On Monitoring and Controlling QoS Network Domains*

Ahsan Habib, Sonia Fahmy, Bharat Bhargava
CERIAS and the Department of Computer Sciences
Purdue University, West Lafayette, IN 47907–1398

{habib, fahmy, bb}@cs.purdue.edu

ABSTRACT

We design an integrated distributed monitoring, TCP-friendly traffic conditioning, and flow control system for securing network domains. Edge routers monitor (using tomography techniques) a network domain to detect quality of service (QoS) violations– possibly caused by underprovisioning– as well as bandwidth theft and denial of service (DoS) attacks. To bound the monitoring overhead, the router only verifies service level agreement (SLA) parameters such as delay, loss, and throughtput when anomalies are detected. The marking component of the router uses TCP flow characteristics to protect "fragile" flows. The edge routers may also regulate unresponsive flows. Ingress routers propagate congestion information to upstream domains. Preliminary simulation results indicate that this design increases application-level throughput of data applications such as large FTP transfers; achieves low packet delays and response times for Telnet and WWW traffic; and detects traffic-intensive attacks and service violations.

Categories and Subject Descriptors

C.2.5 [**Local and wide-area networks**]: Internet; D.4.8 [**Performance**]: Simulation

General Terms

Algorithms, Design, Performance

1. INTRODUCTION

In the last few years, the areas of network monitoring and network tomography– mapping the Internet by composing several end-to-end measurements– have witnessed a flurry of research activity. These new results, however, have not been integrated with the more mature research on traffic control. Our goal in this paper is to demonstrate that traffic conditioning at network domain edges, together with low-overhead monitoring and unresponsive flow control, mitigate congestion, unfairness, and misbehaving user problems in Internet domains. Monitoring of network activity can aid in detecting denial of service and bandwidth theft attacks, which have become an expensive problem in today's Internet. We will integrate intelligent traffic marking with unresponsive flow control and tomography-based network monitoring, with the objectives of securing network domains from attacks and malicious users, and achieving higher user-perceivable quality of service.

In the remainder of this section, we give some background on the differentiated services architecture– which we use as an underlying quality of service (QoS) framework– and summarize our design. Section 2 discusses previous results related to the components of our proposed edge routers. Our networking monitoring and loss inference techniques for attack detection are discussed in section 3. Section 4 discusses the design of adaptive TCP-aware traffic conditioners. Section 5 explains how to detect and control unresponsive flows during congestion. Our simulation setup for performance evaluation is described in section 6. Section 7 discusses our main results. We conclude in section 8.

1.1 Differentiated Services

The differentiated services (diff-serv) architecture [4] is a simple approach to enhance quality of service (QoS) for data and multimedia applications in the Internet. In diff-serv, complexity is pushed to the boundary routers of a network domain to keep core routers simple. The edge routers at the boundary of an administrative domain shape, mark, and drop traffic if necessary. The operations are based on Service Level Agreements (SLAs) between adjacent domains. The traffic enters a diff-serv domain at an *ingress* router and leaves a domain at an *egress* router. An ingress router is responsible for ensuring that the traffic entering the domain conforms to the SLA with the upstream domain. An egress router may perform traffic conditioning functions on traffic forwarded to a peering domain. In the core of the network, Per Hop Behaviors (PHBs) achieve service differentiation. The current diff-serv specification defines two PHB types: Expedited Forwarding [26] and Assured Forwarding (AF) [24]. AF provides four classes (queues) of delivery with three levels of drop precedence (DP0, DP1, and DP2) per class. The Differentiated Services Code Point (DSCP), contained in the IP header DSFIELD/ToS field, is set to mark the drop precedence. When congestion occurs, packets marked with higher precedence (e.g., DP2) must be dropped first. The AF PHBs at core routers use an active queue management algorithm such as Random Early Detection (RED) [20] for IN and OUT of profile (RIO) packets [10]. The RIO algorithm distinguishes between two types of packets, IN and OUT of profile, using two RED instances. To realize three drop precedences, three RED instances can be used.

1.2 Edge Routers

Edge routers perform critical traffic conditioning and control functions. The edge router may alter the temporal characteristics of a stream to bring it into compliance with a traffic profile specified by the network administrator [4]. A traffic meter measures and sorts packets into precedence levels. Marking, shaping, or dropping decisions are based upon the measurement result.

Marking: Markers can mark packets deterministically or probabilistically. A probabilistic packet marker, such as Time Sliding Window marker [14], obtains the current flow rate, *measuredRate*, of a user from the meter. The marker tags each packet based on the

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

targetRate from the service level agreement and the current flow rate. An incoming packet is marked as *IN* profile (low probability to drop) if the corresponding flow has not reached the target rate, otherwise the packet is marked as high drop precedence with probability $1 - p$, where p is given by equation (1):

$$p = \frac{measuredRate - targetRate}{measuredRate} \qquad (1)$$

Shaping/Dropping: Shaping reduces traffic variation and provides an upper bound for the rate at which the flow traffic is admitted into the network. A shaper usually has a finite-size buffer. Packets may be discarded if there is not sufficient space to hold the delayed packets. Droppers drop some or all of the packets in a traffic stream in order to bring the stream into compliance with the traffic profile. This process is know as *policing* the stream.

1.3 Our Basic Design

Our proposed edge router (1) marks TCP traffic with knowledge of TCP congestion control functions, (2) controls unresponsive flows and transfers congestion information upstream, and (3) monitors the network for possible attacks and SLA violations. The three components aim at increasing application-level performance and network resource utilization. Monitoring also aids in detecting and controlling denial of service (DoS) attacks and under-provisioning problems. The edge router components, and the flow of data and control among them, are depicted in figure 1. We describe each component in the next few paragraphs.

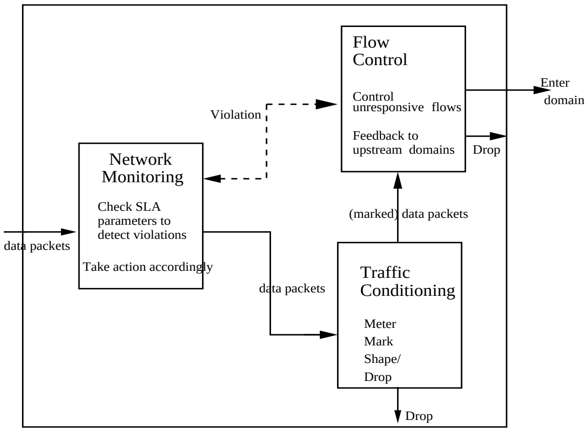


Figure 1: Components of an edge router

SLA Monitoring. QoS-enabled networks can face different attacks from traditional IP network domains. For example, users may inject or re-mark traffic with high QoS requirements which may cause other users to have low throughput, high delay, and packet loss. Our SLA monitoring component flags service violations and bandwidth theft attacks. To monitor the network without core router involvement, we use network tomography techniques such as per-segment loss inference mechanisms [12] in an edge-to-edge manner.

