Computer Science Technical Reports

Department of Computer Science

2000

Traffic Conditioner Design for Assured Forwarding in Differentiated Services Network

Ahsan Habib

Sonja Fahmy
Purdue University, fahmy@cs.purdue.edu

Bharat Bhargava
Purdue University, bb@cs.purdue.edu

Report Number:
00-015

TRAFFIC CONDITIONER DESIGN FOR ASSURED FORWARDING IN DIFFERENTIATED SERVICES NETWORKS

Ahsan Habib
Sonia Fahmy
Bharat Bhargava

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD TR #00-015
November 2000

Traffic Conditioner Design for Assured Forwarding in Differentiated Services Networks*

Ahsan Habib, Sonia Fahmy, Bharat Bhargava

Center for Education and Research in Information Assurance and Security (CERIAS),

and Department of Computer Sciences

Purdue University

West Lafayette, IN 47907-1398, USA

E-mail: {habib, fahmy, bb}@cs.purdue.edu

Abstract

We design and evaluate an edge router for differentiated services networks. The edge router performs intelligent and adaptive traffic conditioning to improve application performance over the assured forwarding behavior. The conditioner is adaptive because the marking algorithm changes based upon the current number of flows traversing it. If there is a small number of flows, the conditioner stores per flow information and uses it to mark intelligently. On the other hand, if there are many flows going through the edge router, the conditioner does not store per flow information in order to scale. The conditioner improves fairness among TCP flows with variable round trip times. Simulation results indicate that our conditioner improves throughput of data extensive applications like large FTP transfers, and achieves low packet delays and response times for Telnet and WWW traffic.

Keywords: Quality of Service, Differentiated Services, Assured Forwarding, Traffic Conditioner.

*This research is supported in part by the National Science Foundation CCR-001712 and CCR-001788, CERIAS, an IBM SUR grant, and the Schlumberger Foundation technical merit award.

1 Introduction

The differentiated services (diff-serv) architecture [1] is a simple and scalable approach to improve Quality of Service (QoS) for data and multimedia applications in IP networks. The diff-serv model uses traffic conditioners at the edges of an administrative domain to shape, mark, and drop traffic if necessary. The operations are based on Service Level Agreements (SLAs) between adjacent domains, as well as the congestion status of the network. In the core of the network, Per Hop Behaviors (PHBs) are used to achieve service differentiation.

The current diff-serv model defines two forwarding mechanisms: Expedited Forwarding (EF) [14] and Assured Forwarding (AF) [10]. EF provides a virtual leased line service, while AF is more suited to applications like virtual private networks (VPNs). For the AF service, core routers use an active queue management technique such as Random Early Detection (RED) [9] with multiple thresholds, as in RIO (RED with IN and OUT) [3]. The Assured Forwarding PHB provides four classes (queues) of delivery for IP packets and three levels of drop precedence per class. These drop precedences determine the relative importance of a packet within an AF queue.

Designing an edge router that intelligently conditions AF traffic has been an active research area. Several studies show that application performance is poor if traffic conditioning at network edges does not consider transport protocol behavior at the end systems, and dropping behavior at the core routers. A number of studies propose to adjust the marking, dropping, or shaping scheme of a traffic conditioner based upon various factors. However, some of these proposals do not scale well. In addition, the studies only consider bulk data applications and do not examine delay-sensitive traffic and WWW traffic.

We study the behavior of transport protocols and use TCP characteristics to develop an intelligent traffic conditioner. Each conditioner feature is studied individually and then they are studied in combination. Our conditioner behaves differently based on the number of flows traversing it. This adaptive design overcomes scalability problems when a large number of flows is going through the edge router. We also improve the fairness of the round-trip time (RTT)-aware traffic conditioner proposed in [18] for a large number of flows. The performance of the conditioner is analyzed both for data intensive applications and delay sensitive applications with realistic traffic models.

The remainder of this paper is organized as follows. Section 2 presents the basics of traffic conditioning and differential drop. Section 3 discusses previous work on diff-serv assured forwarding. Section 4 discusses different design techniques and how to combine them in our proposed traffic conditioner. Section 5 contains all the details of our simulation setup. Section 6 presents and discusses the simulation results. We conclude with a summary and discussion of future work.

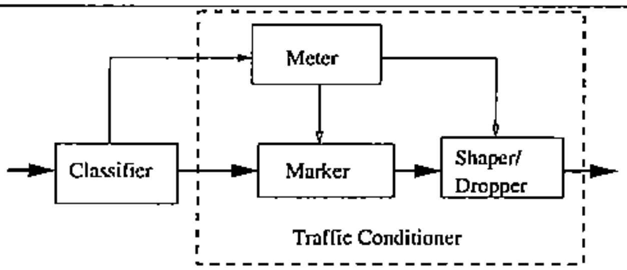


Figure 1: Components of a Traffic Conditioner

2 Background

This section describes the components of a traffic conditioner and how they relate to differential drop at core routers.

2.1 Basics of a Conditioner

A traffic conditioner may contain meters, markers, droppers, and shapers for traffic conditioning functions [1]. The conditioner may re-mark a traffic stream or may discard or shape packets to alter the temporal characteristics of the stream and bring it into compliance with a traffic profile specified by the network administrator. As shown in Figure 1, incoming traffic passes through a classifier, which is used to select a class for each traffic flow. The meter measures and sorts the classified packets into precedence levels. The decision (marking, shaping, or dropping) is made based on the measurement result.

Assured forwarding provides up to three drop precedences for each queue. We assume the drop precedences are DP0, DP1 and DP2, where DP0 means lower precedence to drop, and DP2 means higher. The Single Rate Three Color Marker (srTCM) [12] and Two Rate Three Color Marker (trTCM) [11] are the basic markers applicable to three drop precedences. srTCM meters an IP packet stream and marks its packets as either green, yellow, or red using a Committed Information Rate (CIR) and two associated burst sizes, a Committed Burst Size (CBS) and an Excess Burst Size (EBS). The trTCM uses two rates: a Committed Information Rate (CIR) and a Peak Information Rate (PIR). When traffic exceeds the CIR, packets are marked with drop precedence DP1. If traffic exceeds the PIR, packets are marked with DP2, the highest drop precedence. The Differentiated Services Code Point (DSCP) (contained in the IP header DSFIELD/ToS) is set to mark the DP. At the core of the network, the packets are treated based on the code point. When congestion occurs, packets marked with DP2 have the highest probability to be dropped, followed by DP1, and then DP0.

