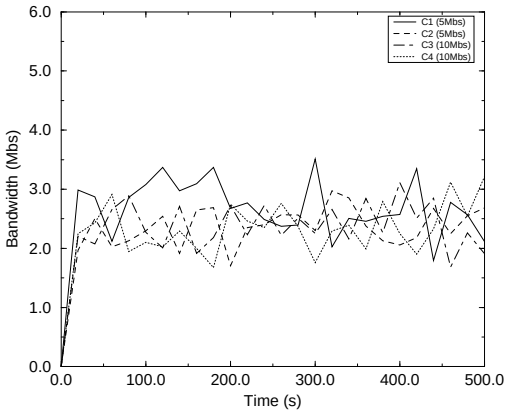


(b) Marking Rates

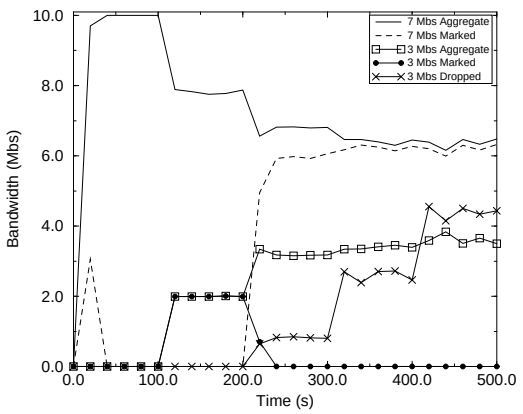
Figure 12: TCP with PMG

last two best-effort sources are added. This time, *C1* needs to mark at $2.17Mbs$ while *C2* needs to mark at $1.17Mbs$. Each connection now gets $0.83Mbs$ of the excess bandwidth. Again, as the graphs show, both the priority and best-effort connections perform as expected.

To examine the impact that the windowing modifications have, we performed the same set of experiments with a PMG. Figure 12 shows the total bandwidth and marking rate for different connections. Since TCP windowing algorithms restricts the connections *C1* and *C2* from competing for the excess bandwidth, the PMG consistently overmarks its packets as shown in Figure 12(b). Increased marking can potentially lead the ERED queue to be full of marked packets, making it behave more like a regular RED queue. As Figure 12(a) shows, loss of priority packets causes periods of time where throughputs of connections *C1* and *C2* drop significantly below their target



(a) Over-subscription using modified TCP



(b) Non-responsive flows across a PMG

Figure 13: Over-subscription and non-responsive flows

rates.

5 Handling Over-subscription and Non-responsive Flows

One of the key advantages of using an adaptive packet marking scheme is that it obviates the need for a signaling protocol. However, since there is no resource reservation, the service guarantees it provides are necessarily soft. In an RSVP-based network, when demand for differential services continually exceeds the supply, admission control is used to deny additional connections access to the services in order to maintain the service levels of the current set of connections. In ToS based networks where no reservations or admission control is in place, the network must instead offer degraded service when the demand exceeds the supply. In both cases, pricing and access policies in conjunction with capacity planning must be used to balance the supply and the demand of resources.

This section describes how over-subscription is handled in our service model. When considering packet marking gateways, over-subscription can be strictly avoided by making sure that the marking bandwidth allocated between connections or sets of connections does not exceed the individual link bandwidths in the network. In the absence of such an allocation, there are several ways that bandwidth can be distributed amongst the competing connections or connection groups. One way is to simply revert back to best-effort service.

The mechanisms presented in this paper can provide best-effort bandwidth sharing when demand exceeds supply. When demand exceeds supply, all connections with non-zero target rates carry only marked packets. Consequently, they only compete for priority bandwidth and the ERED queue at the bottleneck degenerates into to RED queue serving only priority traffic. In the case of the PMG, since the underlying TCP windowing algorithm is not changed, the requested target bandwidth does not influence the throughput a source receives. As a result, each source receives an equal fair share of the bottleneck bandwidth. Over-subscription results in the same outcome when the marking module is integrated within the source. In this case, since the algorithms for growing and shrinking the priority window are independent of the bandwidth demand, the windowing al-