Traffic Conditioning. The routers utilize knowledge of TCP characteristics to give priority to “critical” packets, and mitigate TCP bias to flows with short round trip times (RTTs). While edge routers between a stub domain and a transit domain are not generally overloaded, many edge routers, such as Internet Exchange points among peering domains, are highly loaded. Therefore, the edge routers use packet header information instead of stored state when possible, and use replacement policies to control the amount of state maintained.

Congestion Control. Unresponsive flows do not reduce their transmission rates in response to congestion. Congestion collapse can be mitigated using improved packet scheduling or active queue management [5, 32], but such open loop techniques do not affect congestion caused by unresponsive flows in upstream domains. We need a mechanism to control the rate at which packets enter the domain to the rate at which packets leave the domain. Congestion is detected when many high priority packets are dropped [38]. Ingress routers which detect or infer such drop can regulate unresponsive flows.

We conduct a series of simulation experiments to study the behavior of this framework. Preliminary results show that TCP-aware edge router marking improves throughput of data extensive applications like large FTP transfers, and achieves low packet delays and response times for Telnet and WWW traffic. We also demonstrate how attacks and unresponsive flows alter network delay and loss characteristics, and hence can be detected and controlled.

2. RELATED WORK

Providing QoS in diff-serv networks has been extensively studied in the literature. Clark and Fang introduced RIO in 1998 [10], and developed the Time Sliding Window (TSW) tagger. They show that sources with different target rates can achieve their targets using RIO even for different Round Trip Times (RTTs), whereas simple RED routers cannot. Assured Forwarding is studied by Ibanez and Nichols in [25]. They use a token bucket marker and show that target rates and TCP/UDP interaction are key factors in determining throughput of flows. Seddigh, Nandy and Pieda [35] also show that the distribution of excess bandwidth in an over-provisioned network is sensitive to UDP/TCP interactions. Lin, Zheng and Hou [27] propose an enhanced TSW profiler, but their solution requires state information to be maintained at core routers. We now discuss results related to the three components of our edge router.

2.1 Network Tomography and Violation Detection

Since bottleneck bandwidth inference techniques such as packet pairs were proposed in the early 1990s, there has been increased interest in inference of internal network characteristics (e.g., per-segment delay, loss, bandwidth, and jitter) using correlations among end-to-end measurements. This problem is called *network tomography*. Recently, Duffield et al [12] have used unicast packet “stripes” (back-to-back probe packets) to infer link-level loss by computing packet loss correlation for a stripe at different destinations. This work is an extension of loss inference with multicast traffic, e.g., [1, 7]. We develop a tomography-based, low overhead method to infer delay, loss, and throughput and detect problems that alter the internal characteristics of a network domain.

Network monitoring techniques have also been recently studied. In efficient reactive monitoring [11], global polling is combined with local event driven reporting to monitor IP networks. Breitbart et al [6] use probing-based techniques where path latencies and bandwidth are measured by transmitting probes from a single point of control. They find the optimal number of probes using vertex cover solutions. Recent work on SLA validation [8] uses a histogram aggregation algorithm to detect violations. The algorithm measures network characteristics like loss ratio and delay on a hop-by-hop basis and uses them to compute end-to-end measurements. These are then used in validating the end-to-end SLA requirements. We use an Exponential Weighted Moving Average (EWMA) for delay, and average of several samples for loss as in RON [3], since it is more flexible and accurate.

2.2 Traffic Conditioning

Fang et al [14] proposed the Time Sliding Window Three Color Marker (TSW3CM), which we use as a standard traffic conditioner. Adaptive packet marking [16] uses a Packet Marking Engine (PME), which can be a passive observer under normal conditions, but becomes an active marker at the time of congestion. Yeom and Reddy [39] also convey marking information to the sender, so that it can slow down its sending rate in the case of congestion. This requires modifying the host TCP implementation. Feroz et al [18] propose a TCP-Friendly marker. The marker protects small-window flows from packet loss by marking their traffic as IN profile. We develop similar techniques with reduced overhead [21, 23]. Nandy et al design RTT-aware traffic conditioners [31] which adjust packet marking based on RTTs, to mitigate TCP RTT bias. Their conditioner is based on the steady state TCP behavior as reported by Matthis et al in [29]. Their model, however, does not consider time-outs which we consider in this paper.

2.3 Congestion Control

Floyd et al discuss congestion collapse from undelivered packets in [19]. Congestion collapse occurs when upstream bandwidth is consumed by packets that are eventually dropped downstream. Seddigh et al [36] propose separating TCP (responsive to congestion) and UDP (may be unresponsive) to control congestion collapse caused by UDP. Albuquerque et al [2] propose a mechanism, Network Border Patrol, where border routers monitor all flows, measure ingress and egress rates, and exchange per-flow information with all edge routers periodically. The scheme is elegant, but its overhead is high. Chow et al [9] propose a similar framework, where edge routers periodically obtain information from core routers, and adjust conditioner parameters accordingly. We propose to only send load information during congestion, since core networks may be lightly loaded most of the time. In the Direct Congestion Control Scheme (DCCS) [38], only drops of packets with the *lowest drop priority* are tracked. We follow the same methodology to detect congestion and control unresponsive flows. Aggregate-based Congestion Control (ACC) detects and controls high bandwidth aggregate flows [28]. We use similar IP prefix matching of destination addresses to detect attacks targeting the same destination.

3. TOMOGRAPHY-BASED VIOLATION DETECTION COMPONENT

QoS network domains should detect service violations (excessive delay or loss that customers experience) and bandwidth theft attacks. An attacker can impersonate a legitimate customer by spoofing its identity. Network filtering [17] can detect spoofing if the attacker and the impersonated customer are in different domains, but the attacks may proceed unnoticed otherwise. QoS domains support low priority classes, such as best effort, which are not controlled by edge routers. The service provider should ensure that high priority customers are getting their agreed-upon service, so that the network can be re-configured or re-provisioned if needed, and attackers which bypass or fool edge controls are prevented. In case of distributed DoS attacks, flows from various ingress points are aggregated as they approach their victim. Monitoring can control such high bandwidth aggregates at the edges, and propagate attack information to upstream domains [22]. As with any detection mechanism, the attackers can attack the mechanism itself, but the cost to attack our distributed monitoring mechanism is higher than the cost to inject or spoof traffic, or bypass a single edge router.