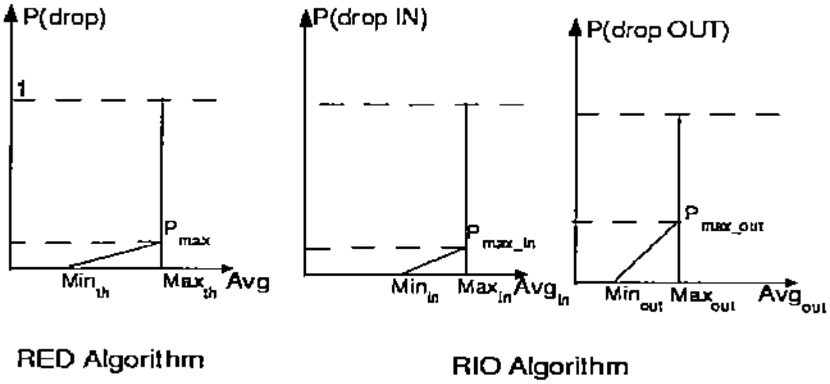


Figure 2: RED and RIO (not drawn to scale)

Shaping the traffic reduces the traffic variation and makes it smooth. It also provides an upper bound for the rate at which which the flow traffic is admitted into the network. It may be necessary to *drop* some packets at the boundary to ensure the flow complies with its profile. This dropping decision is taken by a policer.

2.2 Differential Drop

Edge routers mark packets with a code point that reflects the desired level of service, and core routers forward packets according to their markings. Within each core assured service queue, discrimination among packets is performed using a differential drop algorithm. Different queue mechanisms can be used to realize this preferential drop.

The RIO algorithm distinguishes between two types of packets, IN and OUT of profile, using two RED instances. Each RED instance is configured with a set of parameters: min_{th}, max_{th}, and P_{max}. Figure 2 shows the selection of these parameters for IN and OUT packets. Suppose the parameters for the IN profile packets are min_{in}, max_{in}, and P_{max_in}, and for the OUT of profile packets are min_{out}, max_{out}, and P_{max_out}. To drop OUT packets earlier than IN packets, min_{out} can be chosen smaller than min_{in}. The router drops OUT packets more aggressively by setting P_{max_out} higher than P_{max_in}. max_{out} may be chosen to be smaller than max_{in} so that OUT packets reach the congestion control phase (where probability of drop = 1) much earlier than IN packets. To realize three drop precedences, three REDs can be used. The average queue size is calculated using an exponentially weighted moving average algorithm with parameter w_q.

3 Related Work

Clark and Fang introduced RIO in 1998 [3], and developed the Time Sliding Window (TSW) tagger. The TSW tagger provides a smooth estimation of the TCP sending rate. The tagging algorithm tags packets as OUT once the traffic exceeds a certain threshold. Clark and Fang show that sources with different target rates can approximately achieve their targets using RIO even for different Round Trip Times (RTTs), whereas simple RED routers cannot.

Ibanez and Nichols [13] used a token bucket marker for Assured Service and showed that target rates and TCP/UDP interaction are key factors in determining throughput of flows. The TCP response to packet loss is the main cause for this. They concluded that it is unclear how the Assured Service can be characterized quantitatively for the TCP application. Seddigh, Nandy and Pieda [20] showed that the above mentioned factors are also critical for the distribution of excess bandwidth in an over-provisioned network. Lin, Zheng and Hou [15] proposed an enhanced TSW profiler and queue management algorithm to improve the assured service, but their solution requires state information to be maintained at core routers, which does not scale well.

Yeom and Reddy [22] pass the marking information to the sender, so that sender can slow down its sending rate in the case of congestion. This requires modifying the host TCP implementation. They also use three drop precedences IN, OUT-IN and OUT-OUT to provide better QoS. Storing and searching per flow information at the border router for a large number of flows may, however, not scale well.

Feroz et al [8] propose a TCP-Friendly marker. As TCP applications over Diff-Serv are influenced by bursty packet loss behavior, they use TCP characteristics to design their marker. The main concept is to "protect small-window flows from packet losses" by marking their traffic IN. The authors maintain spacing between IN and OUT tokens allocated for a flow to handle burstiness. Detailed analysis on a good window size threshold (below which a flow is marked as IN) for various situations is not provided [8]. A fixed window size threshold may not always be appropriate. We investigate different thresholds to identify a small window and analyze how they affect the goodput of flows with different RTTs.

Fang, Seddigh and Nandy [5] proposed the Time Sliding Window Three Color Marker (TSW3CM), which we use in this paper. Nandy et al [18] extend the TSW marker to design RTT-aware traffic conditioners. The basic idea of this conditioner is to adjust the packet drop rate in relation to RTT. Hence, the acquired bandwidth for the aggregate becomes less sensitive to RTT. Their conditioner is based on the steady state TCP behavior as reported by Matthis et al [16], i.e., bandwidth is inversely proportional to RTT. This TCP model does not consider timeouts. However, TCP connections time out when a large number of flows is multiplexed onto a bottleneck. We discuss improving the RTT-aware conditioner in

Section 4. Nandy et al also develop target-aware traffic conditioners to distribute excess bandwidth proportionally [18]. We use their idea to develop RTT aware three drop precedence (RTTAware3DP) conditioners.

Bonaventure and Cnodder [2] propose a rate adaptive shaper in combination with srTCM and trTCM to improve the performance of TCP by reducing the burstiness of the traffic. With TCP traffic, this reduction of burstiness is accompanied by a reduction of the number of marked packets, and thus by an improved TCP throughput.

Another marker is the adaptive packet marker by Feng et al [7]. They use a Packet Marking Engine (PME), which can be a passive observer under normal conditions, but becomes an active marker at the time of congestion. The marking rate is adjusted by the throughput. This engine can be source transparent or integrated with the source. The host TCP reacts to the marked/unmarked packet drop differently by maintaining two congestion windows: one for best effort traffic and another for priority traffic. The source-integrated approach is hard to deploy.