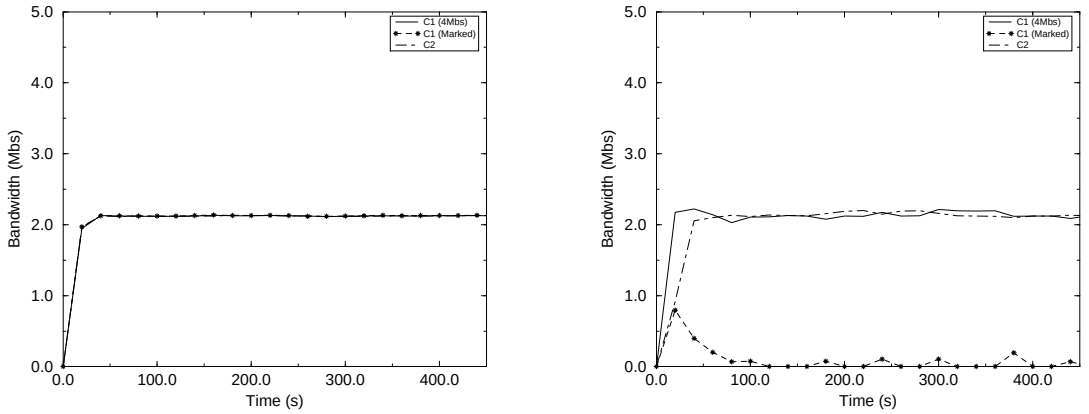
gorithm simply behaves as normal TCP. This adaptation in presence of overload prevents possible congestion collapse. Figure 13(a) shows an example scenario with four connections *C1*, *C2*, *C3*, and *C4* spanning from node *n0* to *n5*. The connections *C1* and *C2* have a target rate of of $5Mbs$ each. Connections *C3* and *C4* aim at a target rate of $10Mbs$. As the figure shows, when using the integrated marking scheme, each connection gets a fair share of the bottleneck bandwidth when the demand exceeds the supply.

Another possible way to handle situations where resources are over-subscribed, is to provide weighted bandwidth sharing depending on the target rates or the importance of the connections or connection groups. Because our schemes assume the use of a single priority bit, it cannot itself be used to provide weighted bandwidth sharing. However, it is possible to implement weighted bandwidth sharing easily through the use of additional priority levels which give the network an indication of the connection's target rate and/or importance. Future work will address such issues.

Another important issue is protection against non-responsive (non-TCP) flows [9, 15]. Such flows have the potential of stealing bandwidth away from other flows which are responsive to congestion. One of the advantages of using a PMG is that it can insulate a connection or a connection group from others. Figure 13(b) shows a scenario where three TCP connections are competing for bandwidth with a non-responsive flow across a $10Mbs$ link. In this simulation, the link uses an ERED queue with separate thresholds for marked and unmarked packets. The aggregate target rate for TCP flows is set at $7Mbs$. The target rate for the non-responsive flow is $3Mbs$. Initially, only the TCP sources are active. They compete fairly for the link bandwidth. The non-responsive flow starts transmitting at $2Mbs$ at $t = 100s$. As shown in the figure, the aggregate throughput of the TCP connections drops from $10Mbs$ to $8Mbs$ when the non-responsive flow becomes active. Note that the PMG marks all of the packets of the non-responsive flow since it is below its $3Mbs$ target. Without additional information, the PMG cannot determine the difference between an idle application or an application being throttled by congestion in the network. At $t = 200s$, the non-responsive flow increases its transmission rate to $4Mbs$, thus exceeding its allocated rate of $3Mbs$. As shown in the figure, the marking rate of this flow immediately drops to zero and the loss rate increases to approximately the difference between the transmission rate and the allocated rate. The non-responsive flow further increases its transmission rate to $6Mbs$ and $8Mbs$ at times $t = 300s$ and $t = 400s$, respectively. Each time, the throughput observed by the non-responsive flow remains fairly constant near its allocated rate of $3Mbs$, while the amount of packets which are dropped increases at the same rate as the transmission rate. Thus, the non-responsive flow gains little by sending any amount above its allocated rate. Note that the non-responsive flow does have some impact on the TCP connections. As Figure 13(b) shows, the aggregate marking rate of the TCP connections approaches the aggregate transmission rate, since the unmarked packets from these connections see the same increased drop rates as the non-responsive flow. In order to provide fairness between connections competing for best-effort bandwidth, it is possible to extend ERED queues with additional fairness mechanisms [15].

