Directions for Web and e-commerce applications security

by Christopher Clifton
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

ON DETECTING SERVICE VIOLATIONS AND BANDWIDTH THEFT IN QOS NETWORK DOMAINS

Ahasan Habib
Sonia Fahmy
Srinivas R. Avasarala
Venkatesh Prabhakar
Bharat Bhargava

CSD TR #01-022
December 2001

On Detecting Service Violations and Bandwidth Theft in QoS Network Domains *

Ahsan Habib, Sonia Fahmy, Srinivas R. Avasarala[†] Venkatesh Prabhakar[‡] Bharat Bhargava
CERIAS and Department of Computer Sciences
Purdue University, West Lafayette, IN 47907–1398, USA
email: {habib,fahmy,bb}@cs.purdue.edu, sra@andiamo.com, vp@senera.net

Abstract

We design and evaluate a simple and scalable system to verify Quality of Service (QoS) in a differentiated services domain. The system uses a distributed edge-to-edge monitoring approach with measurement agents collecting information about delays, losses and throughput, and reporting to a Service Level Agreement Monitor (SLAM). The SLAM detects potential service violations, bandwidth theft, denial of service attacks, and flags the need to re-dimension the network domain or limit its users. Measurements may be performed entirely edge-to-edge, or the core routers may participate in logging packet drop information. We compare the core-assisted and edge-to-edge schemes, and we extend network tomography-based loss inference mechanisms to cope with different drop precedences in a QoS network. We also develop a load-based service monitoring scheme which probes the appropriate edge routers for loss and throughput on demand. Simulation results indicate that the system detects attacks with reasonable accuracy, and is useful for damage control in both QoS-enabled and best effort network domains.

Keywords: Service Level Agreements, Network Tomography, Network Monitoring, Network Security, Quality of Service.

1 Introduction

Internet security lapses have cost U.S. corporations 5.7 percent of their annual revenue, as reported by University of California at Davis economist Frank Bernhard [13]. Specifically, the increase in the number of denial of service attacks (12,805 reported by the San Diego Supercomputer Center in February 2001 [24]), implies that bandwidth theft attacks can become widespread in networks with Quality of Service (QoS) support. Hence, monitoring network activity is required to maintain confidence in the security and QoS of networks, from both the user (ensuring the service level paid for is indeed obtained) and provider (ensuring no unusual activity or attacks take place) perspectives. Developing a low cost distributed monitoring system is the primary focus of this paper.

We use the differentiated services (DS) QoS framework as an underlying network, though our system is not specific to DS. Packets entering a DS domain are classified and the DS field in the IP header is marked at the edge router [25]. The packets then experience specific per-hop behaviors (PHBs) as they are forwarded by the interior (core) routers of the domain depending on their DS field. Currently, the Expedited Forwarding (EF) PHB [19] and the Assured Forwarding (AF) PHBs [18] have been defined. The EF PHB can be used to build a low loss, low latency, end-to-end service. The AF PHB offers different levels of forwarding assurances, each with three drop precedences (e.g., green, yellow and red). Typically, a user has a service level agreement (SLA) with a provider that describes the expected service, user traffic profile, and charging models. The provider uses SLAs, along with other mechanisms, to provision the network appropriately.

Differences in charging models of the service classes can attract attacks that inject marked packets to steal bandwidth and other network resources. Such attacks make use of known vulnerabilities in firewall filter rules to inject

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

[†] Now with Andiamo Systems, San Jose, CA 95134

[‡] Now with Senera Systems, Sunnyvale, CA 94086

traffic or spoof the identity of valid users with high QoS levels. Since the DS framework is based on aggregation of flows into service classes, valid user traffic may experience degraded QoS as a result of the injected traffic. Taken to an extreme, the attacks may result in denial of service. This creates a need for developing an effective defense mechanism that can automate the detection and reaction to attacks on the QoS-provisioned DS network domain.

Although measurement of path characteristics [26, 27] and network monitoring [5, 14, 20] have been extensively investigated, few studies of user SLA validation have been performed [10]. Inspired by recent results on network tomography [1, 6, 7], we infer internal characteristics of a network domain using edge-to-edge probes, and design a distributed monitoring system to detect service violations and bandwidth theft in a network domain. We employ agents on selected routers of the DS domain to efficiently measure packet delays, loss, and throughputs. Measurements are communicated to an SLA Monitor (SLAM). The SLAM analyzes measurements and automatically detects potential attacks and violations of negotiated SLAs, as well as flag the need to re-provision the network by increasing capacity or limiting users.

We also compare core-assisted and pure edge-to-edge approaches for packet loss ratio computation. The comparison can help network providers decide which technique best serves their needs. We inject probes only when necessary to reduce communication overhead. Moreover, we extend stripe-based loss inference [15] to cope with different drop precedences in a QoS network. Throughput measurements are only performed when a delay or loss violation is reported. As with any detection mechanism, the attackers can attach the mechanism itself, but we assume the cost to attack this distributed monitoring mechanism is higher than the cost to inject or spoof traffic, or bypass a single edge router.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 describes the high-level architecture used in our service violation detection system. Section 4 gives the methodology of measurements and SLA violation detection. Section 5 discusses when to probe the network based on network load. Section 6 presents our simulation experiments and results. Finally, section 7 summarizes our conclusions and recommendations for enhanced security.

2 Related Work

A number of related studies have investigated differentiated services security, measurements of QoS parameters, network tomography and monitoring, and SLA verification.

2.1 Network Security

A security analysis for the differentiated services framework is provided in [32]. QoS attacks are classified as either attacking the *network provisioning process*, or attacking the *data forwarding process*. Network provisioning involves configuration of DS nodes by policy distribution points in the network (Bandwidth Brokers (BBs)), through RSVP [4] or SNMP [9]. This process can be attacked by injecting bogus configuration messages, modifying the content of real configuration messages, delaying or dropping such messages. Networks can be secured against such attacks by employing encryption of the configuration messages. Attacks on the data forwarding process are of a more serious nature and can involve injecting traffic into the network with an intent to steal bandwidth or to cause QoS degradation by causing other user flows to experience longer delays, higher loss rates, and lower throughput. Our goal is to detect attacks on the data forwarding process by monitoring the characteristics of a network domain.