4 Proposed Traffic Conditioner

In this section, we discuss techniques to incorporate in a conditioner to improve its performance. Some of these techniques are (loosely or closely) based on ideas proposed in the literature, as cited below, but the implementation strategy and combination of some techniques, and the adaptivity of the conditioner to the number of flows have not been previously examined. We simulate these techniques for various application types in Section 6.

We use the TSW tagger [3], a rate estimator, and the TSW3CM marker [5]. We refer to this combination as a standard conditioner. In addition, we examine:

1. SYN : The first few packets of a TCP flow should not be dropped to allow the TCP congestion window to grow. At the edge router, the first few packets can be identified by their sequence numbers. As TCP sequence number can be arbitrarily random, it needs to store any per flow information (initial sequence number)at the edge. Feroz and et al [8] use a similar technique to protect small windows. To avoid storing per flow information at the edge we give low priority to drop only to SYN packets, which TCP uses to start a connection. In this case we don't need to store per flow information at the edge. We use this property to design our edge router.

2. Small Window (SW): As in [8], we protect small window flows from packet losses by marking them with DP0. TCP grows the congestion window exponentially until it reaches the slow start threshold, *ssthresh*. The congestion window reduces to 1 or half of the *ssthresh* for timeouts or packet loss respectively. We give low drop priority to flows with small congestion window sizes. The window size of a TCP connection is calculated using the sequence

number of packets in the forward direction and the sequence number of acknowledgments (ACKs) in the opposite direction. This technique requires per flow state at the edge router. Although edge routers may generally maintain per flow state, increasing this state can be problematic for the borders between large domains with many flows. We use SW when storing per flow information is possible and use SYN in other cases.

3. Congestion Window Reduction (CWR): In the case of Explicit Congestion Notification (ECN) routers and ECN TCP flows, routers set the congestion experienced bit, which is echoed by the receiver. The sender reduces the congestion window in response, and sets the CWR (Congestion Window Reduced) bit in the next data packets. Giving low drop priority for the packet after a TCP window reduction can avoid consecutive *ssthresh* reductions that lead to poor performance with TCP Reno [4]. This condition is not the same as SW or SYN. The connection may still have a window larger than the SW threshold.

4. RTT-awareness: An RTT-aware traffic conditioner is proposed in [18]. The marker avoids RTT bias of TCP connections through marking packets with high drop priority inversely proportional to the square of their RTTs. This is based upon the steady state TCP behavior in [16]. Equation (1) shows that bandwidth is inversely proportional to RTT where MSS is the maximum segment size and p is the packet loss probability:

   $$BW \propto \frac{MSS}{RTT\sqrt{p}} \tag{1}$$

   The conditioner works well when the number of flows is small because equation (1) accurately represents fast retransmit and recovery. We have observed that for a large number of flows, *small RTT* flows time out because only high RTT flows are protected by the conditioner after satisfying the target rate. Excess bandwidth is mostly given to large RTT flows. To remedy this situation, we can use one of two strategies. First, we can use the throughput approximation by Padhye et al [19], which considers the timeouts. Equation (2) shows this approximation, where b is the number of packets that acknowledged by a received ACK, and T_0 is the timeout length:

   $$BW \approx \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)} \tag{2}$$

   Designing an RTT-aware traffic conditioner using equation (2) may remedy the timeout problem for a large number of flows.

   The second way to avoid the problem is to combine the RTT-aware technique with SW, because SW helps to grow congestion windows and recover from timeouts. We use this method in our simulations (Section 6) and show its

```
If measuredRate <= targetRate
  mark packets as DP0
Else
  If the flow has windowSize < k
  mark packets as DP0
Else
  mark packets as DP0 with probability (1-p)
If packet is not marked DP0
  mark packets as DP1 with probability (1-q)
  mark packets as DP2 with probability q
where p and q are:
p = $\frac{(measuredRate - targetRate)}{measuredRate}$
q = $\left(\frac{minRTT}{aggregateRTT}\right)^2$
```

Figure 3: An RTT Aware Traffic Conditioner with three drop precedence and Small Window Protection

effectiveness. The RTTAware3DP with SW is shown in Figure 3. We also use a target aware conditioner as in [18] (described in the next item).

One problem with RTT-aware conditioners is that edge routers exchange minimum RTT information and each computes the aggregate RTT of flows going through it.

5. Target Rate (TR): Target rate is the most important factor in marking. Nandy et al [18] mark DP1 and DP2 only when target rates have been achieved, and marking is inversely proportional to the square of the flow requested rates if proportional sharing of excess bandwidth is required. Another strategy is to mark packets based on the difference between target rate and exponentially averaged input rate of the aggregate in order to improve fairness. We use the first strategy.

6. Burst: The marker avoids marking high drop priority in bursts to work well with TCP Reno. The shaper avoids burstiness to avoid consecutive packet drops and poor performance.

Each of the above techniques has advantages and limitations. SYN, CWR, and aggregate Target Rate do not need to store per flow information and are simple to implement. On the other hand, SW, Target Rate based on individual information, and Burst need to store significant per flow information. RTT-aware conditioners do not need to store per flow information but edge routers need to exchange aggregate minimum RTT information.

We devise two types of edge routers. One is simple and uses the techniques that do not require per flow information. The other type can store per flow information and is used between domains with a small number of flows traversing the boundary, e.g., a user and an ISP (provider). The edge router can alternate between the two strategies. When the number of flows is large, our edge router uses techniques that do not need to store per flow information, and when the number of flows decreases, the router switches to the mode that gives better performance at the expense of per flow state. The conditioner algorithm is:

```
If numberOfFlows < threshold
  Use Standard with CWR, SW, Burst
Else
  Use Standard with SYN and CWR
```

The threshold to switch between the simple and more complex modes depends on the resources available at the edge router. This is a router configuration parameter. We study the performance of this adaptive conditioner in section 6.