6 Deployment Issues

The Internet is a highly heterogeneous and slowly evolving networking environment. It is impractical to assume that all routers in the Internet will handle priority packets in the same way. As a matter of fact, it is quite likely that only a fraction of them will support service differentiation between packets of different priorities. In order to be successful in this environment, it is important that the packet marking scheme proposed in this paper is capable of handling heterogeneity in the network. More specifically, it should be able to operate in an environment where all routers do



(a) Packet marking gateway

(b) TCP with windowing modifications

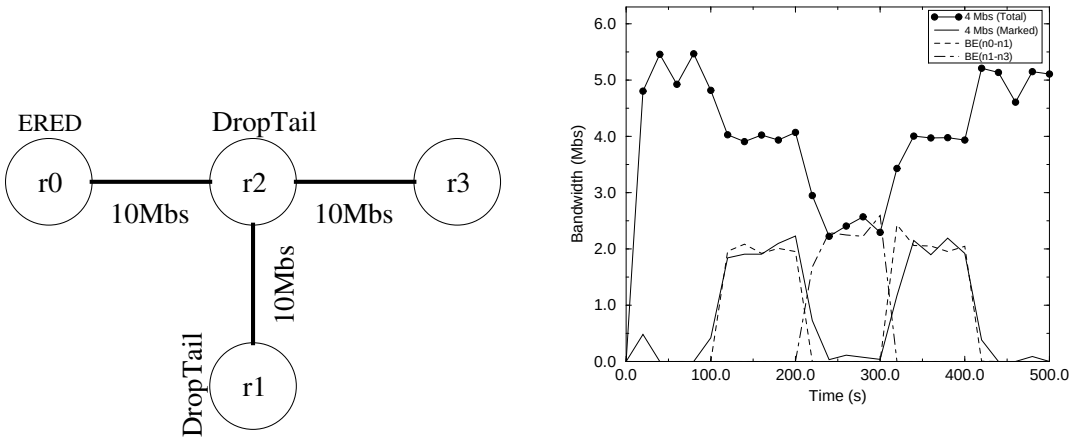
Figure 14: Performance over an all drop-tail network

not support service differentiation between priority and best-effort packets. It is also important that it interoperates with other mechanisms for handling quality of service in the Internet. In the following we discuss these deployment issues in detail.

6.1 Heterogeneous Networks

One of the salient features of the proposed scheme is its ability to perform in a network that does not provide service differentiation. When the packet marking module is external to the source, the sending TCP is unchanged. Thus, the lack of service differentiation simply makes the packet marking ineffective and the TCP sources behave as normal TCP sources in a best-effort network. When the marking module is integrated with the source, however, the situation is little different. In this case, we essentially have two connections with differing priorities. In absence of service differentiation, this scheme can potentially be twice as aggressive as a regular TCP connection. While such behavior may be justified when a user is charged for marked packets, it may be desirable to turn off marking when service differentiation is not supported by the network.

To address this, we implemented a simple mechanism for turning off the marking and modified windowing when the network does not support end-to-end service differentiation. Note that the bottleneck of a connection may shift from a link that supports service differentiation to one that does not and vice versa. Hence detection of service level on a connection path is not a one time process; it requires constant monitoring. To minimize the cost of monitoring and at the same time remain reactive to changes in the network dynamics, we use an exponential back off algorithm to determine monitoring intervals. In particular, the source keeps track of the inter-drop times for both priority and best-effort packets. In a network which supports service discrimination, the number of priority packets transmitted between successive priority packet drops is expected to be substantially greater than the number of best-effort packets transmitted between successive non-priority packet drops. When this is not the case, the source simply turns off the marking and the windowing algorithm, reverting back to normal TCP. After a preset interval, marking is turned on again and the source monitors inter-drop intervals to detect service differentiation. If it fails to



(a) Network (b) Bandwidth graph

Figure 15: Effects of heterogeneity

detect service differentiation, it shuts down marking for twice the duration it had before. If the source observes that service differentiation is supported by the network, the connection continues using the modified windowing algorithm and resets the backoff interval to its initial (smaller) value. While this mechanism successfully handles the cases when service differentiation is either supported or not supported on a connection path, it may not be as effective when the network performs some re-marking of packets. In the following we present experimental results showing the effect of network heterogeneity.

