Adaptive Packet Marking for Maintaining End-to-End Throughput in a Differentiated Services Internet

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

†Department of EECS
University of Michigan
Ann Arbor, MI 48109-2122
{*wuchang,kgshin*}*@eecs.umich.edu*

‡Network Systems Department
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{*kandlur,debanjan*}*@watson.ibm.com*

Abstract

This paper examines the use of adaptive priority marking for providing soft bandwidth guarantees in a differentiated services Internet. In contrast to other proposals for achieving the same objective, the proposed scheme does not require resource reservation for individual connections and can be supported with minimal changes to the network infrastructure. It 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. These mechanisms are shown to have several salient features which make them suitable for deployment in an evolving Internet.

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

1 Introduction

The current Internet offers best-effort service to all traffic. In an attempt to enrich this service model, the Internet Engineering Task Force (IETF) is considering a number of architectural extensions that permit the allocation of different levels of service to different users. One of the outcomes of this effort is an architecture that provides service discrimination by explicit allocation and scheduling of resources in the network. This model, based on the Resource Reservation Setup Protocol (RSVP) [2, 15] and its associated suite of service classes [13, 14], is the Internet incarnation of the traditional "circuit-based" quality of service architecture. While this service architecture provides a solid foundation for providing different classes of service in the Internet, it mandates significant changes to the Internet infrastructure.

The complexity of an RSVP based service architecture has led the IETF to consider other alternatives to service differentiation in the Internet [1]. The crux of this approach [1, 3, 6], known as differentiated services, is to keep the core of the network as simple as possible, pushing most of the complexity to the network edges. In a differentiated services architecture, packets are classified and marked with appropriate ToS [12] value at the edges of the network. At the network core, the routers simply support priority handling of packets based on their ToS values. While the simplicity of the differentiated services architecture is definitely appealing, it has its shortcomings. One of its major weaknesses is that it makes it difficult to provide throughput guarantees to individual flows. In this paper, our objective is to develop control mechanisms that help individual connections or connection groups to maintain end-to-end throughput in a differentiated services Internet.

We consider a network service model that is a modest enhancement to the best-effort service provided by today's Internet. More specifically, we assume that the network supports a one-bit priority scheme with lower loss rates for higher priority traffic. Similar service models have been proposed in a number of recent articles presented in the IETF [1]. We also 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 fixed time intervals. It is rather simple to extend this pricing model to levy higher prices for the high priority traffic. Such a pricing mechanism would encourage judicious use of priority service based on application requirements and usage policies. While pricing is not the focus of this study, we note that one of the key advantages of the proposed architecture is that it can provide simple mechanisms for calculating near-optimal prices based on congestion costs [10].

In our model, the user or network administrator specifies a desired minimum service rate for a connection or connection group and communicates this to a control engine located at or near the host-network interface. The objective of the control engine, which we call a *packet marking engine* (PME), is to monitor and sustain the requested level of service by setting the ToS bits in the packet headers appropriately. By default, all packets are generated as low priority packets. If the observed service rate at the low priority level either meets or exceeds the requested service rate, the PME assumes the role of a passive monitor. If however, the observed throughput falls below the minimum target rate, the PME starts prioritizing packets until the desired target rate is reached. Once the target is reached, it strives to reduce the number of priority packets without falling below the minimum requested rate.

In the rest of the paper, we assume that all connections use TCP as the transport layer protocol. Although the adaptive marking scheme does not specifically depend on TCP, it makes use of the feedback based control mechanism for the purpose of measuring the throughput seen by a connection and adjusting the marking rate accordingly. With suitable modifications the proposed scheme can be adapted to work any transport protocol that is responsive to congestion in the network.

In the following sections, we demonstrate the efficacy and the robustness of the proposed framework. Using extensive simulations, we show that the proposed architecture adapts with the traffic dynamics in the Internet to eliminate the risk of congestion collapse. When used in conjunction with intelligent queue management, it can also identify and penalize non-adaptive and/or malicious flows and hence provides sufficient incentives for applications to be well-behaved. We also address the issue of incremental deployment and discuss how the proposed architecture can be embedded in more elaborate service differentiation frameworks [4, 7].

The rest of the paper is organized as follows. Section 2 provides the background for the discussion that follows. Sections 3 and 4 examine different approaches to adaptive packet marking. Sections 5 and 6 investigate the performance of the proposed architecture in oversubscribed situations and in the presence of non-responsive flows. Section 7 addresses the pragmatics of deploying the proposed scheme in the Internet. We conclude in Section 8.

2 Type-of-Service Architecture

In this section, we present a brief overview of our service architecture. We assume a network infrastructure that supports two traffic types: priority and best-effort. We assume that the traffic types are carried in the ToS bits in the IP header and that by default, all packets are initially sent with their ToS bits cleared (best-effort). For reasons of simplicity, these packets are referred to as unmarked packets. Consequently, we refer to the priority traffic as marked traffic. While there is no guaranteed service level associated with the priority class, it is assumed that the higher priority generally translates into a better quality of service. In line with the Internet design philosophy, in our service architecture, most of the intelligence is at the edges of the network. The routers and gateways provide only modest functionality to support service discrimination, namely appropriate handling of multi-priority traffic.

There are a number of alternative approaches to providing priority services in the network. One obvious technique is to maintain separate queues for each class and serving them according to their scheduling priority. Another approach is to use a common FIFO queue for all traffic and provide service differentiation by applying different drop preferences to marked and unmarked packets. A common FIFO queue simplifies the scheduling functionality 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 impacts on transport protocols such as TCP. We take the latter approach and use an enhanced version of the RED (Random Early Detection) algorithm for providing service differentiation between priority levels. 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 lower than that of unmarked packets.