2.2 Performance Measurements

A large body of research has focused on measuring delay, loss, and throughput in the Internet [26, 27]. Shared Passive Network Performance Discovery (SPAND) [29] is a tool that communicates with distant Internet hosts and reports to a performance server in the same domain. Sharing history to improve future measurement was proven useful. Savage et al propose *Detour* routers as edge devices in Internet clouds that will tunnel traffic to improve Internet performance [28]. These edge routers exchange bandwidth, latency, drop rate among themselves. We also employ intelligent routers at key access points that monitor a network domain. Resilient Overlay Networks (RON) is an architecture to detect and recover from path outages and periods of degraded performance [2]. RON nodes monitor the quality of Internet paths among themselves and use this information to route packets, optimizing application-specific routing metrics. RON uses three different routing metrics: latency, loss and throughput. Measurement techniques in SPAND, Detour, and RON are useful but not directly applicable to violation detection in a QoS network domain.

2.3 Network Tomography

Network tomography is an approach to infer the internal behavior of a network based on purely end-to-end measurements [31]. A number of studies [1, 6, 7] have shown how to infer loss and delay, and discover the topology of a multicast network. Coates and Nowark [11, 12] discuss delay and loss inference using unicast probing in order to monitor TCP flows [31]. Duffield et al [15] use packet "stripes" (back-to-back probe packets) to infer link loss by computing the correlations among packet losses within a stripe at the destinations. Using end-to-end unicast probing, the authors demonstrate how to infer loss characteristics of the links in the network interior. We extend this technique to infer loss in a QoS domain and show how to detect service violations and attacks in that domain based on inferred values.

2.4 Network Monitoring

Many proposals for network monitoring [5, 14] ensure that a network is operating within desirable parameters. In efficient reactive monitoring [14], the authors discuss ways to monitor communication overhead in IP networks. Their main idea is to combine global polling with local event driven reporting. Our core-assisted scheme also uses local event driven reporting and performs global polling only when it is absolutely necessary. Breitbart et al [5] identify effective techniques to monitor bandwidth and latency in IP networks. The authors present *probing-based* techniques where path latencies are measured by transmitting probes from a single point of control. The paper describes algorithms to compute an optimal set of probes to measure latency of paths in a network. We focus on monitoring a network domain to detect attacks. For scalability, our approach involves only edge routers in any QoS parameter measurement.

2.5 SLA Verification

In [10], a histogram-based aggregation algorithm is used to detect SLA violations. The algorithm measures network characteristics on a hop-by-hop basis and uses them to compute end-to-end measurements and validate end-to-end SLA requirements. In large networks, efficient collection of management data is a challenge. While exhaustive data collection yields a complete picture, there is an added overhead. Furthermore, the authors assume that the routes used by SLA flows are known, citing VPN and MPLS [8] provisioning. We use average values to reduce constraints on the network setup, and eliminate the need for knowledge of the set of flows traversing each router.

3 Architecture for SLA Violation Detection

Differentiated Services (DS) [3] pushes complexity to boundary devices which process lower volumes of traffic. The boundary routers where traffic enters a domain, called ingress routers, perform traffic conditioning that consists of traffic classification based on multiple fields in the packet header, traffic metering to ensure conformance to a profile, marking, dropping, shaping or remarking of out-of-profile traffic. Core routers perform simple forwarding based on the DS field. SLAs between the user and provider networks are used to derive filter rules for traffic classification at the ingress routers. Therefore, ingress routers with appropriate configuration of filter rules should prevent non-conforming traffic from entering a DS domain. Though ingress routers serve as a good first line of defense, attackers can still succeed in injecting non-conforming traffic into a DS domain in a variety of ways, e.g.:

1. Attackers can impersonate a legitimate user by spoofing flow identity (IP addresses, protocol and port numbers). Network filtering [16] at routers in the user network can detect such spoofing if the attacker and the impersonated user are on different subnets, but the attacks proceed unnoticed otherwise.

2. Attackers can devise mechanisms to bypass the ingress routers by exploiting some well known vulnerabilities in the firewall filters. Thus, they can inject traffic with their own identity and a desired destination. Alternatively, the traffic can be aggregated from multiple ingress routers.

3. Legitimate users can send traffic in excess of their profiles. Ingress routers will re-mark excess traffic with a code point of a lower service class, e.g., AF red packets or best effort, which affects other user flows of that lower class, as in a denial of service attack.

Such attacks and others escape detection at ingress routers. Co-ordination among boundary routers or support from core routers is required for detection. Changes that can be observed due to the attack traffic in the network include longer per-packet delays, higher average buffer occupancy, and higher packet drop rates. We use these characteristics, specifically delays, loss ratios, and bandwidth achieved by flows *after* aggregation within the domain to detect bandwidth theft attacks and service violations.

Figure 1 depicts our proposed architecture. An SLA Monitor (SLAM) coordinates the monitoring activities inside the DS domain. In the figure, the SLAM is shown as a separate entity in a network domain. However, any edge router can take this responsibility as long as it has sufficient resources and computing capabilities. Loss ratios may be inferred on a pure edge-to-edge basis or using core router assistance. In the core-assisted scheme, egress and core routers send delay and loss measurements respectively to the SLAM for the flows in the domain. Upon request, the ingress sends the number of packets entering a domain per flow to calculate loss ratio. The packet loss is computed as the ratio of the packet drop inside a domain to the total packets entering the domain. Loss ratio of a flow is a better metric than loss rate (i.e., number of drops per second). Another alternative is to measure delay, loss or throughput using only edge routers using network tomography techniques. The SLAM maintains delay and loss information of misbehaving flows only. In addition, the SLAM maintains the SLA parameters for each user for a certain domain. By comparing the delay and loss measurements against the specific user SLA, we can identify potential SLA violations.

4 QoS Parameter Measurement

The SLA parameters used for detecting violations include delay, loss, and throughput. This section describes methods to measure and use these parameters to detect service violations.