5 Simulation Setup

We use the **ns-2** simulator [17] for our experiments. For the standard Diff-Serv implementation, we use software developed at Nortel Networks [21]. This implementation includes the TSW tagger meter, token bucket meter, srTCM and trTCM meters as well as TSW2CM, TSW3CM, token bucket, srTCM, and trTCM markers. We use the TSW tagger meter and TSW3CM marker.

We use the same network topology as in [18]. Each edge router handles flows with different RTTs to simulate different users. The topology is shown in Figure 4. There are three edge devices $E1$, $E2$, and $E3$ and a core device C. The edge devices implement traffic conditioning, while the core device implements the AF PHB using three drop precedences. Three different buffer occupancies are calculated and tracked. These are $q0$, $q1$ and $q2$ for the corresponding drop precedences. The probability of dropping DP0 packets depends on $q0$; DP1 packets depends on $q0 + q1$; and DP2 depends on the total. The RED parameters for $\{min_{th}, max_{th}, P_{max}\}$ used are: for DP0 {40,55,0.02}; for DP1 {25,40,0.05}; and for DP2 {10,25,0.1} [18]. w_q is 0.002 for all REDs. TCP New Reno is used with a packet size of 1024 bytes and a maximum

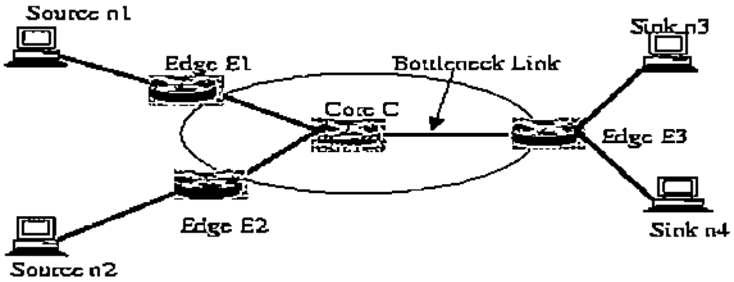


Figure 4: Network Topology used in the simulation. All links are 10 Mbps.

window of 64 packets.

We use two aggregate flows. Flow 1-3 is from node $n1$ to $n3$ and Flow 2-4 is from node $n2$ to $n4$. The number of flows in each aggregate is varied to show small and large number of flows depending on the experiment. We use 10 micro-flows per aggregate as a small number of flows and 200 micro-flows as a large number of flows. Normally RTT for Flow 1-3 is fixed at 20 ms and RTT of Flow 2-4 varies from 1-200 ms.

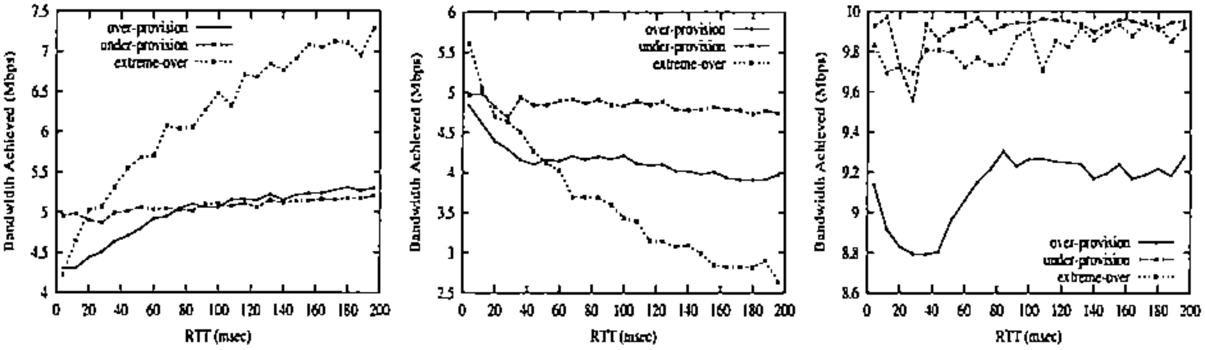
The metrics used to evaluate performance include:

1. **Throughput:** This denotes the total packets received by the receiver application over simulation time. A higher throughput usually means better service for the application (e.g., smaller completion time for an FTP flow). For the ISP, higher throughput is preferable because this means links are well-utilized. This metric is useful for data traffic.

2. **Packet Drop Ratio:** This is the ratio of total packets dropped at the core to the total packets sent. A user can specify that the packet drop ratio should not exceed a threshold. This is a metric for both ISP and user. Lower drop rate reduces bandwidth and resource waste on upstream links.

3. **Packet Delay:** For delay sensitive applications like Telnet, the packet delay is a user metric. We use this metric to show that the user of a delay sensitive application still benefits from intelligent traffic conditioner design.

4. **Response Time:** This is the time between sending a request to a web server and receiving the response back from the server. We show both the time it takes to get the first packet, and that it takes to get all requested data from the server. This metric is only used for WWW traffic.



(a) Aggregate Flow 1-3

(b) Aggregate Flow 2-4

(c) Total of Both Flows

Figure 5: Throughput for standard traffic conditioner in over, under and extremely over-provisioned network for 200 flows. Both flows have the same target rate.

6 Simulation Results

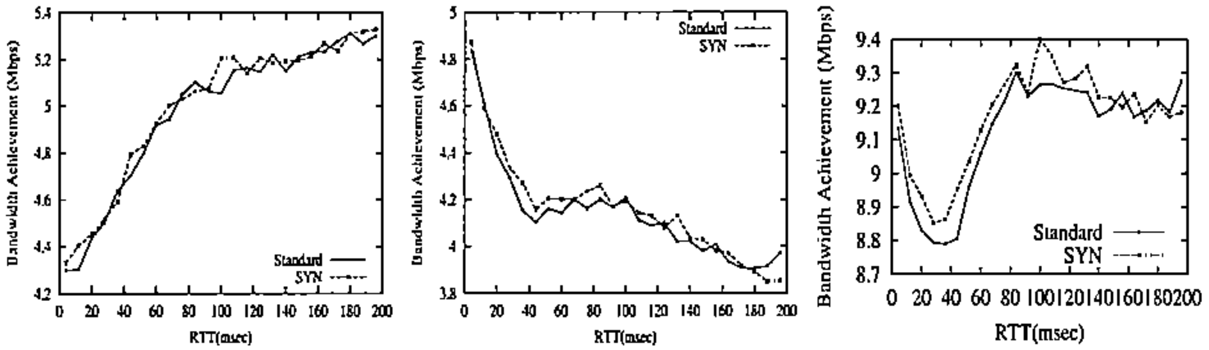
We first study the behavior of the standard traffic conditioner with FTP (loss-sensitive) traffic. Then each design technique discussed in section 4 is analyzed individually and in combination. We also study the performance of the proposed adaptive traffic conditioner, and examine performance with Telnet and WWW (delay and response time-sensitive) applications. Finally, we present our results on the RTT-aware traffic conditioner.