Given this service model, our goal is to develop packet marking schemes which can be deployed at the host-network interface that will allow an individual connection or a connection group to achieve a target throughput specified by the user or the network administrator. 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 the packet marking scheme is to monitor the throughput received by the connection or connection group and appropriately adjust the packet marking so that the sustained rate is maintained satisfying all the policy constraints. Due to the particular nature of the service model, at times it may not be possible to sustain the requested target rate due to over-commitment of resources. Such lapses may also be caused by partial deployment of IP type-of-service or oversubscription. A significant part of our effort goes into detecting such cases and taking appropriate actions whenever required.

In our service architecture, traffic flows are monitored and packets are marked at the host-network interface. However, the service architecture allows for packets to be re-marked at multiple points along the path in order to enforce different policies and service contracts. Consider for example, a campus or enterprise environment where applications running at different hosts may mark packets at certain rates to achieve their respective target throughputs. Packets may be re-marked at the boundary between the internal network and external network to enforce the service agreement with the network service provider. Similar re-marking may also occur in order to enforce a bilateral agreement between service providers when traffic crosses provider boundaries. While our scheme can adapt in an environment where packets are marked at multiple points, in this paper we consider scenarios where packets are marked only once. The impact of packet re-marking is under investigation and will be addressed in future work.

We consider marking mechanisms of two different flavors: (1) where the marking engine is transparent and potentially external to the host, and (2) where the marking engine is integrated with the host. In either case, the packet marking engine (PME) maintains a local state that includes the target throughput requested for a connection or a group of connections. It passively monitors the throughput or the aggregate throughput for a connection or a connection group and adjusts packet marking in order to achieve the target throughput requested by the user. Placing the PME external to the host has significant deployment benefits since it can be integrated into the infrastructure without affecting the hosts and routers. On the other hand, integrating the PME with the host protocol engine can provide a solution that adapts better with the flow and congestion control mechanisms used at the transport layer. In particular, we consider the integration of the PME and the TCP control mechanisms.

3 Source Transparent Marking

A PME 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 PME 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 PME continually adjusts the fraction of packets to be

Every update interval:
  scale = $| 1 - obw/tbw |$
  if $(obw < tbw)$
    $mprob = mprob + \text{scale} * increment$
  else
    $mprob = mprob - \text{scale} * increment$

Figure 1: TCP independent algorithm

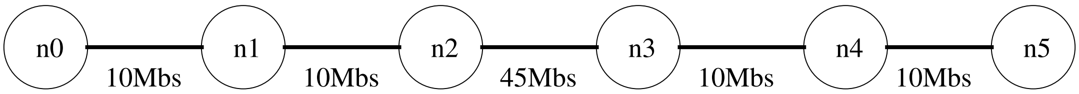


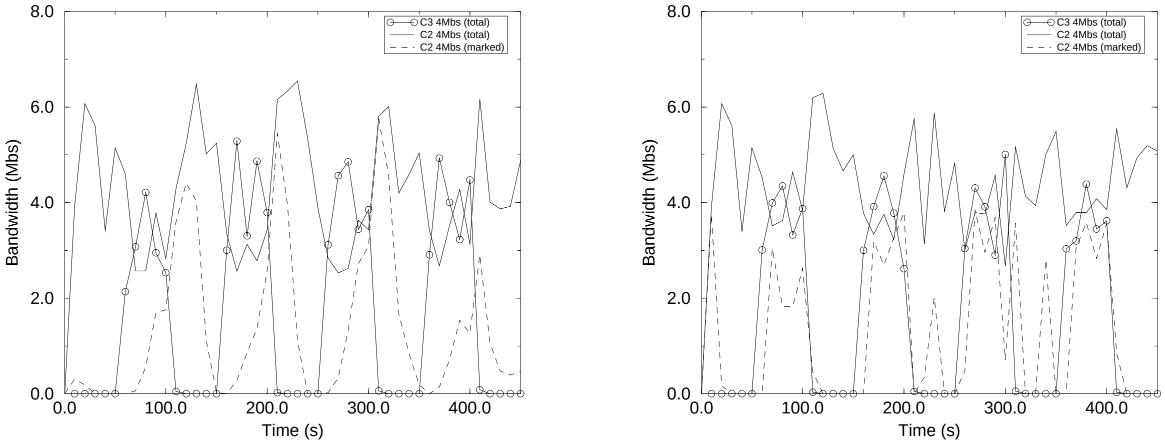
Figure 2: Network topology.

marked in order to sustain a bandwidth 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 PME 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 PME used in our experiments only measures the local bandwidth consumed by a connection. It counts bandwidth against a connection or connection group when it 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 PME 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 PME 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 most important reason for counting local throughput is to give incentive for end hosts to send packets which have a good chance of being delivered. Thus, a malicious or non-adaptive source has its packets counted against itself regardless of whether they have been delivered.

