Edge-to-Edge Measurement-based Distributed Network Monitoring*

Ahsan Habib, Maleq Khan, and Bharat Bhargava
Center for Education and Research in Information Assurance and Security (CERIAS)
and Department of Computer Sciences
Purdue University, West Lafayette, IN 47907
{habib, mmkhan, bb}@cs.purdue.edu

Abstract

Continuous monitoring of a network domain poses several challenges. First, routers of a network domain need to be polled periodically to collect statistics about delay, loss, and bandwidth. Second, this huge amount of data has to be mined to obtain useful monitoring information. This increases the overhead for high speed core routers, and restricts the monitoring process from scaling to a large number of flows. To achieve scalability, polling and measurements that involve core routers should be avoided. We design and evaluate a distributed monitoring scheme that uses only edge-to-edge measurements, and scales well to large network domains. In our scheme, all edge routers form an overlay network with their neighboring edge routers. The network is probed intelligently from nodes in the overlay to detect congestion in both directions of a link. The proposed scheme requires significantly fewer number of probes than existing monitoring schemes. Through analytic study and a series of experiments, we show that the proposed scheme can effectively identify the congested links. The congested links are used to capture the misbehaving flows that are violating their service level agreements, or attacking the domain by injecting excessive traffic.

Keywords: Network Monitoring, Network Security, Quality of Service, Denial of Service.

1 Introduction

Continuous monitoring of a *network domain* is necessary to ensure proper operation of the network by detecting possible service violations and attacks. Attackers can impersonate a legitimate customer by spoofing flow identities. Network filtering [15] at routers can detect such spoofing if the attacker and the impersonated customer are in different domains. Otherwise, the attacks remain undetected. The quality of service (QoS) enabled networks face QoS attacks. In this setting, the attacker is a regular user of the network trying to get more resources (a better service class) than what it has signed (paid) for. A QoS network provides different classes of service for different cost, which can entice attackers to steal bandwidth and other network resources. Such attacks involve injecting traffic into the network with the intent to steal bandwidth or to cause QoS degradation, by causing other customers' flows to experience longer delays, higher loss rates, and lower throughput. Taken to an extreme, this may result in a denial of service (DoS) attack.

A large variety of network monitoring tools can be found in [17]. Many tools use SNMP [9], RMON [28], or NetFlow [11], which are built-in functionality for most routers. Using these mechanisms, a centralized or decentralized model can be built to monitor a network. The centralized approach to monitor network latency, jitter, loss, throughput, or other QoS parameters suffers from scalability. One way to achieve scalability is to use a hierarchical architecture [2, 27]. Subramanyan *et al* [27] design a SNMP-based distributed network monitoring system that organizes monitoring agents and managers in a hierarchical fashion. Both centralized or decentralized models obtain monitoring data by polling each router of a network domain, which limits the ability of a system to scale for large number of flows. The alternative way of polling is to use an event

*This research is sponsored in part by the NSF grants ANI 0219110, CCR-001712, and CCR-001788, CERIAS and IBM SUR grants

reporting mechanism that sends useful information typically in a summarized format only when the status of a monitored element changes. A more flexible way of network monitoring is by using mobile agents [20] or programmable architecture [3]. However, periodic polling or deploying agents in high speed core routers put non-trivial overhead on them. We propose a very low overhead monitoring scheme that does not involve core routers for any kind of measurements. Our assumption is that if a network domain is properly provisioned and no user is misbehaving, the flows traversing through the domain should not experience a high delay or a high loss. An excessive traffic due to attacks changes the internal characteristics of a network domain. This change of internal characteristics is a key point to monitor a network domain.

Edge-to-edge monitoring scheme is studied in [16], which devises a network monitoring mechanism to detect attacks on QoS domains using network tomography [6, 14]. This monitoring mechanism measures the service level agreement (SLA) parameters, and compares these measurements with the values negotiated between a service provider and a user. To infer SLA parameters, the tomography-based scheme constructs a tree from the network topology, and probes the leaves from the root. Probing from the root to all leaves can not infer SLA parameters in both directions of a link. This can be achieved with much higher overhead. Our goal in this work is to devise a low overhead monitoring scheme that can detect attacks in both directions of all links in a network domain.