Figure 14(a) shows the throughput observed by five connections *C1*, *C2*, *C3*, *C4*, and *C5* going from *n0* to *n5* when all of the queues in network are drop-tail queues with no support for service differentiation. Connection *C1* has a target rate of $4Mbs$. All other connections are best-effort. We use a packet marking gateway to mark packets in this example. As expected, bottleneck bandwidth is shared fairly among all five connections. Note that the packets are continually being marked even though they are ignored by the network. This is because the PMG cannot determine that the marks in the packets are being ignored unless it keeps additional connection information. Since the connection is always below its target bandwidth, the PMG simply marks all of the packets. Figure 14(b) shows the same experimental setup as before. However, in this example, the packet marking module is integrated within the source. As the figure shows, *C1* backs off its marking as it detects that the network does not support any service differentiation. Thus, the connection competes fairly with all of the other best-effort connections for the excess bandwidth.

The backoff mechanisms used when the marking module is integrated into the source adapt quickly to changes in the network. This helps the source change its windowing and marking strategy as the bottleneck link shifts from non-priority to priority queues in a heterogeneous network. Figure 15(a) shows a network with 4 nodes where *r0* implements the ERED queueing mechanism while *r1* and *r2* are simple drop-tail gateways. In this network, we simulate two priority connections *C1* and *C2* with $4Mbs$ target bandwidths and several transient best-effort connections. We use the transient connections to move the bottleneck link from *r0-r2* to *r2-r3*. Figure 15(b) shows the throughputs seen by different sources as the bottleneck moves from one link to another. We start with connections *C1* and *C2* going from *r0* to *r3*. In the absence of any other connections,

they do not have to mark any of their packets in order to achieve their target rates. At $t = 100s$, a best-effort connection is spawned between *r0* and *r1*. Since a fair share of the bottleneck bandwidth of $10Mbs$ does not satisfy the target rates of connections *C1* and *C2*, they both mark their packets at a rate of $2Mbs$. From the equations outlined in Section 4, this is the optimal marking rate in this scenario. Each connection, including the best-effort connections, also receives $2Mbs$ of the leftover best-effort bandwidth. At $t = 200s$, the best-effort connection terminates and two new best-effort connections are started between nodes *r1* and *r3*. At this time, the bottleneck link is between *r2* and *r3* which happens to be a drop-tail queue with no support for service differentiation. In this case, even though *C1* and *C2* fail to sustain their target rates, they back off their marking and revert back to the original windowing algorithm. Consequently, all four connections now receive an equal share of the bottleneck bandwidth of $10Mbs$. At $t = 300s$, the best-effort connections terminate and a new best-effort connection is spawned between nodes *r0* and *r1*. At this point, the bottleneck shifts to the link *r0-r2* which supports service differentiation. This change is detected by *C1* and *C2* and they turn on marking to reach their target rate of $4Mbs$. Finally, at $t = 400s$, the best-effort connection terminates, leaving the network in its initial state. The connections *C1* and *C2* once again turn off their marking since they can support their target throughput without packet marking.

6.2 Integration with Alternative QoS Frameworks

Another important pre-requisite for successful deployment is the ability to interoperate and co-exist with other mechanisms for service differentiation. There are two distinct and separate ways of supporting service discrimination over the Internet (explicit reservation using RSVP and ToS). Here, we examine issues in having both mechanisms coexist in a particular network and in having them interoperate when different networks support different and only one of the mechanisms.

Since the ToS-based mechanisms described here rely only on preferential queueing based on the priority levels, they can be embedded without modification into any of the packet and link scheduling frameworks which have been proposed for supporting various integrated services [3, 6, 12]. Hence, co-existence of ToS and RSVP-based mechanisms is quite straight forward.

For networks in which different network segments on the path of a connection offer different types of service differentiation, mapping between priority levels and RSVP service levels is required to ensure interoperability. We first examine the case where ToS priority has to be mapped onto RSVP-based service architecture. In a multi-priority system, this mapping can be fairly involved since there is no definite way of mapping priorities into a traffic specification required to setup an RSVP connection. Handling two levels of priorities used in our model is a little easier. One possible solution is to map a mixed priority stream into a controlled-load connection. Since the traffic compliant to the *Tspec* (negotiated traffic envelope) has the same low loss semantic as the priority traffic in our scheme, priority packets can be reshaped to conform to the *Tspec* through an RSVP-style network. However, this still requires a *Tspec* capturing the characteristics of priority packets.