The local throughput seen by a connection 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 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 PME 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 PME . The marking probability ($mprob$) is periodically updated depending on the observed bandwidth (obw) and the corresponding target bandwidth (tbw). Figure 1 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 mark-



(a) Marking probability increment = 0.01

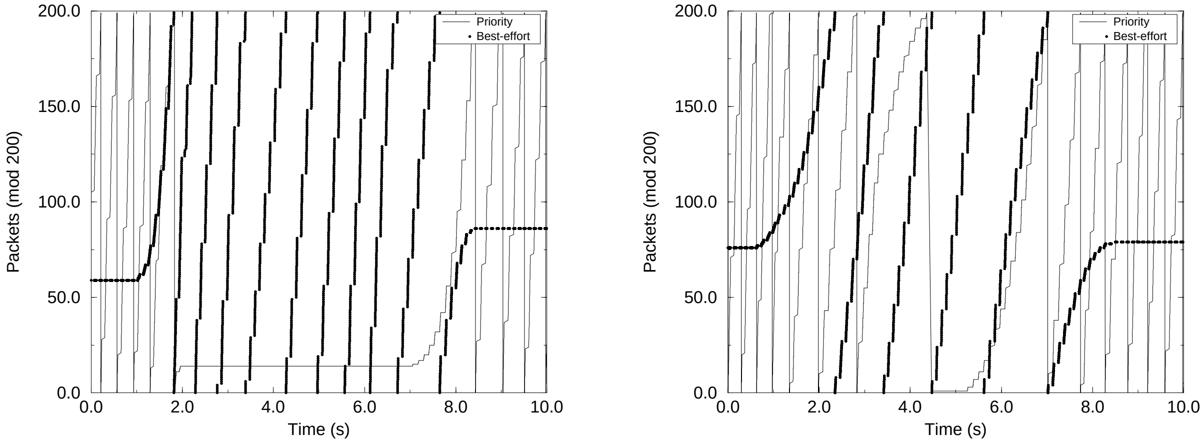
(b) Marking probability increment = 1.00

Figure 3: Effect of external packet marking.

ing 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` [11] network simulator. As shown in Figure 2, 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 has a transmission delay of $10ms$. 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 that of 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 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 bottleneck 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 PME has to mark packets belonging to *C2* and *C3* in order for them to obtain the higher throughput. Figure 3(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 3(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



(a) Marking probability increment = 1.00

(b) Marking with TCP-like algorithm

Figure 4: Burstiness observed using packet marking schemes

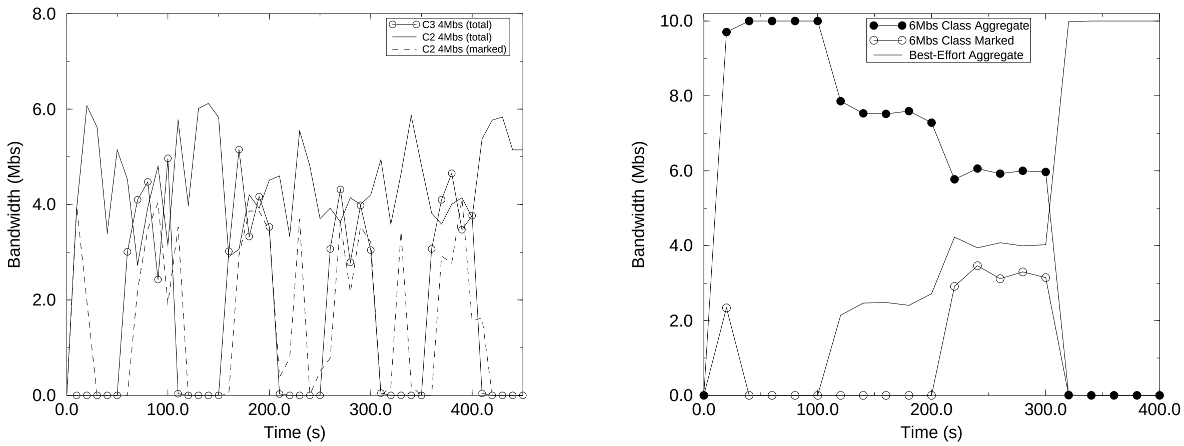
is, when more bandwidth is needed, all packets are marked. Otherwise, packet marking is turned off. Figure 3(b) shows the results from this experiment. As expected, in this experiment, the packet marking probability adapts very quickly to the changes in the network load, thus allowing *C2* to achieve its target rate even during the periods of increased traffic load. This rapid response also allows the PME to turn off 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 4(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 PME quickly cuts down the number of 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 the stability. Even when the PME 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 PME 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 1, we experimented with an algorithm (shown in Figure 5) 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 [8]. As shown in Figure 5, 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

Every acknowledgement:
    pwnd = *mprob* * (obw * rtt)
    if ($obw < tbw$)
        pwnd = pwnd $+ 1/cwnd$
    else
        pwnd = pwnd $- 1/cwnd$
    mprob = pwnd / (obw * rtt)

Figure 5: TCP-like algorithm for changing marking probabilities



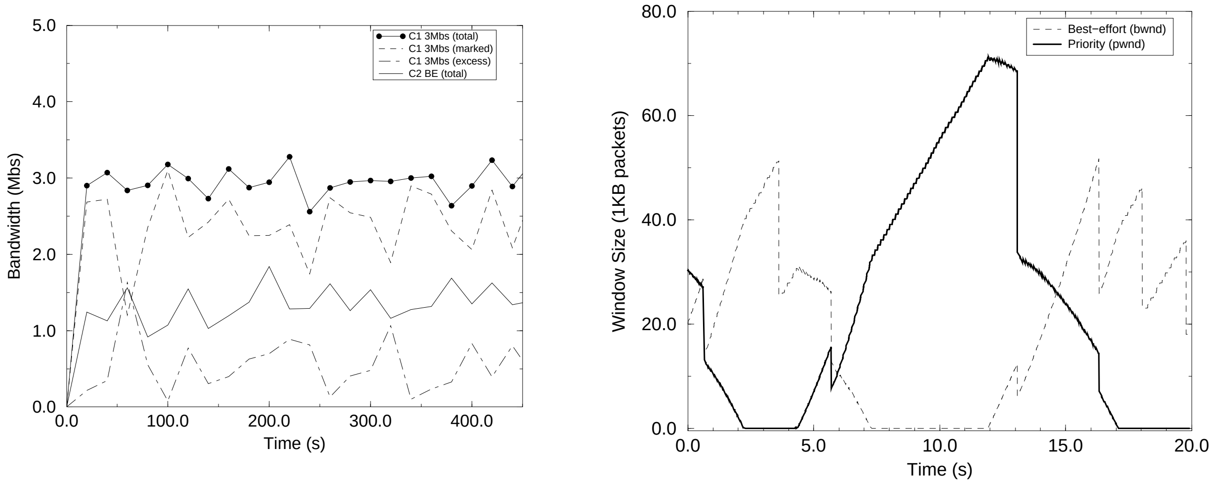
(a) Transient Experiment

(b) Aggregation Experiment

Figure 6: Performance of TCP-like algorithm.