The proposed monitoring scheme has two phases. In the first phase, we continuously measure edge-to-edge link delays to observe any unusual delay pattern. All ingress routers (entry points) sample the incoming traffic to probe delay of the paths followed by a user packet. This measures the delay experienced by a user inside the domain. If the delay is higher than a pre-defined threshold (SLA value), the edge routers conduct intelligent probing for loss measurements. For this probing, an overlay network is formed using all edge routers on top of the physical network. The probing does not calculate loss ratio for each individual link, instead, the congested links with high losses are identified using edge-to-edge loss measurements. We provide two methods to identify congested links: simple method and advanced method. In the simple method, all edge routers probe their neighbors in clockwise and counter-clockwise direction. This method requires only $O(n)$ probing, where n is the number of edge routers. Through extensive analysis, both analytical and experimental, we show that the simple method is very powerful to identify the congested links to a close approximation. We provide an advanced method that searches the topology tree intelligently for probes that can be used to refine the solution of simple method, if necessary. When the network is less than 20% congested the advanced method requires $O(n)$ probes. If the congestion is high, it requires more probes, however, does not exceed $O(n)$. In the second phase of our monitoring process, we use the congested links as a basis to identify edge routers through which traffic are entering into and exiting from the domain. From exiting edge routers, we identify the flows that are violating any SLA agreement. If the SLA is violated for delay and loss, the network is probed to detect whether any user is stealing bandwidth. The service violations can indicate a possible attack on the same domain, or on a downstream domain. In case of a DoS attack, numerous flows from different sources are destined to a victim. These flows aggregate on their way as they get closer to the victim. Monitoring an upstream network domain can detect these high bandwidth aggregates that could result in DoS attacks on downstream domains [16, 21]. To control the attacks, filters are activated at edge routers through which flows are entering into a network domain.

Using simulation, we conduct a series of experiments to evaluate the proposed monitoring scheme. We conclude that the distributed monitoring scheme shows a promise for an efficient and scalable monitoring of a domain. This scheme can detect service violations, bandwidth theft attacks, and tell when many flows are aggregating towards a downstream domain for a possible DoS attack. The scheme requires low monitoring overhead, and detects service violations in both directions of any link in a network domain.

The rest of the paper is organized as follows: The related work is discussed in Section 2. Measuring all necessary network parameters for monitoring purposes is presented in Section 3. This section discusses our proposed monitoring scheme, and analyzes its strength and limitations. Section 4 explains how to use the monitoring scheme to detect service violations and DoS attacks. Experimental results and discussions are

provided in Section 5. We conclude the paper in Section 6.

2 Related Work

One common way of monitoring is to log packets at various points throughout the network and then extract information to discover the path of any packet [24]. This scheme is useful to trace an attack long after the attack has been accomplished. The effectiveness of logging is limited by the huge storage requirements especially for high speed networks. Stone [26] suggested to create a virtual overlay network connecting all edge routers of a provider to reroute *interesting* flows through tunnels to central tracking routers. After examination, suspicious packets are dropped. This approach also requires a great amount of logging capacity.

Many proposals for network monitoring [5, 12] give designs to manage the network and ensure that the system is operating within desirable parameters. In efficient reactive monitoring [12], the authors discuss ways to monitor communication overhead in IP networks. Their main idea is to combine global polling with local event driven reporting to monitor a network. 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. They describe algorithms for computing an optimal set of probes to measure latency of paths in a network, whereas we focus on measuring parameters using distributed agents.

In [10], a histogram-based aggregation algorithm is used to detect SLA violations. The algorithm measures network characteristics on a hop-by-hop basis, uses them to compute end-to-end measurements, and validates end-to-end SLA requirements. In large networks, efficient collection of management data is a challenging task. While exhaustive data collection yields a complete picture, there is an added overhead. The authors propose an *aggregation* and *refinement* based monitoring approach. The approach assumes that the routes used by SLA flows are known, citing VPN and MPLS [8] provisioning. Though routes are known for double-ended SLAs that specify both ingress and egress points in the network, they are unknown in cases where the scope of the service is not limited to a fixed egress point. Like RON [4], we check violations using average values in a recent time frame. This reduces constraints on the network setup, and the need for knowledge of the set of flows traversing each router.