We measure SLA parameters such as delay, packet loss, and throughput to ensure that users are obtaining their agreed upon service. Delay is defined as the edge-to-edge latency; packet loss is the ratio of total flow packets dropped in the domain[1] to the total packets of the same flow which entered the domain; and throughput is the total bandwidth consumed by a flow inside a domain. If a network domain is properly provisioned and no user is misbehaving, the flows traversing the domain should not experience excessive delay or loss. Although jitter (delay variation) is another important SLA parameter, it is flow-specific and therefore, not suitable to use in network monitoring. In this section, we describe edge-to-edge measurement and inference of delay, loss and throughput, and a violation detection mechanism.

3.1 Delay Measurements

Delay bound guarantees made by a provider network to customer flows are for the delays experienced by the flows while traversing between the ingress and egress edges of the provider domain. For each packet traversing an ingress router, the ingress copies the packet IP header into a new packet with a certain pre-configured probability p_{probe}. The ingress encodes the current time into the payload and marks the protocol identifier field of the IP header with a new value. The egress router recognizes such packets and removes them from the network. Additionally, the egress router computes the packet delay for flow i by subtracting the ingress time form the egress time. (We assume NTP is used to synchronize the clocks.) The egress then sends the packet details and the measured delay to an entity we call the *SLA monitor*. At the monitor, the packets are classified as belonging to customer j and the average packet delay of the customer traffic is updated using an exponential weighted moving average (EWMA) (we use a current sample weight 0.2). If this average packet delay exceeds the delay guarantee in the SLA, we conclude that this may be an indication of a SLA violation.

3.2 Loss Inference

Packet loss guarantees made by a provider network to a customer are for the packet losses experienced by its conforming traffic inside the provider domain. Measuring loss by observing packet drop at all core routers and communicating them to the SLA monitor at the edge imposes significant overhead. We use packet stripes [12] to infer link-level loss characteristics inside the domain. A series of probe packets with no delay between the transmission of successive packets, or what is known as a "stripe," is periodically transmitted. For a two-leaf tree spanned by nodes 0, k, R_1, R_2, stripes are sent from the root 0 to the leaves to estimate the characteristics of three links (figure 2). For example, the first two packets of a 3-packet stripe are sent to R_2 and the last one to R_1. If a packet reaches a receiver, we can deduce that the packet has reached the branch point k. By monitoring the packet arrivals at R_1, R_2 and both, we can write equations with three known quantities and estimate the three unknown quantities (loss rates of links $0 - k$, $k - R_1$ and $k - R_1$) by applying conditional probability definitions, as discussed in [12]. We combine estimates of several stripes to limit the effect of non-perfect correlation among the packets in a stripe. This inference technique extends to trees with more than 2 leaves and more than 2 levels [12].

We extend this end-to-end unicast probing scheme to routers with multiple active queue management instances, e.g., 3-color RED [20], and develop heuristics for the probing frequency and the par-

[1] a flow can be a micro flow defined by (source and destination addresses and ports, and protocol identifier) or an aggregate of several micro flows.

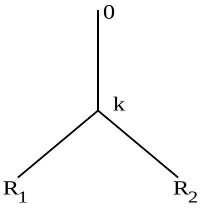


Figure 2: Binary tree to infer per-segment loss from source 0 to receivers R_1 and R_2

ticular receivers to probe to ensure good domain coverage. Assured forwarding queues use three drop precedences referred to as green, yellow, and red. Suppose $\mathcal{P}_{red}$ is the percentage of *red* packets accepted (not dropped) by the active queue. We define percentages for yellow and green traffic similarly, and show how these percentages are computed in the appendix. Link loss can be inferred by subtracting the transmission probability from 1. If L_g, L_y, and L_r are the inferred losses of green, yellow and red traffic respectively, loss can be expressed as:

$$L_{class} = \frac{n_g \mathcal{P}_{green} L_g + n_y \mathcal{P}_{yellow} L_y + n_r \mathcal{P}_{red} L_r}{n_g + n_y + n_r} \qquad (2)$$

where n_i is number of samples taken from i-colored packets. However, when loss of green traffic is zero, we take the average of yellow and red losses. When the loss of yellow traffic is also zero, we report only loss of red probes. We reduce the overhead of loss inference by probing the domain links with high delay only, as determined by the delay measurement procedure. We also measure throughput by probing egress routers *only* when delay and loss are excessive. This helps pinpoint the user or aggregate which is consuming excessive bandwidth, and causing other flows to receive lower quality than their SLAs.

3.3 Violation and Attack Detection

When delay, loss, and bandwidth consumption exceed the predefined thresholds, the monitor concludes there may be an SLA violation or attack. Excessive delay is an indication of abnormal conditions inside the network domain. If there are losses for the premium traffic class, or if the loss ratios of assured forwarding traffic classes exceed certain levels, a possible SLA violation is flagged. The violation can be caused by aggressive or unresponsive flows, denial of service attacks, flash crowds, or network underprovisioning. To detect distributed DoS attacks, the set of links with high loss are identified. If high bandwidth aggregates traversing these high loss links have the same destination IP prefix, there is either a DoS attack or a flash crowd, as discussed in [28]. If this is determined to be an attack, the appropriate ingress routers are notified and the offending user traffic is throttled, as discussed in section 5.

4. TRAFFIC MARKING COMPONENT

We incorporate several techniques in the conditioner to improve performance of applications running on top of TCP. The first few packets of a TCP flow should not be dropped to allow the TCP congestion window to grow. At the edge router, we give low drop priority to SYN packets as indicated in the TCP header. Since TCP grows the congestion window exponentially until it reaches the slow start threshold, *ssthresh*, and the congestion window is reduced to 1 or half of the *ssthresh* for time-outs or packet loss, we may also protect small window flows from packet losses by marking them with DP0, as proposed in [18]. Edge routers use sequence number information in packet headers in both directions to determine this. ECN-Capable TCP may reduce its congestion window due to a time-out, triple duplicate ACKs, or in response to explicit congestion notification (ECN) [34]. In this case, TCP sets the CWR flag in the TCP header of the first data packet sent after the window reduction. Therefore, we give low drop priority for a packet if the CWR or ECN bit is set. This avoids consecutive *ssthresh* reductions that lead to poor performance with TCP Reno [13]. We also mark packets inversely proportionally to the square of the flow requested rates if proportional sharing of excess bandwidth is required [31]. The marker avoids marking high drop priority in bursts to work well with TCP Reno, as proposed in [18].