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 4(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 6(a) shows the result from the same experiment with the PME implementing the packet marking algorithm presented in Figure 5. 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 instability.

While these experiments show how per-connection target throughputs can be achieved, PME can also 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 6(b) shows the results of an experiment where a PME controls 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



(a) Bandwidth graph

(b) Window trace of $3Mbs$ connection

Figure 7: Bandwidth sharing using source-transparent marking

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 PME 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 Source Integrated Approach

One of the problems with having the PME external and transparent to the source is that it has little control on the flow and congestion control mechanisms exercised by the source. This lack of control can have detrimental impact on performance. For example, while a source-transparent PME is fairly effective in maintaining the observed throughput close to the target bandwidth, it often marks more packets than required. In an ideal scenario, a connection that stripes its packets across 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 fails to compete fairly with best-effort connections for its share of best-effort bandwidth. Consequently, the PME marks more packets than it should have, had the connection received its fair share of the best-effort bandwidth.

Figure 7(a) presents results from an experiment that demonstrates this. 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 7 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

```
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
```

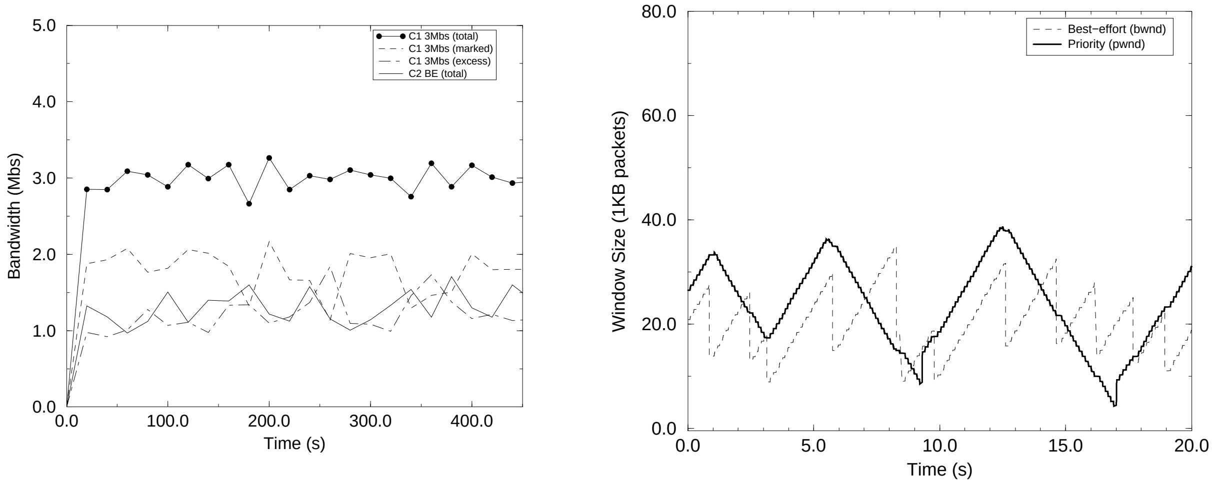
Figure 8: Customized TCP congestion window opening.

```
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 9: Customized TCP congestion window closing.

a much smaller share of the best-effort 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 examine the window trace of the $3Mbs$ connection. Figure 7(b) plots both the priority and best-effort portions of the connection's congestion window. As the figure shows, when the application requires additional bandwidth it must send priority packets *in place* of best-effort packets. Thus, when the connection sends priority packets, it cannot compete fairly for the available best-effort bandwidth.

In order to address this problem, we experimented with a PME that is integrated with the TCP sender. Figures 8



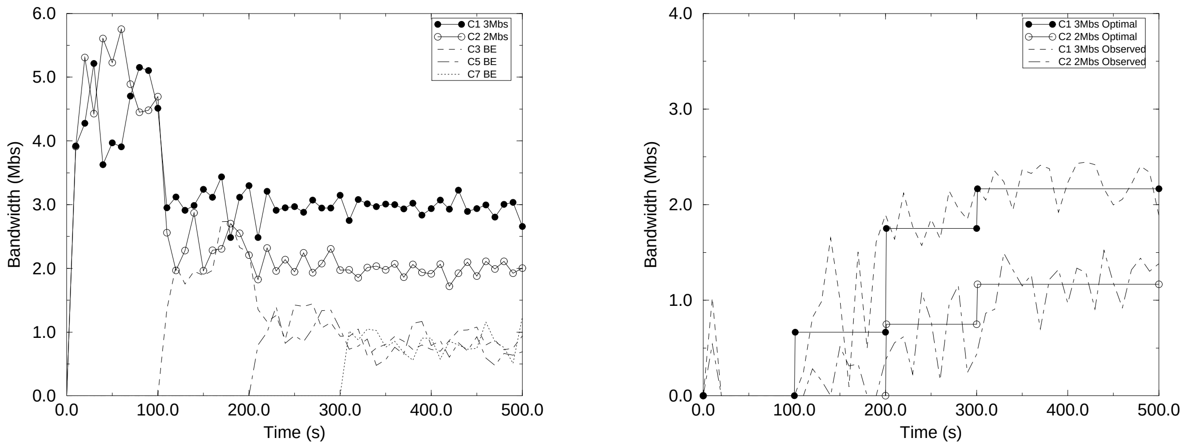
(a) Bandwidth graph

(b) Window trace of $3Mbs$ connection

Figure 10: Bandwidth sharing using source-integrated marking

and 9 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. 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 reduced. 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 threshold (*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 threshold, 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 7. Figure 10 shows results from the experiment presented in Figure 7 using the algorithm described above. In contrast to Figure 7(a), the amount of best-effort bandwidth received by the $3Mbs$ source closely matches the bandwidth received by the best-effort sources. Figure 10(b) shows the priority and best-effort windows of the $3Mbs$ connection. In contrast to Figure 7(b), the connection is able to compete for best-effort bandwidth independent of the priority marking.