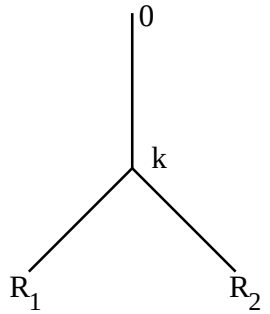


Figure 1: Binary tree to infer loss from source 0 to receivers R_1 and R_2

Duffield *et al* [13] use packet "stripes" (back-to-back probe packets) to infer link loss by computing the correlation of packet loss within a stripe at the destinations. This work is an extension of loss inference for multicast traffic, described in [1, 7]. The stripe-based probing mechanism is adopted to monitor loss characteristics inside a domain without relying on the core routers [16]. To infer loss, a series of probe packets, called a stripe, are sent from one edge router to two other edge routers with no delay between the transmissions of successive (usually three) packets. For example, in a two-leaf binary 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 one link, say $k \rightarrow R_1$ (Figure 1). 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 any receiver, we can infer that the packet must have reached the branch point k. If R_2 gets both packets of a stripe, it is likely that R_1 will receive the last packet of the stripe. The transmission probability A_k for node k can be calculated knowing how many packets are sent from the root and how many of them received by both receiver. Using A_k and number of packets reach to R_1 and R_2, we can calculate the successful transmission probability of the link $k \rightarrow R_1$. Similarly, a complementary stripe is sent to estimate the characteristics of link $k \rightarrow R_2$. By combining estimates of stripes down each such tree, the characteristics of the common path from $0 \rightarrow k$ is estimated. This inference technique extends to general trees by sending probes from root to each ordered pair of leaves [13]. Ji and Elwalid [18] show that measurement-based monitoring using tree is scalable when the probe packets reach the edge routers with high probability. If the internal loss is very high, the solution does not scale for a large network.

The unicast probing scheme is extended in [16] for routers with active queue management. This scheme is used to monitor loss inside a QoS network domain. The stripe-based monitoring scheme requires less overhead than core-based monitoring. In this paper, we propose a scalable scheme that requires much less probes than stripe-based scheme. Kim *et al* [19] collect statistical data from every single router for each service class and then analyze the data to compute edge-to-edge QoS of aggregate IP flows. This approach has very high overhead, and not suitable for real time monitoring.

3 Measurements with Distributed Probing

The service level agreement (SLA) parameters such as delay, packet loss, and throughput are measured to ensure that all users are getting their target share of resources. Delay is the edge-to-edge latency. Packet loss is the ratio of total number of packets dropped from a flow [1] to the total number of packets of the same flow entering into the domain. Throughput is the total bandwidth consumed by a flow inside a domain. Delay and loss are important parameters to monitor in a network domain. Bandwidth measurement is used to detect whether any flow is getting more than its share of resources, which causes other flows to suffer. Although jitter (a delay variation) is another important SLA parameter, it is flow-specific and, therefore, is not suitable to use in network monitoring. A large body of research has focused on measuring delay, loss, and throughput in the Internet [22, 23]. In this section, we describe techniques to measure each parameter. Delay and throughput measurements are discussed in details in [16]. This paper proposes an efficient way that detect links with high losses using edge-to-edge measurements.

The distributed monitoring scheme measures SLA components and compares the measurements to the predefined values to detect service violation. There is one monitoring agent that gets feedback from all other edge routers about delay, loss, and throughput. The monitoring agent can sit on any edge router in the network domain.

3.1 Delay Measurements

To measure delay, the ingress routers copy the IP header of incoming packet into a new packet with a certain pre-configured probability. Copying the header from user traffic to measure delay has a couple of benefits. First, the probe packet follows the same path as the user traffic because the route inside a domain does not change too often. Hence, the probe delay is similar to the delay experienced by the user. Second, if some links do not have any traffic, the links will not be probed, which saves the probing overhead.

The router encodes the current timestamp $t_{ingress}$ into the payload and marks the protocol field of the IP header with a new value. An egress router recognizes such packets and removes them from the network. Additionally, the egress router computes the edge-to-edge link delay for a packet from the difference between

[1] A flow is a micro flow with five tuples (addresses, ports, and protocol) or an aggregate flow that is combined of several micro flows.

its own time and $t_{ingress}$. We ignore minor drifts of the clocks since all routers are in one administrative domain and can be synchronized fairly accurately. The egress classifies the probe packet as belonging to flow i of user j, and update the average packet delay, $avg\_delay^i_j$, for delay sample $delay^i_j(t)$ at time t using an exponential weighted moving average (EWMA):

$$avg\_delay^i_j(t) = \alpha \times avg\_delay^i_j(t-1) + (1-\alpha) \times delay^i_j(t), \quad (1)$$

where α is a small fraction $0 \leq \alpha \leq 1$ to emphasize recent history rather than the current sample alone.