We also use an RTT-aware traffic conditioner to avoid the TCP short RTT bias, if RTT and RTO information is available. Equation (3) shows that, in a simple TCP model that considers only duplicate ACKs [29], bandwidth is inversely proportional to RTT where MSS is the maximum segment size and p is the packet loss probability:

$$BW \propto \frac{MSS}{RTT\sqrt{p}} \qquad (3)$$

An RTT-aware marking algorithm based on this model (proposed in [31]) works well for a small number of flows because equation (3) accurately represents the fast retransmit and recovery behavior when p is small. We have observed that for a large number of flows, short RTT flows time out because only long RTT flows are protected by the conditioner after satisfying the target rates. To mitigate this unfairness, we use the throughput approximation by Padhye et al [33]:

$$BW \propto \frac{MSS}{RTT\sqrt{\frac{2bp}{3}} + To \times min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)} \qquad (4)$$

where b is the number of packets acknowledged by a received ACK, and To is the time-out length. Designing an RTT-aware traffic conditioner using equation (4) is more accurate than using equation (3) because it considers time-outs. Simplifying this equation, we compute the packet drop ratio between two flows, ρ as:

$$\rho^2 = \left(\frac{RTT_1}{RTT_2}\right)^2 \times \frac{To_1}{To_2} \qquad (5)$$

where RTT_i and To_i are the RTT and time-out of flow i respectively [21]. The marker uses both equation (5) and equation (1).

Each of the techniques discussed in this section has advantages and limitations. Protecting SYN, ECN, and CWR packets, and marking according to the target rate do not need to store per flow information and are simple to implement. On the other hand, protecting small window flows and marking according to the RTT and RTO values requires maintaining and processing per flow information. To bound state overhead at the edge routers, we store per flow information at the edge only for a certain number of flows based on available memory. The edge router uses a least recently used (LRU) state replacement policy when the number of flows exceeds the maximum number that can be maintained. Therefore, for every flow, conditioning is based on state information if it is present. If there is no state present, conditioning only uses techniques that rely on header information. The conditioner pseudo-code is given in figure 3.

5. UNRESPONSIVE FLOW CONTROL COMPONENT

This section describes the detection and control of unresponsive flows. SLA monitors (or core routers) inform edge routers of con-

```
for For each incoming flow do
  if there is a complete state entry for this flow then
    statePresent = TRUE
    Update the state table
  else
    statePresent = FALSE
    Add the flow in the state table (replace if needed)
  end if
  if statePresent is TRUE then
    Use Standard TC with SYN, ECN, CWR, small window,
    burst, RTT-RTO
  else
    Use Standard TC with SYN, ECN, and CWR
  end if
end for
```

Figure 3: Algorithm for Adaptive Traffic Conditioner

gestion inside a domain. A shaping algorithm controls unresponsive flows at the time of congestion. In addition, ingress routers of a domain may propagate congestion information to the egress router of the upstream domain.

5.1 Optional Core Router Detection

In section 3, we have shown how tomography-based loss inference techniques can be applied to detect per-segment losses using edge-to-edge probes. An alternative strategy is to track excessive drop of high priority (i.e., green or DP0) packets at core routers, as proposed in [38]. We adapt this technique to detect congestion *only* for unresponsive flows using protocol information from the transport layer. The core router tracks the tuple {source address, destination address, source port, destination port, protocol identifier, timestamp, $btlnkbw$} for dropped $DP0$ packets. The outgoing link bandwidth at the core, $btlnkbw$, helps regulate the flow: edge routers shape more aggressively if the core has a thin outgoing link. The core sends this drop information to the ingress routers *only* when the total drop exceeds a local threshold (thus the flow seems non-adaptive).

5.2 Metering and Shaping

At the egress routers, we distinguish two types of drops: drop due to metering and shaping at downstream routers *sdrop*, and drop due to congestion at core/edge routers, *cdrop*. Egress/core routers communicate this drop information to ingress routers and the upstream egress router. For a particular flow, assume the bottleneck bandwidth is $btlnkbw$ (as given above); the bandwidth of the outgoing link of the flow at the ingress router is $linkbw$; the flow has an original profile (target rate) of $targetrate$; and the current weighted average rate for this flow is $wavg$. In case of $cdrop$, the profile of the flow is updated temporarily (to yield rate $newprofile$) using equations (6) and (7) where $0 < \gamma < 1$ is the congestion control aggressiveness parameter:

$$decrement = cdrop \times packet\_size \times max(1, \gamma \frac{linkbw}{btlnkbw}) \quad (6)$$

$$newprofile = max(0, min(newprofile - decrement, wavg - decrement)) \quad (7)$$

A higher value of γ speeds up convergence, but application QoS may deteriorate. A lower value makes traffic smoother, but it takes longer to readjust the rate. The “max” term in the equation can be ignored if the bottleneck bandwidth information cannot be obtained (tools like pathchar or Nettimer cannot be used), or core router detection (section 5.1) is unavailable. In equation (7), the weighted average of the arrival rate is computed using the Time Sliding Window [10] algorithm.

For *sdrop*, the profile is adjusted as follows:

$$newprofile = max(0, newprofile - sdrop \times packet\_size) \quad (8)$$

The $newprofile$ is initialized to $targetrate$. In the absence of drops, the router increases the adjusted profile periodically at a certain rate $increment$. The rate $increment$ is initialized to a constant number of packets each time the router receives drop information, and is doubled when there is no drop, until it reaches a threshold $\frac{wavg}{f}$, and then it is increased linearly. Thus, the rate adjustment algorithm follows TCP congestion control. At the edge, shaping is based on the current average rate and the adjusted profile. For each incoming flow, if the current average rate is greater than the adjusted profile, some misbehaving flow packets are dropped.

6. SIMULATION SETUP

We use simulations to study the effectiveness of our edge router design. The **ns-2** simulator [30] with the differentiated services implementation of Nortel Networks [37] is used. We use the following RED parameters $\{min_{th}, max_{th}, P_{max}\}$: for DP0 {40, 55, 0.02}; for DP1 {25, 40, 0.05}; and for DP2 {10, 25, 0.1} (as suggested by [31]). w_q is 0.002 for all REDs. TCP New Reno is used with a packet size of 1024 bytes and a maximum window of 64 packets. We vary the number of micro-flows (where a micro-flow represents a single TCP/UDP connection) per aggregate from 10 to 200. We compute the following performance metrics:

Throughput. This denotes the average bytes received by the receiver application over simulation time. A higher throughput usually means better service for the application (e.g., shorter completion time for an FTP flow). For the ISP, higher throughput implies that links are well-utilized.

Packet Drop Ratio. This is the ratio of total packets dropped to the total packets sent. A user can specify for certain applications that packet drop should not exceed a certain threshold.

Packet Delay. For delay sensitive application like Telnet, the packet delay is a user metric.

Response Time. This is the time between sending a request to a Web server and receiving the response back from the server.

7. SIMULATION RESULTS

The objective of this preliminary set of experiments is to evaluate the effectiveness of the three components of our edge router. In the next few sections, we study the performance of each component under various conditions.