To further examine the issue of fair bandwidth sharing, we took a closer look at the packet marking rate and its



(a) Total Bandwidths

(b) Marking Rates

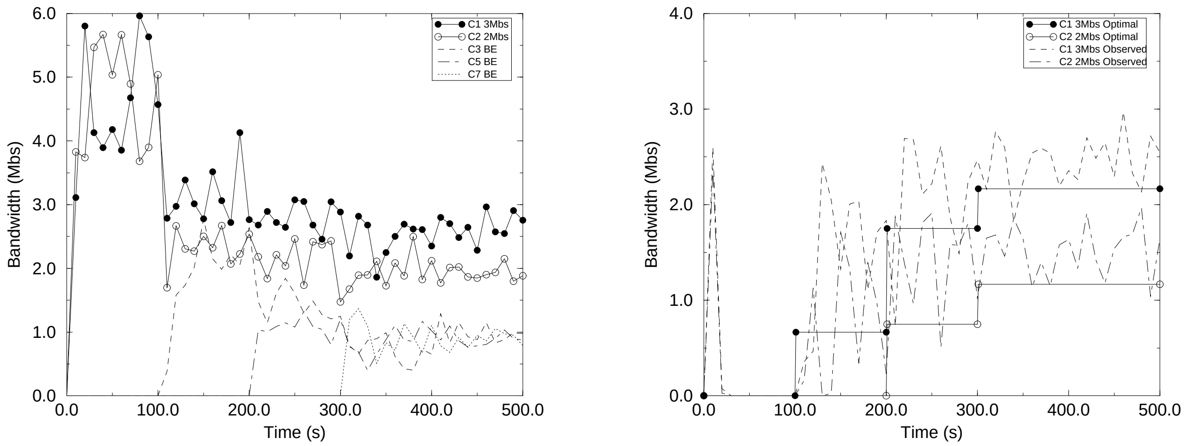
Figure 11: TCP with integrated packet marking

deviation from the theoretically computed optimal marking rate [1]. 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 the 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 under the conditions $r_i \geq 0$, $b \geq 0$, and $\sum_{i=1}^{n} R_i < B$ capture the system constraints:

$$r_i + b = R_i$$
$$\sum_{i=1}^{n} r_i + nb = B$$

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 last two best-effort sources are added. This time, *C1* needs to mark

[1]Marking rate is optimal when it is no more than what is required to sustain the desired throughput level. We note that when optimal marking is achieved, accurate congestion-based pricing [10] can be done using the marking rate of a connection.



(a) Total Bandwidths

(b) Marking Rates

Figure 12: TCP with transparent packet marking

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

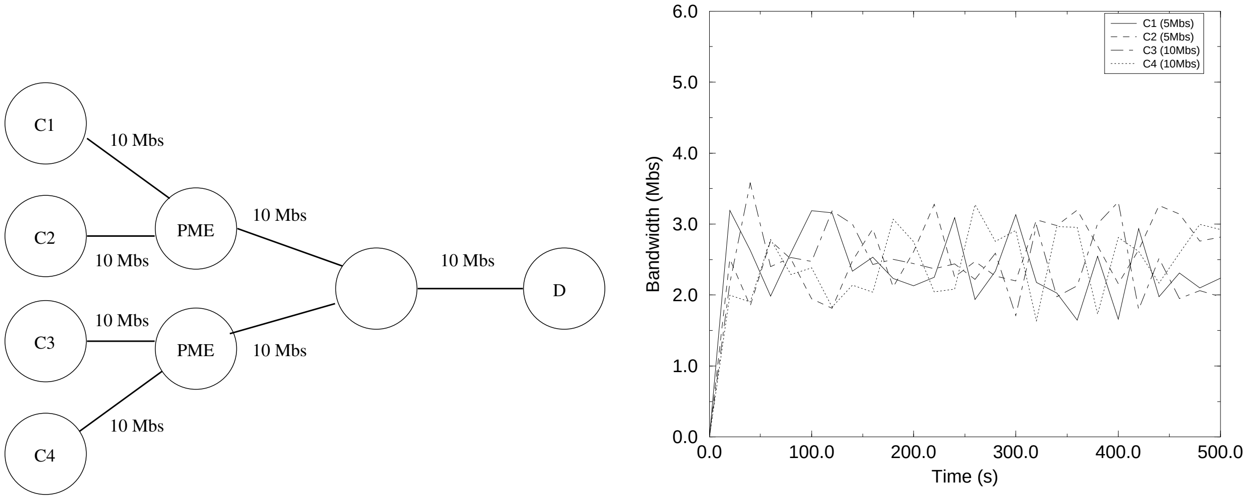
To examine the impact that the windowing modifications have, we performed the same set of experiments with a source-transparent PME. Figure 12 shows the total bandwidth and marking rate for different connections. Since TCP windowing algorithms restricts the connections *C1* and *C2* from competing for the excess bandwidth, the PME consistently overmarks its packets as shown in Figure 12(b). Increased marking can potentially fill the ERED queue with 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 rates.

5 Handling Oversubscription

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 architecture, when demand for service continually exceeds the capacity, admission control is used to deny additional connections access to the network in order to maintain the service levels of the current set of connections. In networks where no reservations or admission control is in place, the network must instead offer degraded service at times of overload. In both cases, pricing and access policies in conjunction with capacity planning must be used to balance the supply and the demand of network resources. This section describes how over-subscription is handled in our service model.

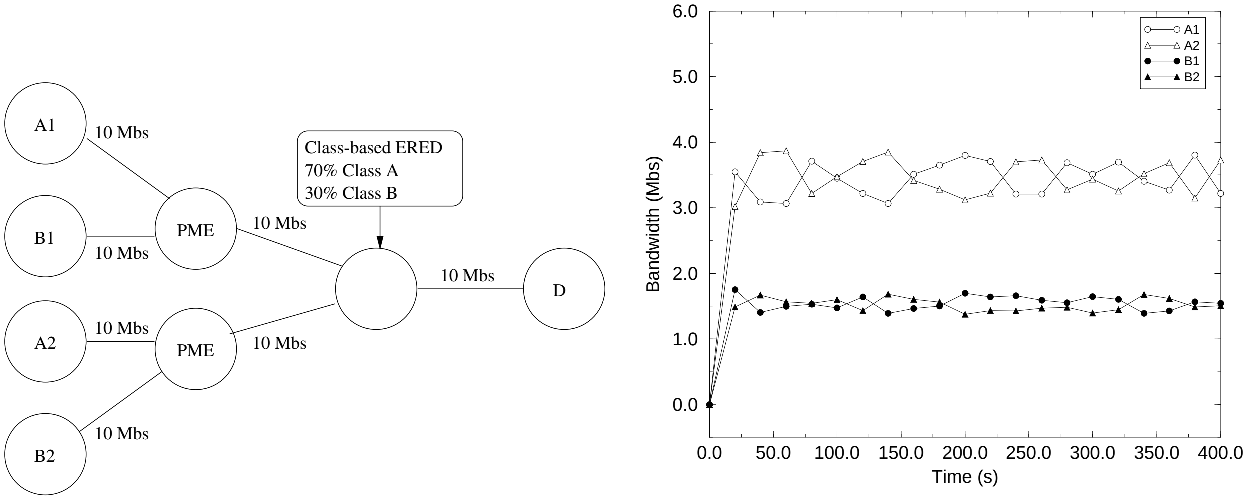
When aggregate demand exceeds capacity, all connections with non-zero target rates carry only marked packets. Consequently, they only compete for priority bandwidth and the ERED queue at the bottleneck degenerates into to RED queue serving only priority traffic. In the case of a source-transparent PME, since the underlying TCP windowing algorithm is not changed, the requested target bandwidth does not influence the throughput a source receives. Consequently, each source receives an equal fair share of the bottleneck bandwidth.

Over-subscription results in the same outcome when the PME is integrated within the source. In this case, since



(a) Network configuration (b) Bandwidth graph

Figure 13: Oversubscription



(a) Network configuration (b) Bandwidth graph

Figure 14: Oversubscription and multiple priorities