4.1 Delay Measurements

Delay bound guarantees made by a provider network to user traffic flows are for the delays experienced by the flows between the ingress and egress routers of the provider domain. Delay measurements either use delay of real user traffic or injected traffic. The first approach is intrusive because encoding timestamps into the data packets would require changing the packets at the ingress and rewriting the original content at the egress after appropriate measurements. The second approach is non-intrusive in that we can inject probe packets with desired control information to enable an egress router to recognize such probes, perform measurements and delete the probes from the traffic stream. We adopt the second approach in our design. For each packet traversing an ingress router, with a certain pre-configured probability p_{probe}, the ingress copies the packet IP header into a new probe packet. A timestamp $t^i_{ingress}$ is encoded into the payload of flow i, and an identifier field is marked with a new value in the probe packet. The egress router removes probes from the traffic stream, and computes $delay^i_j$ for a packet from flow i of user j traffic as:

$$delay^i_j = t^i_{egress} - t^i_{ingress} \tag{1}$$

where t^i_{egress} is the time the packet of flow i traverses the egress router. The egress forwards the packet details and the measured delay information to the SLAM. The encoded timestamp should follow a well-known format, e.g., Coordinated Universal Time (UTC), and a standard protocol like Network Time Protocol (NTP) should be used to maintain clock synchronization. Alternatively, the two-way delay from ingress to egress and back to ingress can be divided by two if links are approximately symmetric. At the SLAM, we classify the packet as belonging to flow i of user j and update the average packet delay of user j traffic, avg_delay_j, using an exponential weighted moving average (EWMA):

$$avg\_delay_j = \alpha \times avg\_delay_j + (1 - \alpha) \times delay^i_j \tag{2}$$

where α is a small fraction to emphasize recent history rather than the current sample alone. If this average packet delay exceeds the delay guarantee in the SLA, we conclude that a service violation may have occurred. If the network is properly provisioned and all flows do not misbehave, delay for user j should not exceed its delay guarantee.

Determining the probability with which we should inject probe packets is not an easy task. If there are M edge routers in a network domain, N^i flows (on the average) passing through an edge router i, and p^{ij}_{probe} is the probability that an edge router i and flow j will be selected to probe for latency, then $MN^i p^{ij}_{probe}$ is the average number of probe packets injected into the network domain. To keep the volume of these control messages low, we must select a low probability. However, if the probability is too low, the chance of undetected violations is higher. Therefore, we vary the probing probability value dynamically over time at each edge router. The change in this probability is performed at all edge routers autonomously making sure the edges do not use the same random number generator sequence or seed.

4.2 Loss Measurements

Packet loss guarantees made by a provider network to a user are for the packet losses experienced by its conforming traffic inside the provider domain. To compute the loss ratio (rather than the less meaningful loss rate), the number of packet drops, as well as the number of packets traversing the domain, are required. Core routers can detect the number of packets dropped, and edge routers can compute the number of packets traversing the domain. We refer to this loss measurement mechanism as the *core-assisted* scheme for loss measurement. An alternative mechanism is to use stripe-based probing to infer loss characteristics inside a domain [15]. A series of probe packets is sent, with no delay between the transmission of successive packets, or what is known as a "stripe." The scheme, which was designed as an end-to-end scheme, can be adapted to the edge-to-edge scenario and to QoS networks. We refer to this strategy as the *edge-to-edge* (or *stripe-based*) loss measurement scheme.

4.2.1 Edge-to-Edge Stripe-based Loss Inference

For a two-leaf binary tree spanned by the nodes 0, k, R_1, R_2 (see figure 2), stripes are sent from the root 0 to the two leaves to estimate the characteristics of each of the 3 links, as proposed in [15]. If a packet reaches a receiver, we can infer that the packet reached the branch point k. The first two packets of a 3-packet stripe are sent to a receiver, e.g., R_2, and the last one to the other receiver. A complementary stripe is sent in the reverse manner. The transmission probability A_k for node k can be computed as:

$$A_k = \frac{Z_{R_1} Z_{R_2}}{Z_{R_1 \cup R_2}} \tag{3}$$

where Z_i represents the empirical mean of a binary variable which takes the value 1 when all packets sent to i reach their destination and 0 otherwise. The mean is taken over n identical stripes. By combining estimates of stripes down each such tree, the characteristics of links $0 - k$, $k - R_1$ and $k - R_2$ can be estimated. This inference technique also extends to general trees. Consider an arbitrary tree where for each node k, $R(k)$ denotes the subset of leaves descended from k. Let $Q(k)$ denote the set of ordered pairs of nodes in $R(k)$. For each $(R1, R2) \in Q(k)$, a stripe should be sent from the root to the receivers R_1 and R_2.

A QoS network domain defines different traffic classes to provide differentiated services. We extend the above unicast probing scheme to routers with active queue management that supports different drop precedences. For example, the assured forwarding (AF) mechanism is realized using several queues where each queue has three drop precedences referred to as green, yellow, and red. 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 is $\geq R_{max}$. Let $\mathcal{P}'_{red}$ be the percentage of packet drops due to the behavior of active queue management for red packets, and let $\mathcal{P}'_{yellow}$ and $\mathcal{P}'_{green}$ be defined similarly for yellow and green packets. These percentages can be computed as:

$$\mathcal{P}'_{red} = \frac{R_{max} - R_{min}}{R_{max}} \times p_{red} + \frac{G_{max} - R_{max}}{B} \times 100 \tag{4}$$

$$\mathcal{P}'_{yellow} = \frac{Y_{max} - Y_{min}}{Y_{max}} \times p_{yellow} + \frac{G_{max} - Y_{max}}{B} \times 100 \tag{5}$$

$$\mathcal{P}'_{green} = \frac{G_{max} - G_{min}}{G_{max}} \times p_{green} \tag{6}$$

where B is the buffer (queue) size.

Let $\mathcal{P}_{red} = 1 - \mathcal{P}'_{red}$ be the percentage of *red* packets accepted by the active queue. We can define percentages for yellow and green traffic similarly using equations (5) and (6). Link loss can be inferred by subtracting the transmission probability (A_k from equation (3)) from 1. Therefore, if L_g, L_y, and L_r are the inferred losses of green, yellow and red traffic, respectively, the loss of a traffic class is expressed as shown in equation (7), where n_i is number of samples taken from traffic of type i:

$$L_{class} = \frac{n_g \mathcal{P}_{green} L_g + n_y \mathcal{P}_{yellow} L_y + n_r \mathcal{P}_{red} L_r}{n_g + n_y + n_r} \tag{7}$$

However, when loss of green traffic is zero, we take the average of yellow and red losses. When the loss of yellow traffic is zero, we report only loss of red probes.

4.2.2 Core-Assisted Loss Measurements

An alternative method to measure loss is to record packet drops for every flow over a time interval Δt seconds at the core routers, and periodically report the values to the SLAM. The SLAM maintains the average number of packets dropped for user j, avg_drop_j, and updates it as:

$$avg\_drop_j = \alpha \times avg\_drop_j + (1 - \alpha) \times drop_j \tag{8}$$

where α is a small fraction, and $drop_j$ is the total packet drop for user j over time interval Δt. The weighted average resolves the problem of wrap-around of the total packet drop count during the life time of a flow. To compute the loss ratio, incoming packet count information is obtained from ingress routers which anyway monitor all flows for profile checking, shaping and marking. This ensures that core routers need not transmit information to the SLAM unless there are sufficient packet drops to suspect attacks or violations.