6.1 Design Techniques

The objective of this experiment is to study how each design technique can affect the performance of the standard traffic conditioner individually and collectively. The RTT of aggregate Flow 1-3 is fixed at 20 ms and the RTT of aggregate Flow 2-4 is varied from 1 to 200 ms. The RTTs, window size for SYN and SW, and target rate to provision the network are input parameters for this experiment. The output parameters are Throughput and Packet Drop Ratio.

Standard Conditioner: We test the conditioner for both small (10 micro flows) and large (200 micro flows) number of flows, in under and over provisioned networks. All flows have same target rate. For the over provisioned case, the committed rate, CIR, is 2 Mbps and peak rate, PIR, is 3 Mbps for each aggregate flow. For extremely over provisioned, CIR is 0.2 Mbps and PIR is 0.3 Mbps, and for under provisioned CIR is 6 Mbps and PIR is 10 Mbps.

Figure 5 shows achieved bandwidth in an under, over and extremely over-provisioned network. The figure shows the bandwidth achievement of each aggregate flow as well as the total bandwidth achievement by both flows for varying RTTs



(a) Aggregate Flow 1-3

(b) Aggregate Flow 2-4

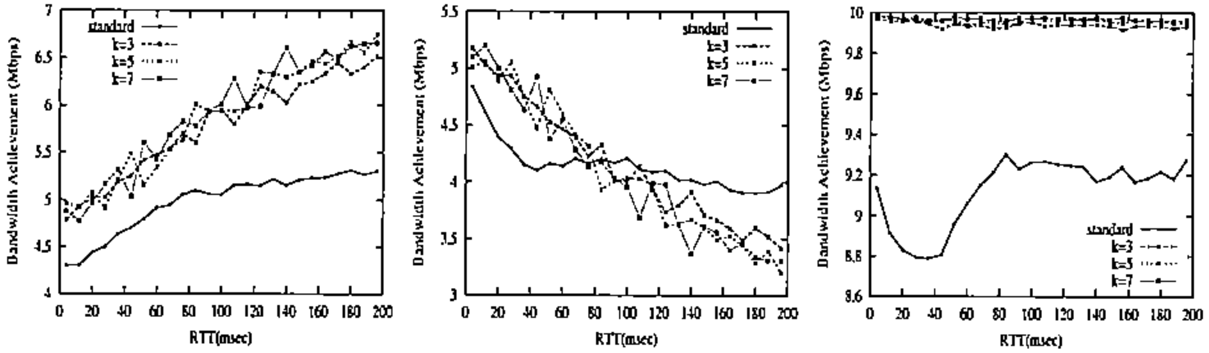
(c) Total of Both Flows

Figure 6: Throughput of SYN in an over-provisioned network for 200 flows.

of flow 2-4. The under-provisioned case exhibits high link utilization. Both flows achieve close to 5 Mbps for that case, which is a desirable outcome. In the over-provisioned cases, small RTT connections are favored. When RTT is small for Flow 2-4 (< 20ms), it gets more bandwidth than Flow 1-3, while Flow 1-3 is favored at the expense of Flow 2-4 when its RTT is longer. If the network is extremely over-provisioned, we see more unfairness and higher packet drop ratio. This is because TCP connections are very aggressive for the flow with small RTT. Due to the fluctuation of the sending rate, TCP loses more packets. As the RTT of Flow 1-3 is fixed, it has almost the same packet drop ratio throughout the the experiment but the drop ratio decreases when RTT of Flow 2-4 increases. This is because for higher RTT, TCP can estimate the sending rate more accurately.

SYN : SYN is useful for short-lived connections and high degrees of multiplexing. Figure 6 shows the achieved bandwidth for SYN. Even though the bandwidth improvement is not significant (200 Kbps for the total), SYN can be used when other expensive techniques (in terms of complexity to deploy) can not be used. We use this technique in our adaptive conditioner when number of flows is high.

Small Window: Small window (SW) works both for small and large number of micro flows as well as short and long lived flows. To study the effect of the window size, *k*, on achieved bandwidth on both flows, *k* is varied from 3 to 10. If the window size of a flow is less than *k*, the flow packets are marked DP0. Figure 7 shows the achieved bandwidth for different *k* values for an over-provisioned network. A larger value of *k* helps the (more aggressive) small RTT connection (Flow 1-3) achieve more bandwidth at the expense of the large RTT flow (Flow 2-4) due to the preferential drop at the core. This contrast is even more clear in an under-provisioned network. Both flows achieve more bandwidth than the standard



(a) Aggregate Flow 1-3

(b) Aggregate Flow 2-4

(c) Total of Both Flows

Figure 7: Throughput comparison for different window sizes of standard conditioner with Small Window for 200 flows in an over-provisioned network.