the algorithms for growing and shrinking the priority window are independent of the bandwidth demand, the windowing algorithm 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 the network. The connections *C1* and *C2* have a target rate of of $5Mbs$ each while connections *C3* and *C4* aim at a target rate of $10Mbs$. As the figure shows, by using the integrated marking scheme, each connection gets a fair share of the bottleneck bandwidth when the demand exceeds the capacity.

Another approach to handle over-subscribscription is to provide weighted bandwidth sharing depending on the target rates or the importance of the connections or connection groups. Since the proposed scheme uses only a single

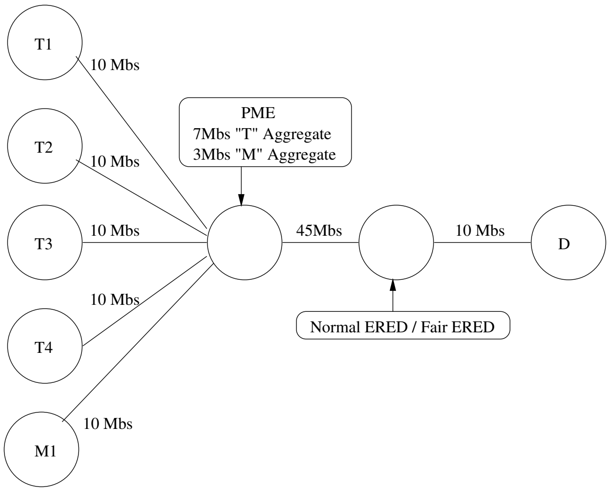


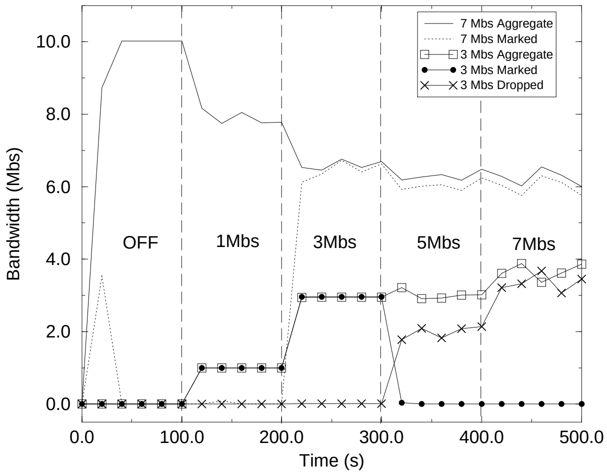
Figure 15: Network Topology

priority bit, it cannot itself be used to provide weighted bandwidth sharing in times of over-subscription. However, it is possible to implement weighted bandwidth sharing by using additional priority levels which give the network an indication of the connection's target rate and/or importance. For example, consider a more elaborate service architecture where additional priority bits are used to direct traffic into different ERED queues. These queues are then served using any one of various proposed queueing disciplines, such as weighted-fair queueing, class-based queueing, or even strict priority queueing. Figure 14(a) shows an example scenario in which an additional bit is used to select separate queues in a class-based queue [7]. In this example, the class-based queue is configured to provide applications and/or hosts in one class (A) with at least 70 percent of the allocated bandwidth. The remaining 30 percent of the bandwidth is allocated to the other class (B). When the applications in class A (A1 and A2) and class B (B1 and B2) request more bandwidth than is available, the additional priority encoding allows the network to maintain weighted bandwidth sharing between the two classes as shown in Figure 14(b).

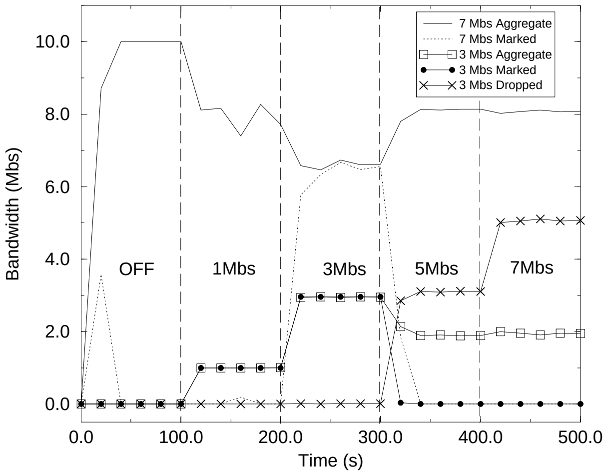
6 Dealing with Non-responsive Flows

One of the potential risks in an adaptive approach to service differentiation is that proliferation of applications which do not adapt to network dynamics can lead to severe performance degradadtion and even congestion collapse. Thus, an important issue in deploying the proposed scheme is the protection of the network against non-responsive flows [5, 9]. A salient feature of our scheme is that it provides performance incentives for applications to adapt to network dynamics and help avoid congestion collapse.

Figure 15 shows a network configuration which consists of four TCP connections (T1, T2, T3, and T4) which are competing for bandwidth with a non-responsive flow (M1) across a $10Mbs$ link. The aggregate target rate for the TCP connections is $7Mbs$. The target rate for the non-responsive flow is $3Mbs$. Initially, only the TCP sources are active and each competes fairly for the link bandwidth. The non-responsive flow starts transmitting at $1Mbs$ at $t = 100s$, and at $3Mbs$ at $t = 200s$. As shown in the figure, the aggregate throughput of the TCP connections drops when the non-responsive flow becomes active, but remains at a rate close to $7Mbs$. At $t = 300s$, the non-responsive flow increases its transmission rate to $5Mbs$, 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 reason why this happens is that the PME observes that the



(a) Non-responsive flows using a normal ERED queue