The procedure to compute the loss ratio, *CalcLossRatio*, executed every Δt seconds, proceeds as follows:

1. Core i reports to the SLAM whenever packet drop of user j, $drop_j^i$, exceeds a *local* threshold

2. The SLAM computes the total drop for time interval Δt, $drop_j = \sum_{i=1}^{N_c} drop_j^i$, where N_c is number of core routers.

3. If the total drop for user j exceeds a global threshold
   a. The SLAM sends a query to all edge routers requesting their current rates for user j
   b. The SLAM computes total incoming rate for user j from all edge routers
   c. The SLAM computes the loss ratio for user j as the ratio of $drop_j$ and the total incoming rate for user j where both values are computed over the same interval
   d. If the loss ratio exceeds the SLA loss ratio for user j, a possible SLA violation is reported

Selecting the local threshold value after which drops are reported by each core router is difficult since the core router does not have any information about the user SLAs. For each AF class k, we must compute a local drop threshold, T_k. May et al [22] give the loss probability for an AF class k as:

$$LR_k = 1 - \sum_{n=0}^{B} k\alpha_k(n)\pi_k(n) \tag{9}$$

where Bk is the buffer size of the class k queue, $\alpha_k(n)$ represents the probability that a class k packet is accepted given that n packets are in the queue and $\pi_k(n)$ is the stationary distribution of the buffer content. The local threshold at each core router for class k packets can thus be set to $T_k = (LR_k + \epsilon)\lambda_j$, where λ_j is the expected arrival rate of user j, and ϵ is a small value. The local threshold limits state maintenance at the core router to the subset of the total number of flows experiencing the highest loss ratio, since we are only interested in flows that result in the aggregate traffic experiencing high loss ratio. If a flow exhibits a high loss ratio, however, this does not mean that this particular flow is misbehaving. Therefore, we employ throughput measurements as discussed next.

4.3 Throughput Measurements

The objective of throughput measurements is to ensure no user is consuming excessive bandwidth and starving others. This may not be detectable by a single ingress router if the user uses multiple ingress routers. The service provider typically employs policies which allow users to consume extra bandwidth of lower service classes. The SLAM probes egress routers for throughput following a loss or delay violation report. Each egress measures the average rate at which user traffic is leaving the network domain. The SLAM computes the throughput for each user as the sum of the bandwidth consumed by the user at all egresses. If this throughput exceeds the SLA bandwidth then there may be a violation. The router may also periodically compute throughput values even in the absence of increased delay or loss as a sanity check.

5 Load-based Monitoring

In this section, we discuss load-based domain monitoring used to probe the network domain for loss and throughput only when attacks or violations are likely. As discussed in section 4, headers of delay probe packets are copied from user packets so that the probes follow the same routes as the user packets. This technique is not applicable to loss measurement using stripes since the SLAM does not know the egress routers from which user packets leave. The stripe-based approach requires a pair of edge routers as receivers to send probe packets to, and we discuss how to determine this pair of routers in this section. If many edge routers are idle or many links are under-utilized, the SLAM does not probe the entire domain for loss or throughput information.

5.1 Probing Strategy

Let E be the set of all domain edge routers (both egress and ingress). One of these routers can act as an SLA Monitor (SLAM), or a separate entity can be used to act as SLAM. The algorithm proceeds as follows:

1. Each ingress router copies the header of user packets with probability p_{probe} to probe the network for delays

2. When an egress router receives these probes, the egress computes the edge-to-edge delay. If the delay exceeds a certain threshold, it reports delay along with the identity of both the ingress and egress routers to the SLAM. There is a trade-off between checking the threshold at the egress versus at the SLAM, because in the former case egresses need to maintain more information about the network domain, while the latter approach increases communication overhead

3. The SLAM maintains the set of edge routers E' to send stripes to, in order to infer loss on active links, where $E' \subseteq E$. The SLAM also maintains a spanning tree of the network topology. A set of edge routers, S_i, which we refer to as complementary edges, is associated with each edge i. The next subsection explains how S_i is constructed. At time t, the SLAM computes the set E' as:

$$E'(t) = \bigcup_i S_i(t) \tag{10}$$

   Since $E' \subseteq E$, the communication overhead of probing for loss will be less than or equal to the communication overhead of probing all edges. Another improvement can be achieved by maintaining another level of nodes in the tree (parents) for each node. The SLAM can save probes by not using nodes at the same level and same parent as complementary nodes for any particular node i in the list.

4. The SLAM probes the network for throughput approximation only when the inferred loss is higher than the pre-configured threshold.

5. Using delay, loss, and throughput approximations, the SLAM can detect violations or bandwidth theft attacks with reasonable accuracy

5.2 Complementary Edges

With stripe-based unicast probing, the source needs to send certain packets of a stripe to a receiver and the remainder of the stripe to a different receiver. Based on which shared link loss needs to be inferred, another edge router (leaf node) of the tree must be used as a complementary receiver. For a given node V, a sibling of V can serve as a complementary node for V. We describe an algorithm to find complementary edges for each edge router of a given tree. These complementary edges will be used to infer link loss from the root to all links up to the closest common ancestor (CCA) of both receivers, and from the CCA to both end receivers. The union of the complementary edges of two edge routers will give all edge routers to use as receivers in the stripe-based methodology to infer loss of required links.

Algorithm *ComplementaryEdges (Tree T, Node V)*: The tree is traversed backwards starting from V.

```
1. $C' \leftarrow \emptyset, P = parent(V)$
2. while $P <>$ root of the tree
       Add leaf $X$ to $C'$ where $CCA(V, X) = P$
       $P \leftarrow parent(P)$
3. return $C'$
```

This algorithm only needs to be executed initially when the network is setup and when it is reconfigured with additional routers/links. The result is stored at the SLAM.

5.3 SLAM Functionality

For each incoming *delay* control packet, the SLAM updates the average delay of the user using equation (2). Then it compares the delay to the SLA delay to detect violations for that user. If the delay has been violated, the SLAM updates the list of edges to send stripes to for loss inference. If there is any loss at the core for the EF traffic class, an SLA violation is flagged. If the inferred AF loss ratio exceeds a certain threshold, the SLAM queries the edges for user throughputs and checks whether there is a throughput violation. The SLAM also compares some report packet DS fields with the flow SLA to ensure that an attack does not occur in the form of injecting packets with a higher service than allowed for that user. For each violation, the SLAM informs the administrator who may choose to throttle that particular user traffic.