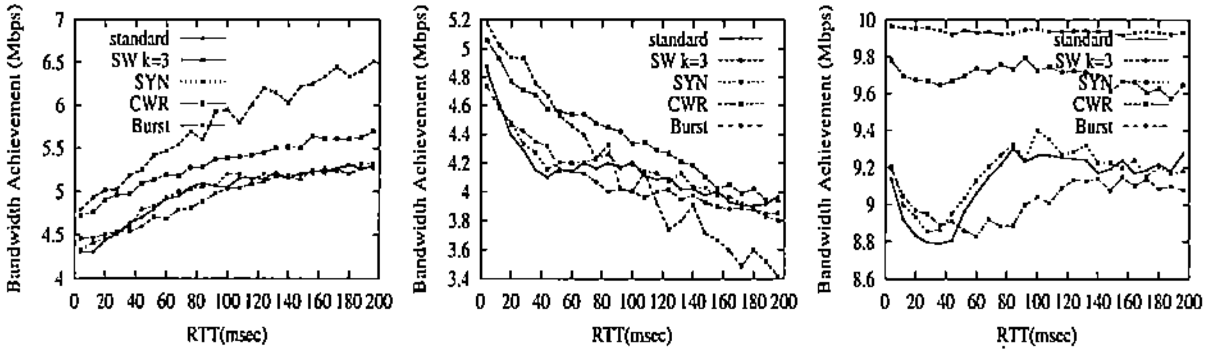
conditioner as long as the RTT of Flow 2-4 is less than 80 ms. After that, Flow 1-3 continues gaining bandwidth and Flow 2-4 losing bandwidth. Using SW, total bandwidth in an over-provisioned network is close to the link capacity (figure 7(c)). Thus SW significantly improves utilization. The choice of *k* depends on policy. A higher value of *k* such as 7 or 8 may favor short RTT flows and result in more unfairness against long RTT flows, while a lower value of *k* (e.g., 3) avoids this problem.

Congestion Window Reduction (CWR): Giving priority to CWR packets helps grow the congestion window and reach equilibrium. Results show that CWR helps Flow 2-4 achieve high throughput. Flow 1-3 times out and has high packet drop.

Burst: Avoiding bursty marking and shaping packet bursts improves achieved bandwidth over the standard traffic conditioner. The improvement is more significant for both flows when RTT is low. Flow 2-4 achieves its highest bandwidth in an over-provisioned network when Burst and CWR are combined for low RTT. The "Burst" technique exhibits the lowest packet drop ratio for both flows.

Target Rate: We use a Target Aware traffic conditioner to divide excess bandwidth in an over-provisioned network in proportion to the subscribed target rates [18]. This feature has no effect in cases of congestion.

Combinations and Overall Performance: Figure 8 compares different design techniques. Small window contributes most to total bandwidth gain, followed by CWR and SYN (figure 8(c)). SW favors short RTT connections (Flow 1-3), but it reduces packet drop ratio and timeouts for Flow 2-4 as well, compared to the standard traffic conditioner. Burst is effective



(a) Aggregate Flow 1-3

(b) Aggregate Flow 2-4

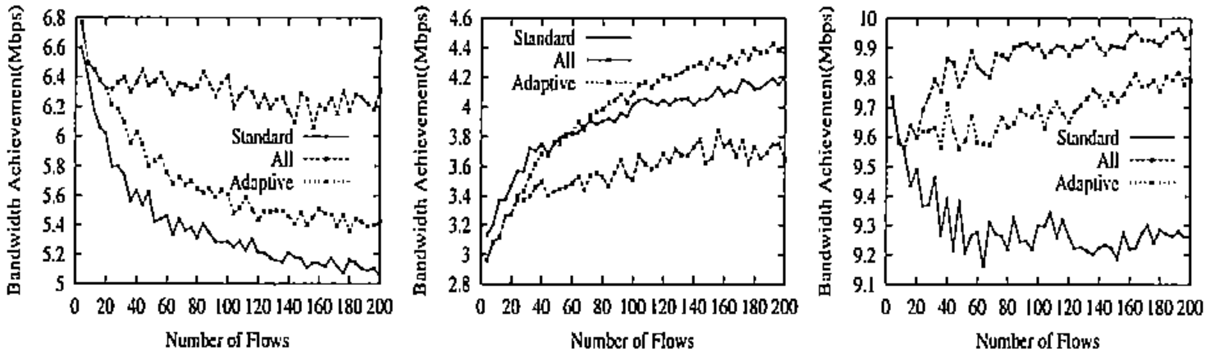
(c) Total of Both Flows

Figure 8: Throughput of standard traffic conditioner for all design techniques with 200 flows.

for short RTT (less than 40 ms). If SW is not used, Burst+CWR achieves higher bandwidth than any other combination. Using all design techniques together has advantages over small window alone. We summarize as follows:

- All design techniques have some advantages over the standard traffic conditioner.
- Small Window (SW) works better than any other technique alone. However, it favors flows with shorter RTTs. Using a small value of window size k improves overall throughput without excessively punishing long RTT flows.
- CWR favors long RTT flows.
- Giving priority to the first packet (SYN) slightly improves throughput. SYN does not require per flow information at the edge router.
- CWR, SYN and SW with $k = 3$ can be grouped together to improve high RTT connection performance. Alternatively, SW with $k = 7$ and Burst can be grouped together to favor small RTT connections. SYN is not needed when SW is used, since SYN is subsumed by SW.
- Some design techniques are only effective under certain conditions. For example, SYN is only effective for short lived flows.

Adaptive Conditioner: Our adaptive conditioner selects the design techniques used based on number of active flows present. The conditioner may work without per flow information, which makes it more scalable. Figure 9 compares achieved bandwidth with standard, adaptive, and standard with all design techniques (referred to as "All"). The figure



(a) Aggregate Flow 1-3

(b) Aggregate Flow 2-4

(c) Total of Both Flows

Figure 9: Comparison of achieved bandwidth among standard, adaptive and standard with all techniques, for an over-provisioned network.

shows that when the number of flows is less than 20, all three have similar performance. When the number of flows exceeds 40, "all" outperforms the others. In this case, Flow 1-3 gains more than 1 Mbps bandwidth and Flow 2-4 loses around 0.5 Mbps. Total bandwidth achieved for "all" is close to the bottleneck link capacity (10 Mbps). The adaptive conditioner is more fair in the sense that Flow 1-3 does not steal bandwidth from flow 2-4, and total achieved bandwidth is close 10 Mbps. When the number of flows increases, the gap between achieved bandwidth of both flows decreases in the standard and adaptive conditioner. The adaptive conditioner achieves 0.5 Mbps total bandwidth more than the standard conditioner.