The reverse mapping between RSVP flows and priority levels can potentially be performed by a PMG placed at the boundary between such networks. Currently there are two services which have been proposed and advanced through the IETF: guaranteed and controlled-load. It is hard to map guaranteed service into priorities in a ToS domain. Mapping controlled-load service into priorities in ToS network is a potentially simpler task. Previous work, in fact, has shown how one could implement a controlled-load service variant [10] through use of ERED queues. In this case, the delay bounds are loosened in favor of maintaining packet ordering. One simple way of handling controlled-load flows entering a ToS based network is to simply mark all incoming packets that

are *Tspec* compliant. Note that the amount of marking in the network must still be controlled in order to provide guarantees. Failure to do so will cause the ERED queues to degenerate to regular RED queues without any support for service differentiation.

7 Conclusions

In this paper, we have proposed and analyzed adaptive packet marking algorithms for providing soft bandwidth guarantees over the Internet. We have considered marking algorithms that are external and transparent to the source, and algorithms that are integrated with the congestion and flow control mechanisms at the source. Both sets of algorithms have advantages and disadvantages from the standpoint of performance and deployment issues. The results presented in this paper clearly demonstrate that simple service differentiation, when used in conjunction with adaptive source control, can be an effective means to provide quality of service in the Internet.

This work can be extended in several ways. We are currently investigating the impact of marking packets at multiple places in the network. Also under investigation is the interaction and interoperability of the proposed schemes with alternative mechanisms to support quality of service in the Internet. Generalization of the two priority ToS scheme to multiple priorities is also under consideration.

References

[1] P. Almquist. Type of Service in the Internet Protocol Suite. *RFC 1349*, July 1992.

[2] R. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on Queue Management and Congestion Avoidance in the Internet. *Internet Draft draft-irtf-e2e-queue-mgt-00.txt*, March 1997.

[3] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: An Overview. *RFC 1633*, June 1994. ISI/MIT/PARC.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification. *Internet Draft draft-ietf-rsvp-spec-16.txt*, November 1997. ISI/PARC/IBM/UM.

[5] D. Clark. A Model for Cost Allocation and Pricing in the Internet. *MIT Workshop on Internet Economics*, March 1995.

[6] D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. In *Proc. of ACM SIGCOMM*, pages 14–26, August 1992.

[7] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. *RFC 1883*, December 1995.

[8] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. *Internet Draft draft-ietf-ipngwg-ipv6-spec-v2-00.txt*, July 1997.

[9] K. Fall and S. Floyd. Router Mechanisms to Support End-to-End Congestion Control. ftp://ftp.ee.lbl.gov/papers/collapse.ps, February 1997.

[10] W. Feng, D. Kandlur, D. Saha, and K. Shin. Understanding TCP Dynamics in an Integrated Services Internet. In *Proc. of NOSSDAV '97*, May 1997.

[11] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *ACM/IEEE Transactions on Networking*, 1(4):397–413, August 1993.

[12] S. Floyd and V. Jacobson. Link-sharing and Resource Management Models for Packet Networks. *IEEE/ACM Transactions on Networking*, 3(4), August 1995.

[13] IETF BOF Notes. Future Directions for Differential Services BOF (fddifs). April 1997.

[14] V. Jacobson. Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM*, pages 314–329, August 1988.

[15] D. Lin and R. Morris. Dynamics of Random Early Detection. In *Proc. of ACM SIGCOMM*, September 1997.

[16] S. McCanne and S. Floyd. http://www-nrg.ee.lbl.gov/ns/. ns-LBNL Network Simulator, 1996.

[17] J. Postel. Internet Protocol. *RFC 791*, September 1981.

[18] J. Reynolds and J. Postel. Assigned Numbers. *RFC 1340*, July 1992.

[19] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. *Internet Draft draft-ietf-intserv-guaranteed-svc-08.txt*, February 1997. Xerox/BBN/IBM.

[20] J. Wroclawski. Specification of Controlled-Load Network Element Service. *Internet Draft draft-ietf-intserv-ctrl-load-svc-05.txt*, May 1997. MIT.

[21] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, pages 8–18, September 1993.