6 Simulation Results

We conduct a series of experiments to investigate the delay, loss, and throughput approximation methods described in section 4. We use the **ns-2** simulator [23], with the standard differentiated services implementation by a group from Nortel Networks [30]. TCP New Reno is used, with a packet size of 1024 bytes and a maximum window of 64 packets. We employ a similar network topology to the one in [15] to evaluate both the core-assisted and stripe-based loss ratio approximations. The topology is shown in figure 3. Multiple hosts are connected to all edges to create flows along all links in the topology. A number of flows froñ $E1$, $E2$ and $E3$ are destined to hosts connected to edge router $E6$ to simulate attacks on the link $C4 - E6$.

6.1 Delay, Loss, and Throughput Approximations

We measure delay when the network is adequately provisioned or over-provisioned (and thus experiences little loss) and then we simulate an attack on router $E6$. This scenario is illustrated in figure 4. Under light load, the end-to-end delay of $E1 - E6$ link is 100 ms; $E1 - E7$ delay is 100 ms; and $E5 - E4$ delay is 160 ms. With the attack traffic, the average delay of the $E1 - E6$ link increases up to 180 ms (figure 4(b)). Since all the core router to core router links have a higher capacity than other links, $C4 - E6$ becomes the most congested link, increasing the delay for all traffic traversing $E6$. The delay of the $E5 - E4$ link does not increase because this path is not congested. Therefore, delay patterns are a good indication of the presence of excess traffic inside a network domain. As previously discussed, the frequency of delay probing is a critical parameter. Sending fewer probes reduces overhead but using only a few probes can produce inaccurate estimation, especially that some of the probes are lost in the presence of excess traffic. Figure 5 shows that introducing more delay probes may increase the delay of actual traffic. Figure 5 (b) shows that sending only 5 probes per second is inadequate because as much as 80% of the probes may be lost. Sending probes at a rate of 10 to 15 per second is a good choice in this experiment.

Figure 6 shows loss approximation using the core-assisted scheme. As the scheme uses an exponential weighted moving average of the drop values and the number of incoming packets traversing edge routers, the initial approximated values deviate from the actual values. Thus initial data (the first two seconds) should be discarded. The approximated value is very close to the actual one after that. According to the simulation setup, link $C4 - E6$ exhibits an increased loss ratio for the $E1 - E6$ as depicted in figure 6.

Using striped probes for loss inference (as proposed in [15]) produces reasonably accurate approximations if core routers do not employ active queue management or service differentiation. In assured forwarding, packets marked as "red" have a high drop probability while "green" packets have low drop probability. We send stripes of different colors to infer loss in this case. Figure 7 shows the loss of probes with different drop precedences. Figure 8 depicts the inferred loss of link $C4 - E6$ using these striped unicast probes at different frequencies. The objective of this experiment is to determine how often a stripe should be sent to infer loss accurately. The figure shows that at least 20 stripes per second are required to infer a loss ratio close to the actual value. The figure also demonstrates that a longer time is required for convergence in the striped-based scheme than in the core-assisted scheme.

Figure 9 shows the throughput approximation of different flows traversing a network domain. There are several aggregate flows going through the domain. We measure throughput for flow $F1$ that follows the path $E3$ to $E6$, flow $F2$ that follows path $E1$ to $E6$, flow $F3$ that follows path $E2 - E6$, and flow $F4$ that follows path $E5 - E4$. Other aggregate flows follow paths $E1 - E7$ and $E3 - E7$. The throughput approximation procedure (discussed in section 4) is used to compute the average rate at the egress routers. Figure 9 shows an initial fluctuation between actual and approximated throughput measurements due to the average calculation. After a few seconds, the values are close to each other. Measurement at the egress routers detects distributed attacks entering through different ingress routers of a domain.

6.2 Detecting Attacks and Service Violations

In this section, we demonstrate the detection of mild and severe distributed denial of service attacks. In figures 10 and 11, label "No attack" means the network does not have significant traffic in excess of its capacity. This scenario has little loss inside the network domain. This is the normal case with proper network provisioning and traffic conditioning at the edge routers. Labels "Attack 1" and "Attack 2" denote situations when traffic is injected into the network domain from different ingress points. At each ingress point the flows do not violate any profiles, but the aggregate traffic is excessive. The intensity of the attacks is increased during time *t=15 seconds* to *t=45 seconds*. The delay increases by 30% during Attack 1 and by 50% during Attack 2 (figure 10). Packet drops of 15 to 25% result in case of Attack 1, and drops of more than 35% result with Attack 2, as depicted in figure 11. We use equation (7) to compute overall traffic loss in the QoS network.

The SLA Monitor (SLAM) can thus aid in the detection of denial of service (DoS) and distributed DoS (DDoS) attacks in a network domain. When the SLAM detects an anomaly (high delay and high loss), it polls the edge devices for throughputs of existing flows, in order to detect high bandwidth aggregates. This is similar to the method used in [21], where aggregate-based congestion control (ACC) agents match the prefix of the IP destination addresses to declare high bandwidth flows going to the same destination address. In our core-assisted scheme, the core router similarly sends the packet drop information, together with the source and destination IP addresses to the SLAM. The SLAM performs IP prefix matching to detect any possible DDoS attack through this domain. If there is an attack, the SLAM sends control information to all ingress routers to throttle (filter out) packets of this flow or at least control their rates. The differentiated services architecture can help to propagate such messages to the upstream domain all the way to the source if possible.

6.3 Comparative Evaluation

Based on our experiments, we present a quantitative measure of performance to compare the core-assisted, load-based and edge-to-edge approaches. We used the topology shown in figure 3 to experiment with the approaches. We compare communication overhead, accuracy, convergence time, implementation overhead, and flexibility. We consider a domain $\mathcal{D}$ with M edge routers and N core routers. The total injected probes and size of each probe packet are used to compute the communication overhead in bytes. In the edge-to-edge approach, a stripe of s packets is transmitted from the monitor to every egress routers pair. For the network domain, the total number of probe packets is $s \times (M - 1) \times (M - 2) \times f$, where f is the frequency of stripes per unit time. The communication overhead is therefore $s \times (M - 1) \times (M - 2) \times f \times packet\_size$.