6.2 Telnet and WWW Traffic

We compare the performance of telnet and WWW applications (which are more delay/response time sensitive) with the standard conditioner and with all (SYN, SW, CWR, Burst) design techniques together.

For the Telnet experiment, the setup is the same as before, but the metric used is the average and maximum packet delay time for each Telnet micro flow in an aggregate flow. We simulate 200 users from nodes $n1$ and $n2$ connected to nodes $n3$ and $n4$ over the bottleneck link. The topology is same as figure 4 but link capacities are changed to introduce congestion. All links have a capacity of 1 Mbps except link $C1 - E3$, the bottleneck link, which is 200 kbps.

Figure 10 shows the average packet delay of Flow 1-3 with standard versus standard with all techniques conditioners. Most of the flows with the standard conditioner have higher packet delay than with all techniques. The result is same for

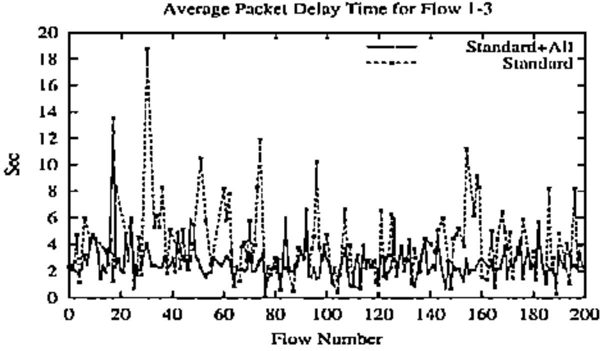


Figure 10: Comparison of average packet delay for standard and standard+all techniques for Telnet traffic.

Flow 2-4. We have repeated this experiment for 100 micro flows and the outcome is the same: average Telnet packet delay is reduced with the designed conditioner.

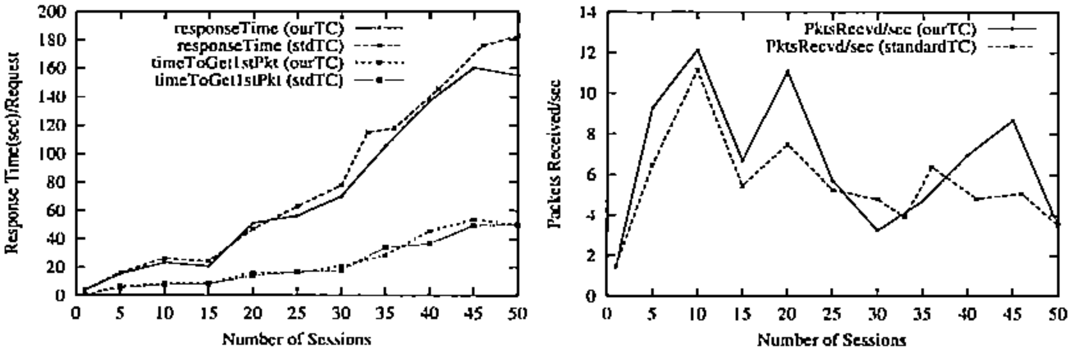
As web traffic constitutes most (60%-80%) of the Internet traffic, we test our traffic conditioner with the WWW traffic model in **ns-2** [17]. (Details of the model are given in [6].) The model uses HTTP 1.0 with TCP Reno. Servers are attached to $n3$ and $n4$ of Figure 4, while $n1$ and $n2$ are used as clients. A client can send a request to any server. Each client generates a request for 5 pages with a variable number of objects (e.g., images) per page. We use the default **ns-2** probability distribution parameters to generate inter-session time, inter-page time, objects per page, inter-object time, and object size (in kB).

Figure 11 shows the average response time per WWW request received by the client in a heavily congested network. The network setup is same as with Telnet traffic except the bottleneck link $C1-E3$ has a capacity of 100 kbps only. Figure 11(a) shows that our conditioner (with all techniques) reduces response time over the standard traffic conditioner. Figure 11(b) shows that our conditioner results in increased transfer rate to clients over the standard conditioner. If the network is over-provisioned, the performance gain is insignificant.

6.3 RTT-Aware Traffic Conditioner

The RTT-aware conditioner (both 2 and 3 DPs) [18] is unfair when a large number of flows is being multiplexed. With a large number of flows, Flow 2-4 (higher RTT flow) gets almost all of the extra bandwidth after satisfying the target rate of both flows. Figure 12 shows that Flow 1-3 achieves only 2.3 Mbps whereas Flow 2-4 gets 7.52 Mbps at RTT=100 ms.

We examine the reason for this behavior and how to overcome the problem. As Flow 2-4 has a large RTT, it gets higher



(a) Average response time in seconds

(b) Data transferred per second

Figure 11: a. Response time, and b. Data transfer rate among clients and servers per second for WWW traffic in an extremely congested network.

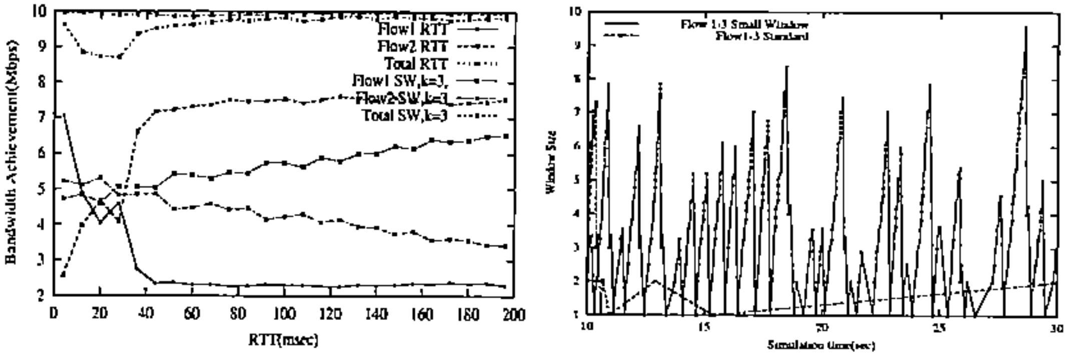
priority over Flow 1-3 after satisfying the target rate. As a result, many micro flows of Flow 1-3 time out and Flow 1-3 cannot achieve more than the target rate. Figure 12 shows that a micro flow of Flow 1-3 times out, and its congestion window (*cwnd*) remains small.