The egress router sends the average delay to the monitor. If the average packet delay of path k exceeds the delay guarantee in the SLA SLA^i_{delay} for flow i, i.e. $avg\_delay^k > SLA^i_{delay}$, it is an indication of an SLA violation. If the network is properly provisioned and flows do not misbehave, there should not be any delay greater than SLA^i_{delay} for any flow i. A high delay can be caused by some flows that are violating their SLAs or bypassing the SLA checking, which is an attack.

If the delay exceeds a certain threshold, the monitor needs to probe the network for loss and throughput. We discuss the throughput measurement first, and then discuss the loss estimation to isolate congested links. Identifying the congested links is necessary to detect egress and ingress routers involved in high traffic paths, which helps to detect and control attacks.

3.2 Throughput Measurements

The objective of checking throughput violation is to ensure that nobody is consuming any extra bandwidth (beyond the SLA). The attackers can send a lot of best effort (BE) traffic to consume bandwidth, because BE traffic is not controlled at the ingress routers. Consumption of excess bandwidth by any flow can deteriorate the QoS for many others. This can not be detected by a single ingress or egress router, if the user sends through multiple ingress routers at a rate lower than the SLA. For each ingress router, the user does not violate the SLA but as a whole he does. The service provider may allow a user to take extra bandwidth as long as everybody else is not harmed. This depends on the policy of the service provider.

The monitor measures throughput by probing globally all egress routers when the monitor suspects any violation in delay and loss. Egress routers of a QoS domain maintain the aggregate flow rate for each user. This rate is a close approximation of the bandwidth consumption by each flow inside the domain [16]. When the monitor gets throughput of all flows from egress routers, it calculates the throughput for user j, as: $B^j = \sum_{i=1}^{N} B^{ij}$, where B^{ij} is bandwidth consumed by user j at edge router i and N is the total number of edges. If SLA^j_{bw} is the bandwidth guarantee for user j, $B^j > SLA^j_{bw}$ indicates an SLA violation by user j. To detect bandwidth theft that does not change delay or loss pattern, the monitor can periodically poll egress routers.

3.3 Loss Measurements

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 drops at all core routers is an easy task. It imposes, however, an excessive overhead on the core routers by forcing them to record each drop entry, and periodically sending it to the monitor. The *stripe-based* approach is an edge-to-edge mechanism, described in Section 2, to measure loss in a domain.

We follow a different strategy. An interesting observation is that service violation can be detected without exact loss value of each internal link, instead, it requires to check whether a link has loss higher than the specified threshold or not. We propose a new approach to detect links with high losses by edge-to-edge measurements. The link with a high loss is referred to as a *congested link*. The distributed probing detects all congested links using edge-to-edge loss measurements. These links are used to detect flows that pose threats to other flows by consuming extra resources.