7.1 Detecting Attacks and SLA Violations

In this section, we investigate the accuracy and effectiveness of the delay, loss, and throughput approximation methods for detecting violations discussed in section 3. We use a similar network topology to that used in [12] as depicted in figure 4. We connect multiple hosts to all edges to create several flows along all links in the topology. Many flows are created from hosts attached to $E1$, $E2$, and $E3$, and destined to hosts connected to edge router $E6$

so that the link $C4 - E6$ is highly utilized. We first measure delay when the network is correctly provisioned or over-provisioned (and thus experiences little delay and loss). The delay of $E1 - E6$ is 100 ms; $E1 - E7$ delay is 100 ms; and $E5 - E4$ delay is 160 ms. Attacks are simulated on router $E6$ through links $C3-C4$ and $C4-E6$. With the attack traffic, the average delay of the $E1 - E6$ link increases from 100 ms to approximately 180 ms. Since all the core router to core router links have a higher capacity than other links, $C4 - E6$ becomes the most congested link. Figure 5 shows that when there is no attack, the end-to-end delay is close to the link transmission delay. As seen from the simulations, excess traffic introduced by the attacker increases the edge-to-edge delay inside the network domain. The frequency of delay probing is a critical parameter that affects the accuracy of the estimation. Sending fewer probes reduces overhead but using only a few probes can produce inaccurate estimation, especially when some of the probes are lost in the presence of excess traffic due to an attack.

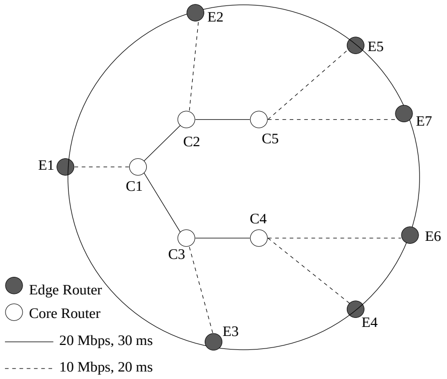


Figure 4: Topology used to infer loss and detect service violations. All edge routers are connected to multiple hosts.

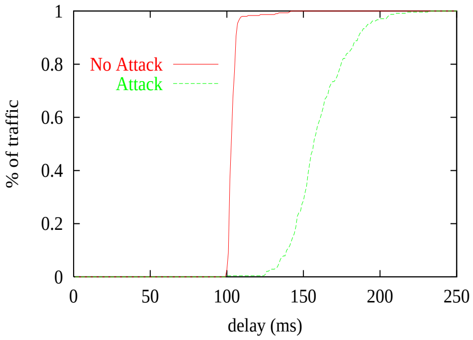


Figure 5: Cumulative distribution function (CDF) of one way delay from E1 to E6

We demonstrate detection of such abnormal conditions using delay measurements in three scenarios labeled “No attack”, “Attack 1”, and “Attack 2” in figure 6. “No attack” indicates no significant traffic in excess of capacity. This is the normal case of proper network provisioning and traffic conditioning at the edge routers. Attacks 1 and 2 inject more traffic into the network domain from different ingress points. The intensity of the attack is increased during time *t=15* seconds to *t=45* seconds. Loss is inferred when high delay is experienced inside the network domain. To infer loss inside a QoS network, green, yellow, and red probes are used. We use equation (2) to compute overall traffic loss per class in a QoS network. The loss measurement results are depicted in figure 7. The loss fluctuates with time, but the attack causes packet drops of 15% to 25% in the case of Attack 1 and more than 35% with Attack 2. We find that it takes approximately 10 seconds for the inferred loss to converge to the same value as the real loss in the network. Approximately 20 stripes per second are required to infer a loss ratio close to the actual value. For more details on the probing frequencies and convergence of the estimations, see [22].

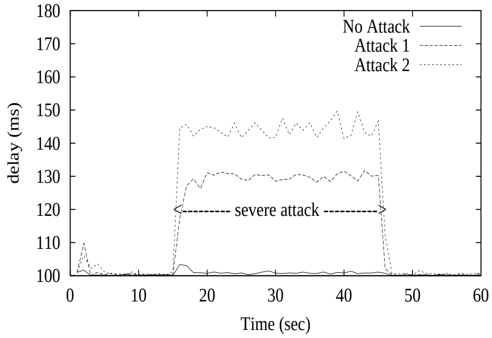


Figure 6: Observed delay at the time of an attack.

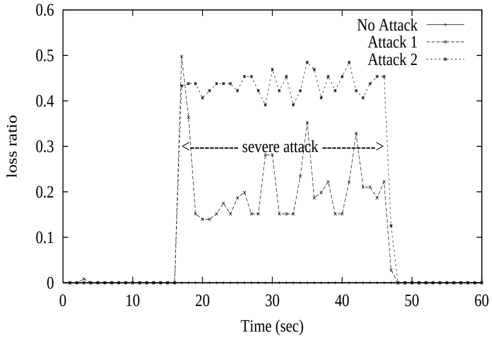
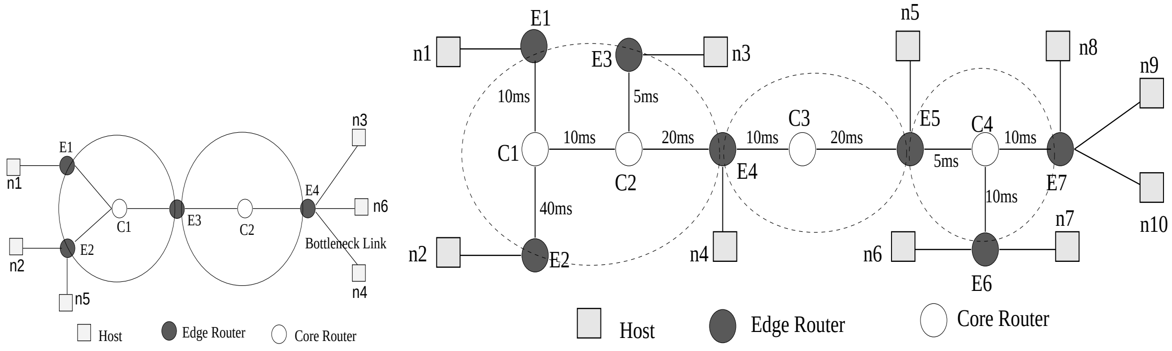


Figure 7: Overall loss follows the same pattern as delay.

Delay and loss estimation, together with the appropriate thresholds, can thus indicate the presence of abnormal conditions, such as distributed DoS attacks and flash crowds. When the SLA monitor detects such an anomaly, it polls the edge devices for throughputs of flows. Using these outgoing rates at egress routers, the monitor computes the total bandwidth consumption by any particular user. This bandwidth is compared to the SLA bandwidth. By identifying the congested links and the egress routers connected to the congested links, the downstream domain where an attack or crowd is headed is identified. Using IP prefix matching, we determine whether many of these flows are aggregated towards a specific network or host. If the destination confirms this is an attack, we can control these flows at the ingress routers.