The core-assisted loss measurement scheme overhead depends on the number of packets core routers send to the SLAM to report excessive drop for certain flows. We assume there are F flows traversing each edge router, and each flow has P packets on average. We define θ as the percentage of misbehaving flows. If d bytes are required to record drop information of each flow, then each core needs to send $C = max(1, \frac{F \times \theta \times d}{packet\_size})$ control packets to the SLAM. To compute the *loss ratio*, the monitor queries all edges for packet count information of the misbehaving flows. Every edge will reply to this query. The total number of packets exchanged is $(2M + N) \times C$ packets (recall that N is the number of core routers). Therefore, the communication overhead is $(2M + N) \times C \times packet\_size$. We compute the communication overhead for the core-assisted approach based on attack information provided in [24]. In [24], the authors observed an average of 4268 backscatter attacks per week over a three week period of time by monitoring a sole ingress link into a lightly utilized /8 network. They show that 50% of the attacks last for 10 minutes, 30% last for 30 minutes, 17% last for 60 minutes, 2% last for 5 hours and 1% last for 10 hours or more.

Accuracy is computed using the deviations of approximating the loss ratio from the actual loss ratio value. We calculate accuracy based on our experimental results, with $f = 20$ as the probing frequency for the edge-to-edge approach. Implementation overhead considers which components of the network must be modified. The edge-to-edge approach needs to modify only edge routers, while the core-assisted approach requires change to both edge and core routers. The edge-to-edge scheme is thus considered more flexible since it is easier to deploy. However, the core-assisted approach gives more insight into the performance characteristics of the network domain and has higher accuracy and shorter convergence time. Figure 12 depicts a quantitative comparison of the three approaches. Note that we use a high percentage of misbehaving flows as in [24]. For a large domain with millions of flows per second, the core-assisted approach exhibits a higher communication overhead over a short period with many attacks, but it has a lower overhead over a longer time scale. The load-based approach is the same as the edge-to-edge approach in all respects, except that it reduces communication overhead by probing only the necessary regions.

7 Conclusions

We have investigated methods to detect service level agreement violations in QoS networks. These methods are useful for network re-dimensioning, as well as for detection of denial of service and bandwidth theft attacks. The core-assisted loss measurement method is powerful but difficult to deploy. An alternative edge-to-edge stripe-based loss inference scheme for different drop precedences was thus proposed. In the edge-to-edge probing approach, a low network probing rate has been shown to give incorrect results due to the loss of probes in case of excess traffic caused by an attack. A large number of probes, however, increases actual traffic delay and loss. We have shown that using probes with different drop precedences is necessary to infer loss in a QoS network. Our proposed load-based monitoring technique can aid in detecting attacks such as malicious traffic remarking or injection, without excessive overhead. Our approach can be integrated with an adaptive admission control or flow control scheme to regulate traffic dynamically and control an attack as soon as it is detected. The scheme can be used in any general network architecture (not only a QoS network).

References

[1] A. Adams and et al. The use of end-to-end multicast measurements for characterizing internal network behavior. *IEEE Communications*. 38(5), May 2000.

[2] D. Anderson, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay network. *In proc of 18th ACM Symp on Operating Systems Principles (SOSP)*, Banff Canada, Oct 2001.

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

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP). RFC 2205, Sept 1997.

[5] Y. Breitbart and et al. Efficiently monitoring bandwidth and latency in IP networks. *In proceedings of IEEE INFOCOM*, Alaska, April 2001.

[6] T. Bu, N.G. Duffield, F. Lo Presti, and D. Towsley. Network tomography on general topologies. *ACM SIGMETRICS*, June 2002.

[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] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan. A framework for multiprotocol label switching. Internet Draft, 2000.

[9] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol (SNMP). IETF RFC 1157, May 1990.

[10] 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.

[11] M. J. Coates and R. Nowak. Network delay distribution inference from end-to-end unicast measurement. *in Proc. of the IEEE International Conference on Acoustics, Speech,and Signal Processing*, May 2001.

[12] M. J. Coates and R. Nowak. Network loss inference using unicast end-to-end measurement. *in Proc. ITC Conf. IP Traffic, Modeling and Management, Monterey, CA*, Sept. 2000.

[13] D. F. DeLong. Hackers said to cost U.S. billions. *E-Commerce Times Online*, Feb 8, 2001.

[14] M. Dilman and D. Raz. Efficient reactive monitoring. *In proceedings of IEEE INFOCOM*, Alaska, April 2001.

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

[16] 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.

[17] 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.

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

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

[20] J. Jiao, S. Naqvi, D. Raz, and B. Sugla. Toward efficient monitoring. *IEEE Journal on Selected Areas in Communications*, 18(5), April 2000.

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

[22] M. May, J. Bolot, A. Jean-Marie, and C. Diot. Simple performance models of differentaited services schemes for the Internet. *In proceedings of IEEE INFOCOM*, New York, March 1999.

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

[24] D. Moore, G. M. Voelker, and S. Savage. Inferring internet denial-of-service activity. *In proceedings of USENIX*, 2001.

[25] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated service field (DS field) in the IPv4 and IPv6 headers. RFC 2474, December 1998.

[26] V. Paxson. *Measurement and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California, Berkeley, Computer Science Division, 1997.

[27] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. Framework for IP performance metrics. IETF RFC 2330, May 1998.

[28] S. Savage and et al. Detour: a case for informed internet routing and transport. *IEEE Micro, v 19, no 1*, Jan 1999.

[29] S. Seshan, M. Stemm, and Katz R. H. Spand: Shared passive network performance discovery. *In proc of USENIX Symposium on Internet Technologies and Systems (USITS '97)*, Dec 1997.

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

[31] Y. Tsang, M. J. Coates, and R. Nowak. Passive unicast network tomography based on TCP monitoring. Technical report, Nov. 2000.

[32] F. Zhi, S. W. Felix, T. S. Wu, H. Huang, and F. Gong. Security issues for differentiated service framework. Internet Engineering Task Force Draft, October 1999.

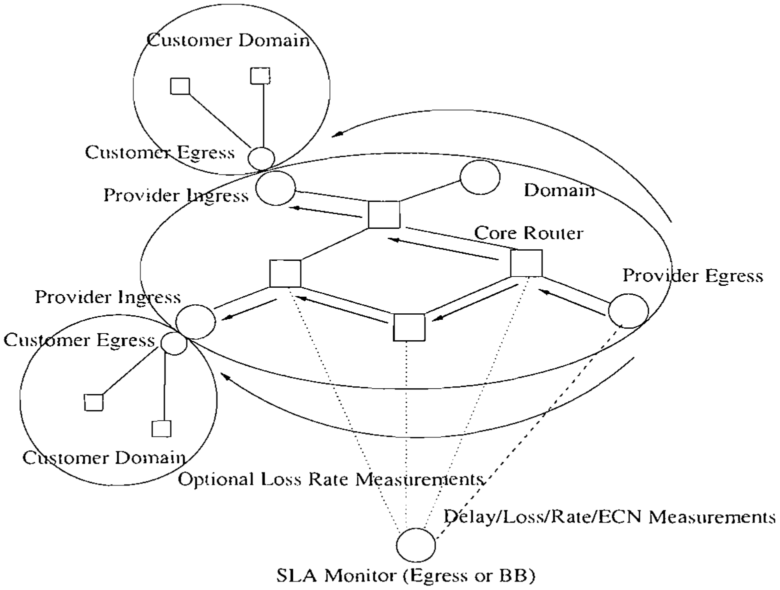


Figure 1: An architecture for detecting SLA violations and bandwidth theft attacks. Assistance from core routers is optional.