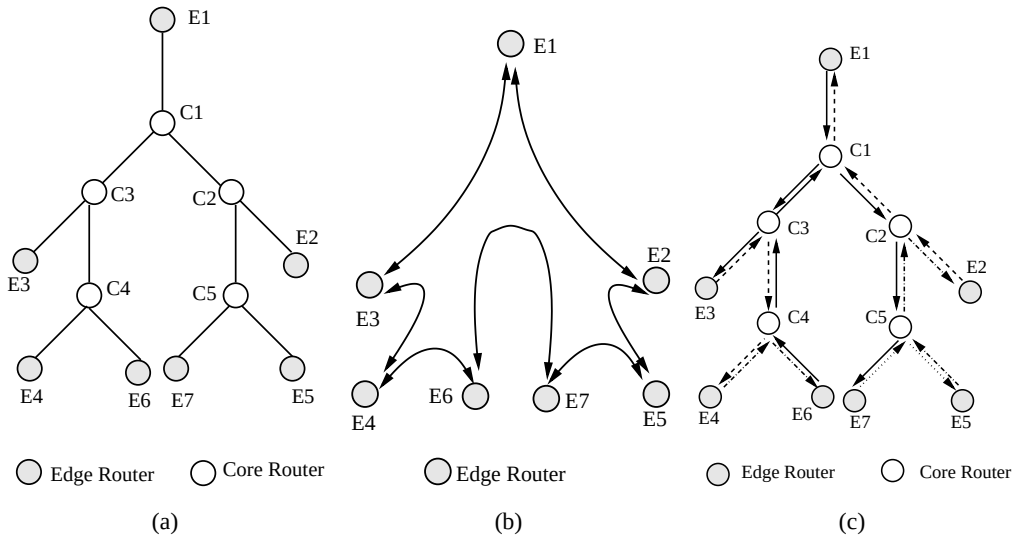


Figure 2: (a) Tree topology transformed from a network domain. (b) All probing agents at the edge routers form a virtual network with both neighbors in an ordered sequence. (c) Direction of internal links for each probing.

To apply our distributed probing, we convert the network topology into tree structure. This converting mechanism is discussed later in this section. The tree contains core routers as internal nodes and edge routers as leaf nodes. Any leaf can be used to put a monitoring agent that collects statistics from other edge routers to check SLA violations. The probing agents sit only at the edge routers and knows their neighbors. The neighbors are determined by visiting the tree using depth first search algorithm starting from any edge router, and putting all edge routers in an ordered sequence. All probing agents form a virtual network on top of the physical network. The probes follow edge-to-edge path in the virtual network. We equivalently refer the tree topology or the virtual network to an *overlay* network. A typical spanning tree of the topology, the corresponding overlay network, direction of all internal links for each probe are shown in Figure 2. The following definitions and observations are used to describe the properties of the overlay network, and to identify congested links in the proposed simple and advanced method.

Definition 1 Overlay Network. *To connect all edge routers with their neighbors in a network domain, we build a virtual network and define as an overlay network. The edge routers use this overlay for probing.*

Definition 2 Terminal core router. *A core router, which is connected to only one other core router in an overlay network is called a terminal core router. In Figure 2, the core routers $C4$ and $C5$ are the terminal core routers.*

Definition 3 Probe path. *A probe path $\mathcal{P}$ is a sequence of routers (either core or edge) $< E_1, C_1, C_2, \ldots, C_n, E_n >$ where a router appears in the sequence only once and a physical link exists between two adjacent routers. A probe packet originates at the edge router E_1, passes through the core routers $C_1, C_2, \ldots, C_{n-1}$, and C_n, in the given order, and terminates at the edge router E_n. We also represent the probe path $\mathcal{P}$ by the set of links, $\{E_1 \rightarrow C_1, C_1 \rightarrow C_2, \ldots, C_n \rightarrow E_n\}$.*

Definition 4 Link direction. *A link $u \rightarrow v$, we say link from node u to v, is in inward direction (IN) with respect to node v. Similarly, the same link is in outward (OUT) direction with respect to node u.*

Lemma 1 *If a core router C is connected to two routers (core or edge) R_1 and R_2 only, the duplex path $R_1 \leftrightarrow C \leftrightarrow R_2$ can be replaced with duplex link $R_1 \leftrightarrow R_2$, and both links are functionally equivalent in the distributed probing scheme.*

Proof: See Appendix for the proof and figure. □

Lemma 2 *In an overlay network, every core router is connected to at least three other routers.*

Proof: If a core router C is connected to two routers only, C together with its two connecting links can be replaced by a single link (by Lemma 1). If C is a terminal core router and C is not connected to any edge router, that is, C is connected to only one other router, C can never be included in a probe path and can be simply removed. Hence, an overlay network can be constructed in which all core routers are connected to at least three other routers. □

Lemma 3 *An overlay network can be constructed in such a way that every terminal core router is connected to at least two edge routers.*

Proof: Since a terminal core router t is connected to only one other core router (by Definition 1), if t is not connected to at least two edge routers, t can be removed from the network (by Lemma 1). Therefore, an overlay network can be constructed, where every terminal core router is connected to at least two edge routers. □