7.2 Adaptive Conditioning

As discussed in section 4, TCP-aware marking can improve application QoS. We first perform several experiments to study each marking technique separately and study all combinations. We find that protecting SYN packets is useful for short-lived connections and very high degrees of multiplexing. Protecting connections with small window sizes (SW) contributes the most to total bandwidth gain, followed by protecting CWR packets and SYN. SW favors short RTT connections, but it reduces packet drop ratio and timeouts for long RTT connections as well, compared to a standard traffic conditioner. Not marking in bursts is effective for short RTT



(a) Simple topology

(b) Multiple domain topology

Figure 8: Simulation topologies. All links are 10 Mbps. Capacity of the bottleneck links are altered in some experiments.

connections. If SW is not used, Burst+CWR achieves higher bandwidth than any other combination. The RTT-RTO based conditioner mitigates the RTT-bias among short and long RTT flows. This is because when the congestion window is small, there is a higher probability of time-outs 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. A micro flow also recovers from time-outs when RTO as well as RTT is used to mark packets. All these marking principles are integrated together with an adaptive state replacement policy, as given in figure 3. We now evaluate the performance of this adaptive traffic conditioner with FTP and CBR traffic, Telnet and WWW applications. The network hosts and routers are ECN-enabled for all experiments in this section, since we use the ECN and CWR packet protection mechanism. Additional results can be found in [23].

Figure 9(a) compares the bandwidth with the standard and with the adaptive (figure 3) conditioner for the simple topology shown in figure 8 (a). The total throughput is measured over the simulation time at the receiving end. *"Max"* is the bandwidth when the standard conditioner is combined with all marking techniques and stores per-flow information for all flows. The adaptive conditioner outperforms the standard one for all aggregate flows. The adaptive conditioner is more fair in the sense that short RTT flows do not steal bandwidth from long RTT flows and total achieved bandwidth is close to 10 Mbps (bottleneck link speed).

Figure 8(b) depicts a more complex simulation topology where three domains are interconnected (all links are 10 Mbps). The link delay between host and the edge is varied from 1 to 10 ms for different hosts connected to a domain to simulate users at variable distances from same edge routers. Aggregate flows are created between nodes $n1$-$n8$, $n2$-$n9$, $n3$-$n4$, $n5$-$n6$, and $n7$-$n9$. Thus, flows are of different RTTs and experience bottlenecks at different links. Not all flows start/stop transmission at the same time– flows last from less than a second to a few seconds. $C2$-$E4$, $E5$-$C4$ and $C4$-$E7$ are the most congested links. Figure 9(b) shows the total bandwidth gain for this topology with different conditioners. From the figure, the adaptive conditioner performs better than the standard one, and achieves performance close to the maximum capacity. In addition, the adaptive conditioner improves fairness between short and long RTT flows, without requiring large state tables.

When each aggregate flow contains 200 micro flows, the soft state table for the adaptive conditioner covers only a small percentage (4.16%) of the flows passing through it. We use a table for the 50 most recent micro-flows. Table 1 shows that the bandwidth achieved with the adaptive conditioner always outperforms standard conditioner. Note that when critical TCP packets are protected, they are charged from the user profile to ensure that UDP traffic is not adversely affected.

Micro flows	Standard Bandwidth	Adaptive Bandwidth	*Max* Bandwidth	Adaptive (% flows covered at E4)
10	12.65	12.87	12.87	41.16
50	12.18	13.84	14.20	16.66
100	11.67	13.48	14.89	8.33
200	11.77	13.61	14.91	4.16

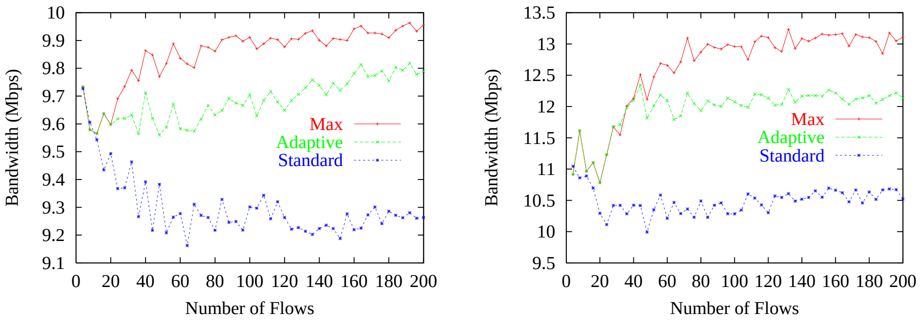
Table 1: Bandwidth shown is in Mbps. State table size = 50 micro-flows.

Conditioner	Avg response time (sec), first pkt	Std dev	Avg response time (sec), all pkts	Std dev
Standard	0.48	0.17	2.23	0.78
Adaptive	0.45	0.14	2.15	0.75

Table 2: Response time for WWW traffic. Number of concurrent sessions = 50

We also study performance with Telnet (delay-sensitive) and WWW (response time sensitive) applications. For the Telnet experiments, the performance metric used is the average packet delay time for each Telnet packet. We use the topology is Figure 8(b), but capacity of the $C1$-$E4$ and $E5$-$E7$ links is changed to 0.5 Mbps and all other link capacities are 1 Mbps to introduce congestion. We simulate 100 Telnet sessions among hosts $n1$-$n8$, $n2$-$n9$, $n3$-$n4$, $n5$-$n6$, and $n7$-$n9$. A session transfers between 10 and 35 TCP packets. Results show that the adaptive conditioner reduces packet delay over the standard conditioner for short RTT flows.

Since web traffic constitutes most (60%-80%) of the Internet traffic, we study our traffic conditioner with the WWW traffic model in **ns-2** [30]. Details of the model are given in [15]. The model uses HTTP 1.0 with TCP Reno. The servers are attached to $n6$, $n8$ and $n9$ in figure 8 (b), and $n1$, $n2$ and $n5$ 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. The default **ns-2** probability distribution parameters are used to generate inter-session time, inter-page time, objects per page, inter-object time, and object size (in kB). The network setup is same as with Telnet traffic. Table 2 shows the average response time per WWW request received by the client. Two response times are shown in the table; one is to get the first packet and another is to



(a) Simple topology. Adaptive state table size=20 micro-flows

(b) Multi-domain topology. Adaptive state table size=50 micro-flows

Figure 9: Achieved bandwidth with the standard conditioner and adaptive conditioner. *"Max"* is the bandwidth when the standard conditioner is combined with all marking techniques and stores per-flow information for all flows.

get all data. The table shows that our adaptive conditioner reduces response time over the standard traffic conditioner. The adaptive conditioner does not change the response time significantly if the network is not congested.