The RTT-aware conditioner avoids this problem by using SW. This is because with a larger number of flows, the per micro flow bandwidth share is small and thus, the "steady-state" *cwnd* is reduced. When *cwnd* is small, there is a larger probability of timeout in the case of packet drops. Protecting packets (via DP0 marking) when the window is small reduces time-outs, especially back-to-back time-outs. The micro flow recovers from timeouts when SW is used as shown in the algorithm in Figure 3.

The behavior discussed above does not occur when the network is under-provisioned because none of the flows gets priority over others. In an under-provisioned network, SW increases the throughput of Flow 1-3 at the expense of Flow 2-4, as discussed in the previous subsection. Fluctuations occur when RTT is relatively low for both connections. The fluctuations can be overcome using the Burst technique. CWR helps Flow 2-4 achieve more bandwidth as before. The RTT-aware conditioner works in the same manner as [18] for a small number of flows, and the effect of all design techniques is similar to the standard conditioner discussed before.

7 Conclusions and Future Work

In this paper, we have analyzed traffic conditioners for small and large numbers of flows, as well as for over and under-provisioned networks. We have proposed a traffic conditioner that uses several design techniques to improve application



(a) Throughput in Mbps

(b) Congestion window of a micro-flow

Figure 12: a. Throughput of an RTT-aware traffic conditioner and b. Comparison of congestion window size for standard and standard with small window for a micro flow of Flow 1-3.

performance. All techniques discussed improve performance but small window (SW) protection contributes the most. However, it favors mostly small RTT flows in our experiments. A lower threshold for the window size reduces this unfairness, without compromising the total bandwidth gain. Congestion Window Reduced (CWR) packet protection favors long RTT flows, while burst avoidance (Burst) is effective when round trip time (RTT) is small.

The techniques can be grouped as follows. CWR, SYN , and SW with small threshold (e.g., 3) are grouped to improve high RTT connection performance. Alternatively, SW with threshold 7 and Burst are grouped to favor short RTT connections. Making the conditioner RTT-aware makes it unfair when there is a large number of flows. With SW, however, the unfairness no longer exists.

The techniques can also be grouped based on whether they need to store per flow information. An edge router can implement the simple (no per flow state) or complex (per flow state) methods only, or it can incorporate an adaptive design that dynamically switches among the two. Our proposed conditioner has been shown to improve FTP throughput, reduce packet delay for Telnet and response time for WWW traffic.

This work can be extended in a number of ways. The effect of the interaction between TCP and UDP on the conditioner will be studied. For the adaptive conditioner, we plan to study how to set the threshold to switch between two modes based on available resources. We will also incorporate a feedback mechanism between exit and entry routers to avoid resource waste on upstream links due to packet drop downstream.

Acknowledgements

The authors would like to thank Nabil Seddigh, Peter Pieda and Biswajit Nandy from Nortel Networks for their valuable comments.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for Differentiated Services. RFC 2475, December 1998.

[2] O. Bonaventure and S. de Cnodder. A rate adaptive shaper for Differentiated Services. Internet Draft draft-bonaventure-diffserv-rashaper-00.txt, work in progress, 1999.

[3] D.D. Clark and W. Fang. Explicit allocation of best effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6, 4:362–374, 1998.

[4] S. Fahmy. *New TCP standards and flavors*, chapter 13 of *High Performance TCP/IP Networking*. Prentice Hall, Inc., 2001.

[5] W. Fang, N. Seddigh, and B. Nandy. A Time Sliding Window Three Colour Marker. RFC 2859, June 2000.

[6] A. Feldmann, A. C. Gilbert, P. Huang, and W. Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. *ACM SIGCOMM '99*, pages 301–313, 1999.

[7] W.C. Feng, D. Kandlur, D. Saha, and K.G. Shin. Understanding and improving TCP performance over networks with minimum rate guarantees. *IEEE Transactions on Networking*, 7, 2:173–186, April 1999.

[8] A. Feroz, S. Kalyanaraman, and A. Rao. A TCP-Friendly traffic marker for IP Differentiated Services. *Proc. of the IEEE/IFIP Eighth International Workshop on Quality of Service - IWQoS*, 2000.

[9] S. Floyd and V. Jacobson. Random Early Detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1, 4:397–413, 1993.

[10] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB group. RFC 2597, June, 1999.

[11] J. Heinanen, T. Finland, and R. Guerin. A two rate Three Color Marker. RFC2698, September 1999.

[12] J. Heinanen and R. Guerin. A single rate Three Color Marker. RFC 2697, September 1999.

[13] J. Ibanez and K. Nichols. Preliminary simulation evaluation of an Assured Service. Internet Draft, draft-ibanez-diffserv-assured-eval-00.txt, August 1998.

[14] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding PHB. RFC 2598, June 1999.

[15] W. Lin, R. Zheng, and J. Hou. How to make Assured Services more assured. *In Proceedings of ICNP*, Oct 1999.

[16] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion aviodance algorithm. *ACM SIGCOMM Computer Communication Review*, 27, No. 3:67–82, 1997.

[17] S. McCane and S. Floyd. Network simulator ns-2. http://www.isi.edu/nsnam/ns/, 1997.

[18] B. Nandy, N. Seddigh, P. Pieda, and J. Ethridge. Intelligent Traffic Conditioners for Assured Forwarding based Differentiated Services networks. *IFIP High Performance Networking (HPN 2000), Paris*, June 2000.

[19] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. *IEEE SIGCOMM '98*, 1998.

[20] N. Seddigh, B. Nandy, and P. Pieda. Bandwidth assurance issues for TCP flows in a Differentiated Services network. *submitted to Globecom 99*, 1999.

[21] F. Shallwani, J. Ethridge, P. Pieda, and M. Baines. Diff-Serv implementation for ns. http://www7.nortel.com:8080/CTL/#software, 2000.

[22] I. Yeom and N. Reddy. Realizing throughput guarantees in a Differentiated Services network. *IEEE Int. Conf. on Multimedia Comp. and Systems*, June 1999.