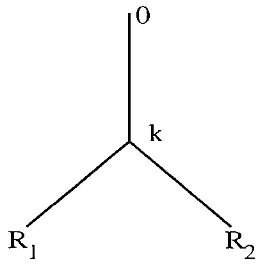


Figure 2: Binary tree to infer losses on 3 segments

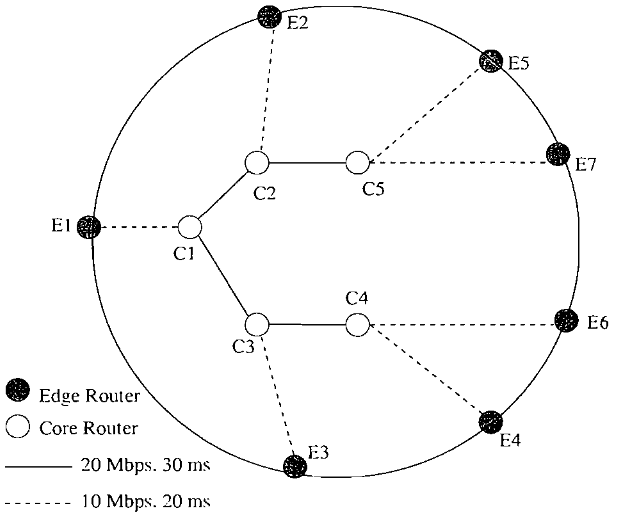
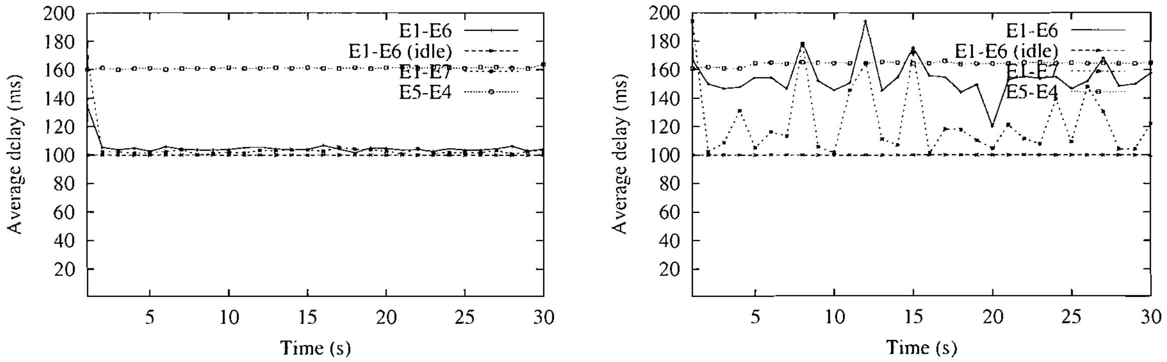


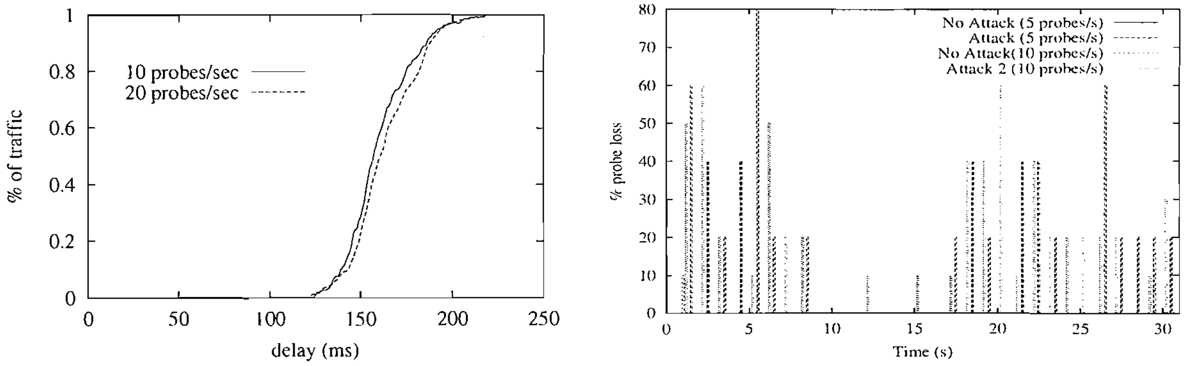
Figure 3: Topology used in simulations. All edge routers are connected to multiple hosts.



(a) Delay under light load

(b) Delay with excessive traffic

Figure 4: Link delay patterns change with increasing traffic in the network domain.



(a) Delay for different probe frequencies

(b) Probe loss

Figure 5: (a) Link delay slightly changes when more probes are introduced. (b) Probing at a low rate may be affected by probe losses when excess traffic is introduced by an attacker.

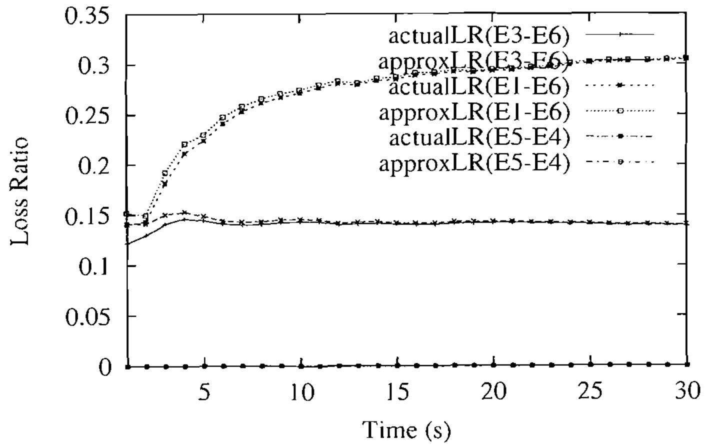


Figure 6: Loss approximation with the core-assisted scheme.