7.3 Congestion Control

We conduct experiments to demonstrate the role of the congestion control mechanism in preventing congestion collapse. Figure 8(a) depicts the simple topology used to demonstrate congestion collapse due to unresponsive flows. An aggregate TCP flow with 10 micro-flows from host $n1$ to $n3$ and a UDP aggregate flow with 10 micro-flows from host $n2$-$n4$ are created. Both flows have the same target rate (5 Mbps). Figure 10 shows how TCP and UDP flows behave with respect to changing the bottleneck bandwidth ($btlnkbw$) from $1 - 5$ Mbps. The x-axis denotes the $btlnkbw$ and y-axis gives the throughput achieved by both flows. Figure 10(a) shows that the TCP flow gets its share of 5 Mbps all the time because it does not go through the congested link. When the bottleneck bandwidth is 1 Mbps, 4 Mbps are wasted by UDP flows in the absence of the flow control. Figure 10(b) shows that, with flow control, the TCP flow gets an extra 8 Mbps when $btlnkbw$ is 1 Mbps. The flow control mechanism prevents congestion collapse due to undelivered packets.

We also experiment with varying the rate ratio, $R_r = \frac{Sending Rate}{Profile}$ for UDP traffic. A R_r of 0.5 means that the flow is sending at 50% of its profile and a R_r of 4 means the flow is sending at four times its profile. When the UDP sending rate is zero, TCP can use the entire 10 Mbps, and there is no shaping (shaping drop is zero) at the edge. When the UDP sending rate causes drops at the bottleneck link (e.g., when $btlnkbw$= 1 Mbps), congestion collapse occurs in the absence of flow control. With flow control, even when R_r is 4 (the profile is 5 Mbps and UDP is sending at 20 Mbps), there is no congestion collapse.

A more complex topology with multiple domains (figure 8(b)) and with cross traffic is also used to study the flow control framework. An aggregate of TCP flows $F1$ between $n1 - n8$ is created, in addition to several UDP flows such as $F2$, $Cr1$, $Cr2$, and $Cr3$ between $n2 - n9$, $n3 - n4$, $n5 - n6$, and $n7 - n10$ respectively. These Crs are used as cross traffic. The start and the finish times of the Crs flows are set differently to change the overall traffic load over the path for the flows $F1$ and $F2$. There are 10 micro flows per aggregate in this setup. Flows $F1$ and $F2$ have same profile with target rate 5 Mbps, and cross traffic sending rate is 2 Mbps.

Figure 11 illustrates the bandwidth of these aggregate flows with and without flow control. The cross traffic achieves the same target in both schemes, because the flows do not send more than their profiles and they do not encounter any bottleneck. If there is no flow control, $F1$ (TCP) cannot achieve its target 5 Mbps. With flow control, $F1$ obtains more than the target. This is because, after controlling UDP, TCP uses the remaining bandwidth.

8. CONCLUSIONS

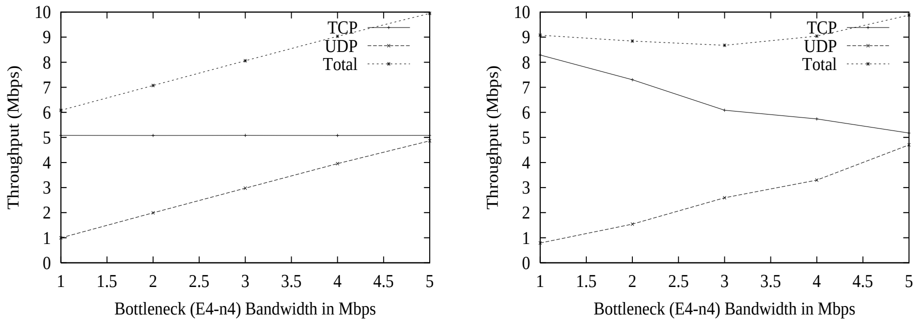
We have investigated tomography-based edge-to-edge probing methods to detect service level agreement violations in QoS networks, together with TCP-aware conditioning and flow control for unresponsive flows. SLA violation detection is useful for network re-dimensioning, as well as for detecting distributed denial of service attacks. We design methods that use edge-to-edge packet stripes to infer loss for different drop precedences in a QoS network, based on observed delays. Aggregate throughputs are then measured to detect distributed denial of service attacks or flash crowds.

Marking, shaping, and policing are also adapted to respond to detection results and adapt to flow characteristics. We give priority to critical TCP packets and mark according to flow characteristics. We use an adaptive conditioner that overwrites previous state information based on a least recently used strategy. Marking is based on information in packet headers if state information for a flow is unavailable. The adaptive conditioner is shown to improve FTP throughput, reduce packet delay for Telnet, and response time for WWW traffic. The conditioner also mitigates TCP RTT bias if it can deduce the flow RTT and RTO. Finally, we have designed a simple method to regulate unresponsive flows to prevent congestion collapse due to undelivered packets.

Most of our ideas can be applied to any architecture that supports service differentiation, or directly with active queue management techniques at network routers. For example, the RED algorithm at network routers can itself protect critical TCP packets, e.g., CWR marked packets, from drop without requiring any additional state. The adaptive conditioner concept can also be employed to keep some window size information and use that in RED dropping decisions. We are currently implementing the edge router, and setting up a simple testbed to validate the simulation results of our framework.

9. REFERENCES

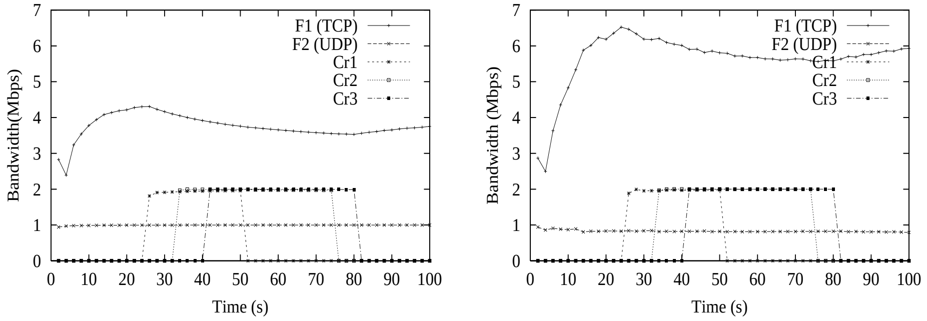
[1] A. Adams and et al. The use of end-to-end multicast measurements for characterizing internal network behavior.