(b) Non-responsive flows using a Fair ERED queue

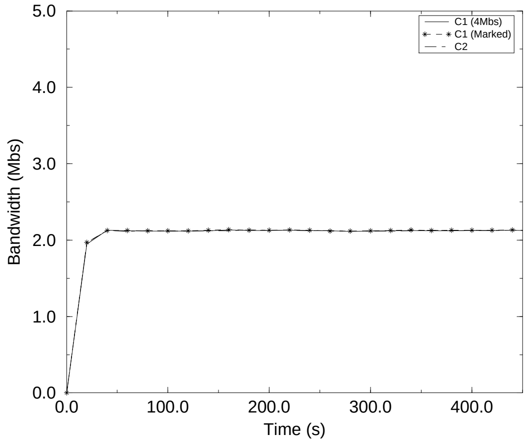
Figure 16: Non-responsive flows

non-responsive flow is sending packets at a rate which is higher than its given rate. In order to encourage sources to send packets which are deliverable, the PME counts every packet it receives for a particular flow against its allocation. The non-responsive flow further increases its transmission rate to $7Mbs$ at $t = 400s$. Again, the throughput observed by the 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 transmitting any amount above its allocated rate.

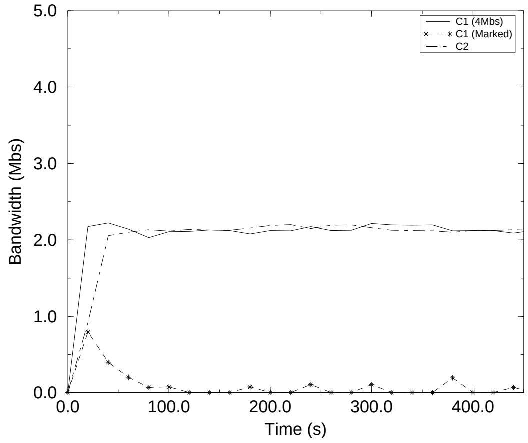
In the previous experiment, the non-responsive flow does, in fact, have a negative impact on the TCP connections. As Figure 16(a) shows, the aggregate marking rate of the TCP connections approaches the aggregate transmission rate, since the unmarked packets from the non-responsive flow dominates any of the excess bandwidth available. In effect, the non-responsive flow obtains all of the available best-effort bandwidth while shutting out all other well-behaved connections. In order to provide better fairness between connections competing for best-effort bandwidth, we enhanced the bottleneck ERED queue with additional fairness mechanisms based on FRED [9]. Figure 16(b) shows the results of the experiment. As the figure shows, when the non-responsive flow begins transmitting at a rate higher than $3Mbs$, the PME reduces its marking to zero as described earlier. Since the flow does not respond to congestion signals given by the bottleneck queue and continues to send an inordinate amount of unmarked packets, the fair ERED queue detects the flow and limits its throughput to a fair share of the best-effort bandwidth. In this case, a fair share of the bandwidth is $2Mbs$. Thus, by sending over its target rate of $3Mbs$ without regard to congestion in the network, the non-responsive flow reduces its own observed throughput to $2Mbs$. Note that given a fair share of the best-effort bandwidth, the TCP flows can now maintain their $7Mbs$ aggregate target rate without marking any packets. This is in contrast to Figure 16(a), where the TCP flows are forced to have all of their packets marked in order to maintain their target rate. Thus, the malicious flow hurts itself while helping other flows as it sends over its target rate without regard to network congestion.

7 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



(a) Packet marking gateway



(b) TCP with windowing modifications

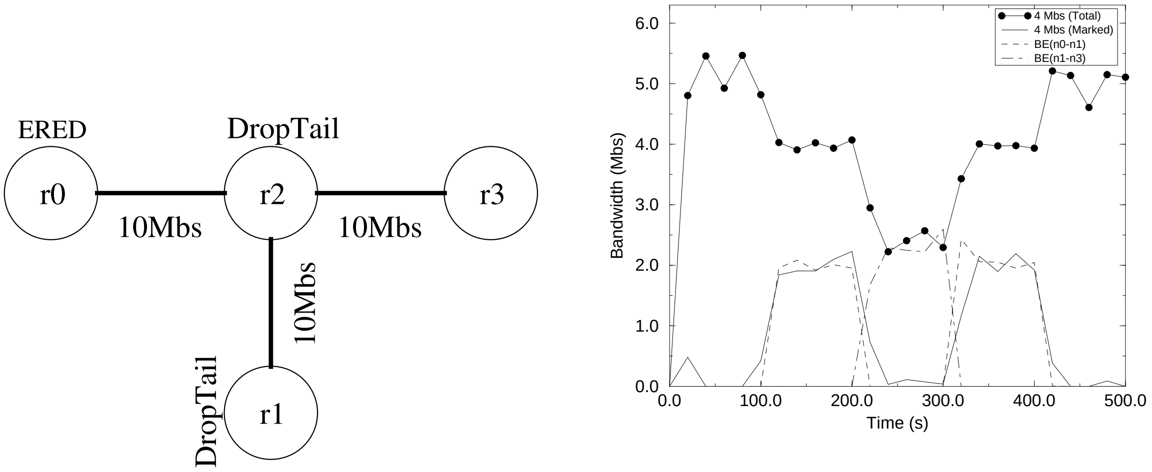
Figure 17: Performance over an all drop-tail network

fraction of them will support service differentiation between packets of different priorities. In order to be successful in this environment, it is important that any packet marking scheme proposed is capable of handling heterogeneity in the network. More specifically, it should be able to operate in an environment where all routers do not support service differentiation between priority and best-effort packets.