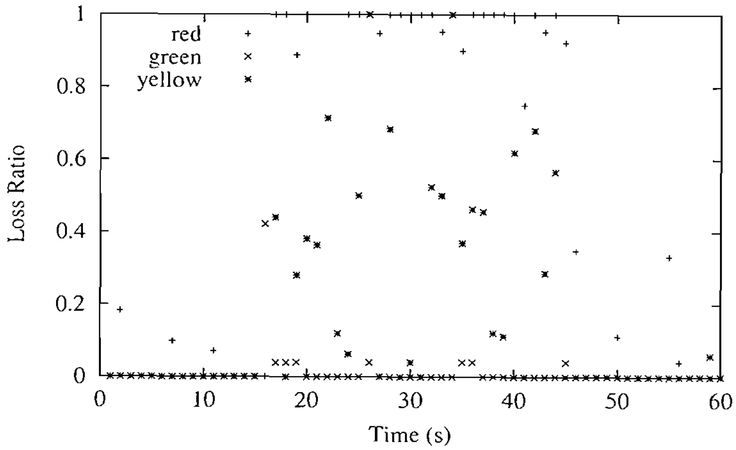
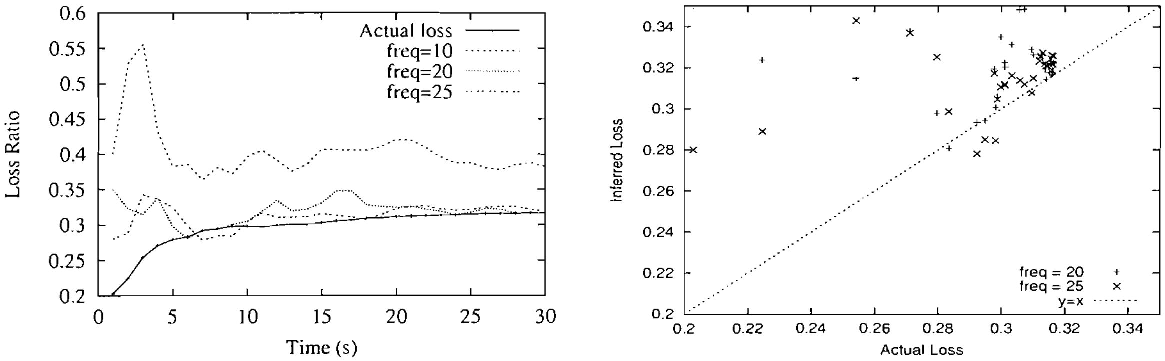


Figure 7: Loss of probe packets in the presence of high excess traffic. Green probes see high loss when a severe attack starts. Yellow and Red probes experience high drops as expected



(a) Inferred loss at various probing frequencies

(b) Scatter plot of inferred loss values

Figure 8: (a) Inferring loss of link $C4 - E6$ using striped unicast probes. "freq" denotes transmitted stripes per second. (b) Scatter plot of inferred loss values for different probing frequencies

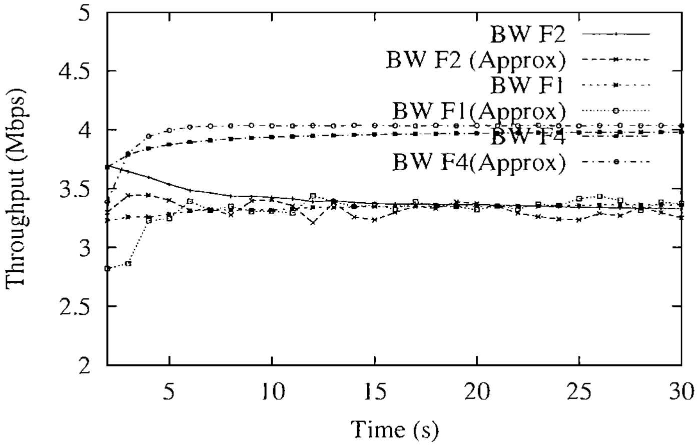


Figure 9: Throughput approximation results computed as in section 4

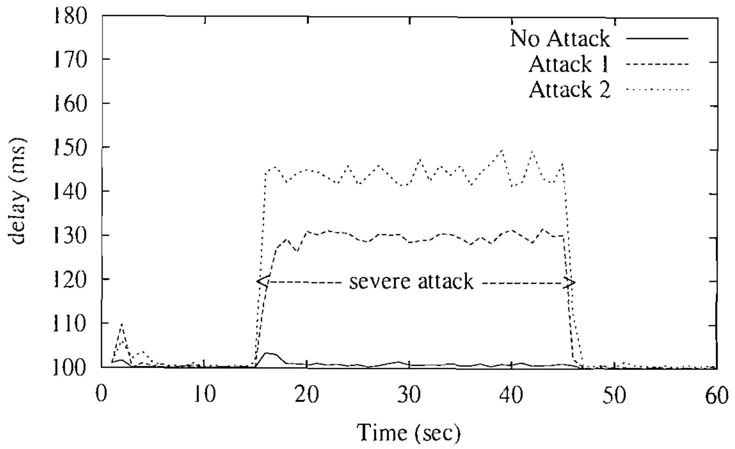


Figure 10: Observed delay at the time of an attack. "Attack 1" results in packet loss in excess of 15-25%. "Attack 2" increases packet loss to more than 35%

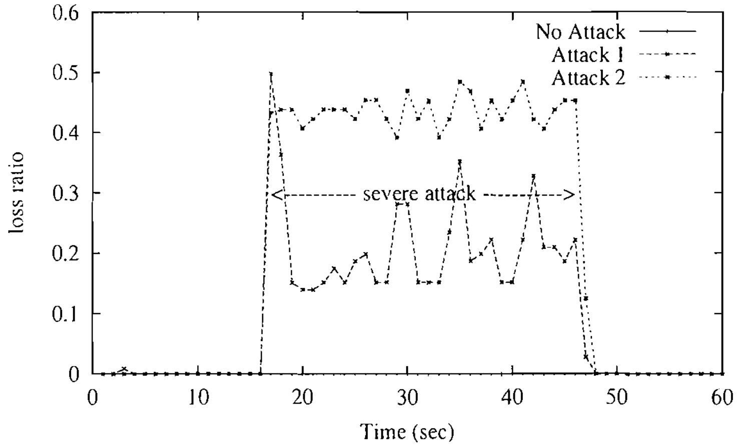


Figure 11: Overall loss follows the same pattern as delay