(a) No Flow Control

(b) With Flow Control

Figure 10: a. Without flow control and the TCP gets only 5 Mbps when bottleneck bandwidth is 1 Mbps of topology in Figure 8(a). b. With Flow control and now TCP gets 8 Mbps. Both flows have the same profile.



(a) No Flow Control

(b) With Flow Control

Figure 11: Dynamic adjustment of F2 flow works fine in the presence of cross traffic. TCP flow (F1) gets more bandwidth with the flow control scheme.

IEEE Communications, 38(5), May 2000.

[2] C. Albuquerque, B. Vickers, and T. Suda. Network Border Patrol. *in Proc. IEEE INFOCOM*, 2000.

[3] D. Anderson, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. *in Proc. ACM Symp on Operating Systems Principles (SOSP)*, Banff Canada, Oct 2001.

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

[5] B. Braden and et al. Recommendations on queue management and congestion avoidance in the internet. RFC 2309, April 1998.

[6] Y. Breitbart and et al. Efficiently monitoring bandwidth and latency in IP networks. *in Proc. IEEE INFOCOM*, Alaska, April 2001.

[7] R. Cáceres, N. G. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions on Information Theory*, Nov 1999.

[8] M. C. Chan, Y.-J. Lin, and X. Wang. A scalable monitoring approach for service level agreements validation. In *Proceedings of the International Conference on Network Protocols (ICNP)*, pages 37–48, Nov 2000.

[9] H. Chow and A. Leon-Garcia. A feedback control extension to differentiated services. Internet Draft, draft-chow-diffserv-fbctrl-00.pdf, March 1999.

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

[11] M. Dilman and D. Raz. Efficient reactive monitoring. *in Proc. IEEE INFOCOM*, Alaska, April 2001.

[12] N. G. Duffield, F. Lo Presti, V. Paxson, and D. Towsley. Inferring link loss using striped unicast probes. *in Proc. IEEE INFOCOM*, April Alaska, April 2001.

[13] S. Fahmy. *New TCP standards and flavors, High Performance TCP/IP networking*, chapter 13. Prentice Hall, Inc., 2002.

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

[15] 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. *in Proc. ACM SIGCOMM*, pages 301–313, 1999.

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

[17] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing agreements performance monitoring. RFC 2827, May 2000.

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

[19] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, August 1999.

[20] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993. ftp://ftp.ee.lbl.gov/papers/early.ps.gz.

[21] A. Habib, B. Bhargava, and S. Fahmy. A round trip time and timeout aware traffic conditioner for differentiated services networks. *in Proc. IEEE International Conference on Communication (ICC), New York*, Apr-May 2002.

[22] A. Habib, S. Fahmy, S. R. Avasarala, V. Prabhakar, and B. Bhargava. On detecting service violations and bandwidth theft in QoS network domains. *Computer Communications*, 2002.

[23] A. Habib, S. Fahmy, and B. Bhargava. Design and Evaluation of an Adaptive Traffic Conditioner in Differentiated Services Networks. *in Proc. IEEE International Conference on Computer Communication and Networks (IC3N), Arizona*, pages 90–95, Oct 2001.

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

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

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

[27] W. Lin, R. Zheng, and J. Hou. How to make Assured Services more assured. *in Proc. ICNP*, Oct 1999.

[28] R Mahajan and et al. Controlling high bandwidth aggregates in the network. Technical Report, ACIRI, Feb 2001.

[29] 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.

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

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

[32] T. Ott, T. Lakshman, and L. Wong. SRED: Stabilized RED. *in Proc. IEEE INFOCOM*, March 1999.

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

[34] K. Ramakrishnan and S. Floyd. A proposal to add Explicit Congestion Notification (ECN) to IP. RFC2481, January 1999.

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

[36] N. Seddigh, B. Nandy, and P. Pieda. Study of TCP and UDP interaction for the AF PHB, 1999.

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

[38] H. Wu, K. Long, S. Cheng, and J. Ma. A Direct Congestion Control Scheme for Non-responsive Flow Control in Diff-Serv IP Networks. Internet Draft, draft-wuht-diffserv-dccs-00.txt, August 2000.

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

Appendix

Percentages used in multi-priority loss inference:

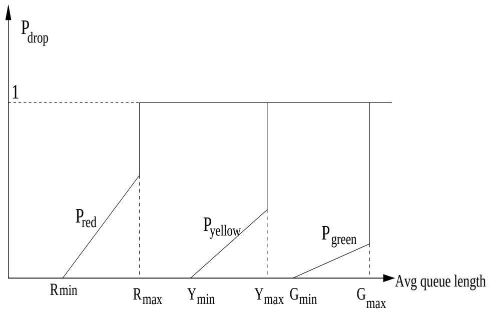


Figure 12: RED Parameters for Three Drop Precedences

Figure 12 depicts the drop probabilities in RED with three drop precedences. The *red* traffic has higher drop priority than *yellow* and *green* traffic. The red traffic is dropped with a probability P_{red} when the average queue size lies between two thresholds R_{min} and R_{max}. All incoming red packets are dropped when the average queue length exceeds R_{max}. P_{yellow} and P_{green} are similar. Suppose $\alpha_G(n)$ is the probability that an incoming green packet will be accepted by the queue given that n packets are in the queue. $\alpha_Y(n)$ and $\alpha_R(n)$ are defined similarly for yellow and red traffic respectively. The α values for green packets are defines as follows:

$$\begin{aligned}
&\alpha_G(n) = 1, && if \quad n < G_{min}\\
&\alpha_G(n) = 0, && if \quad n > G_{max}\\
&\alpha_G(n) = 1 - P_{green}\frac{n - G_{min}}{G_{max} - G_{min}}, && otherwise
\end{aligned} \tag{9}$$

The equations are similar for yellow and red traffic. Let $\mathcal{P}'_{red}$ be the percentage of packet drops due to active queue management for red packets, and let $\mathcal{P}'_{yellow}$ and $\mathcal{P}'_{green}$ be defined similarly for yellow and green respectively. These percentages can be computed as:

$$\mathcal{P}'_{red} = \frac{R_{max} - R_{min}}{R_{max}} \times P_{red} + \frac{G_{max} - R_{max}}{B} \times 100 \tag{10}$$

$$\mathcal{P}'_{yellow} = \frac{Y_{max} - Y_{min}}{Y_{max}} \times P_{yellow} + \frac{G_{max} - Y_{max}}{B} \times 100 \tag{11}$$

$$\mathcal{P}'_{green} = \frac{G_{max} - G_{min}}{G_{max}} \times P_{green} \tag{12}$$

where B is the buffer (queue) size at the router. The percentage of class k traffic accepted by an active queue can be expressed as:

$$\mathcal{P}_k = 1 - \mathcal{P}'_k \tag{13}$$