One of the salient features of the proposed scheme is its ability to operate in a network that does not provide service differentiation. When the PME is transparent to the source, TCP transmission control mechanisms are not affected as a result of packet marking. Thus, the lack of service differentiation simply makes the packet marking ineffective and the TCP sources behave as if they are operating in a best-effort network. When the PME is integrated with the source, the situation is little different. In this case, we essentially have two connections with differing priorities. Hence, 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 differentiation 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 backoff 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 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.

Figure 17(a) shows the throughput observed by five connections *C1*, *C2*, *C3*, *C4*, and *C5* going from *n0* to *n5*



(a) Network

(b) Bandwidth graph

Figure 18: Effects of heterogeneity

when all of the queues in the 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 source-transparent PME 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 the network does not honor their ToS bits. This is because the PME cannot determine that the ToS bits in the packets are being ignored unless it keeps additional information. Since the connection is always below its target bandwidth, the PME simply marks all of its packets. Figure 17(b) shows the same experimental setup as before. However, in this example, the PME 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 PME is integrated into the source adapt quickly to the changes in the network. This helps the source adapt its windowing and marking strategy as the bottleneck link shifts from non-priority to priority queues in a heterogeneous network. Figure 18(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 18(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 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.

8 Conclusions

In this paper, we have proposed and analyzed adaptive packet marking algorithms for providing soft bandwidth guarantees over a differentiated services 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, and 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. Finally, generalization of the two priority ToS scheme to multiple priorities is also under consideration.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. *IETF Request for Comments 2475*, December 1998.

[2] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification. *IETF Request for Comments 2205*, September 1997.

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

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

[5] K. Fall and S. Floyd. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, February 1998.

[6] W. Feng, D. Kandlur, D. Saha, and K. Shin. Understanding TCP Dynamics in an Integrated Services Internet. In *Proceedings of NOSSDAV*, pages 295–306, St. Louis, Missouri, May 1997.

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

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

[9] D. Lin and R. Morris. Dynamics of Random Early Detection. In *Proceedings of ACM SIGCOMM*, pages 127–137, Cannes, France, September 1997.

[10] J. MacKie-Mason and H. Varian. Pricing the Internet. In *Public Access to the Internet*, pages 269–314, May 1995.

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

[12] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. *IETF Request for Comments 2474*, December 1997.

[13] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. *IETF Request for Comments 2212*, September 1997.

[14] J. Wroclawski. Specification of Controlled-Load Network Element Service. *IETF Request for Comments 2211*, September 1997.

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