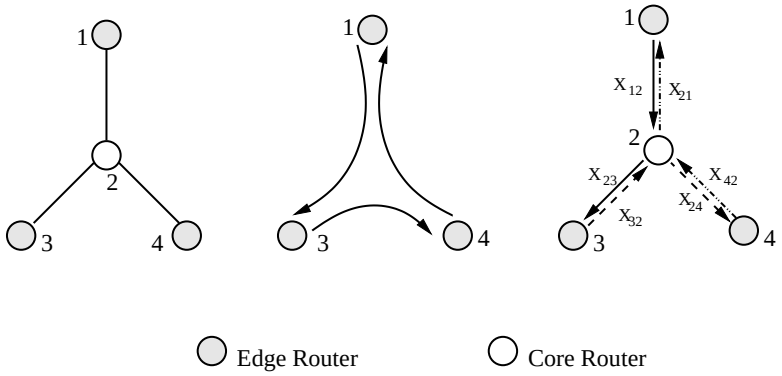


Figure 3: (a) Spanning tree of a simple network topology. (b) Each edge router probes its neighbor edge router in counter-clockwise direction (c) Direction of internal links for each probing.

3.3.1 Simple Method

In this solution, we conduct total two rounds of probing. One in the counter-clockwise direction, and another round of probing in the clock-wise direction from any edge router. The former one is referred to as *first round* of probing, and the latter one is referred to as *second round* of probing. In each round, probing is done in parallel.

We describe the loss monitoring scheme with a simple network topology. In this example, Figure 3b, edge router 1 probes the path $1 \rightarrow 3$, router 3 probes the path $3 \rightarrow 4$, and 4 probes the path $4 \rightarrow 1$. Let $P_{i,j}$ be a boolean variable that represents the outcome of a probe between edge routers i to j. $P_{i,j}$ takes on value 1 if the measured loss exceeds the threshold in any link within the probe path, and 0 otherwise. Notice that $P_{i,j} = 0$ for $i = j$. We express the outcome of a probe in terms of combination of all link status. Let $X_{i,j}$ be a boolean variable to represent the congestion status of an internal link $i \rightarrow j$. We refer X to a *congestion variable* in this paper. For Figure 3c, we can write equations as follows:

$$X_{1,2} + X_{2,3} = P_{1,3} \quad X_{3,2} + X_{2,4} = P_{3,4} \quad X_{4,2} + X_{2,1} = P_{4,1}, \tag{2}$$

where (+) represents a boolean "OR" operation. We express status of internal links of any probe path of a network topology in terms of probe outcomes. The equation with all internal links on probe path $\mathcal{P}_{i,j}$ and its outcome $P_{i,j}$ for an arbitrary topology is shown below.

$$X_{i,k} + \sum_{n=k}^{n=l-1} X_{n,n+1} + X_{l,j} = P_{i,j}. \tag{3}$$

Note that loss in path $1 \rightarrow 3$ might not be same as loss in path $3 \rightarrow 1$. This path asymmetry phenomenon is shown in [25]. In general, $X_{i,j}$ is independent of $X_{j,i}, \forall_{ij}, i \neq j$.

The second round of probing, Figure 3(a), is done from $1 \rightarrow 4$, $4 \rightarrow 3$, and $3 \rightarrow 1$. We express the outcome of this round of probing in terms of internal links as follows:

$$X_{1,2} + X_{2,4} = P_{1,4} \quad X_{4,2} + X_{2,3} = P_{4,3} \quad X_{3,2} + X_{2,1} = P_{3,1} \tag{4}$$

The sets of equations (2 and 4) are used to detect congested link in the network. For example, if the outcome of the probing shows $P_{1,3} = 1$, $P_{1,4} = 1$, and rest are 0, we get the following:

$$X_{1,2} + X_{2,3} = 1 \qquad X_{1,2} + X_{2,4} = 1 \tag{5}$$

All other probes do not see congestion on its path, i.e., $X_{3,2} = X_{2,4} = X_{4,2} = X_{2,1} = X_{2,3} = 0$. Thus, the equation set (5) reduces to $X_{1,2} = 1$. Similarly, if any of the single link is congested, we can isolate the congested link. Suppose, two of the links, $X_{1,2}$ and $X_{2,3}$, are congested. The outcome of probing will be $P_{1,3} = 1$, $P_{1,4} = 1$, and $P_{4,3} = 1$, which makes $X_{3,2} = X_{2,4} = X_{4,2} = X_{2,1} = 0$. This leaves the solution as shown in equation(6). Thus, the distributed scheme can isolate links with high loss in this topology.

$$X_{1,2} + X_{2,3} = 1 \qquad X_{1,2} = 1 \qquad X_{2,3} = 1 \tag{6}$$

Analysis of Simple Method. The strength of simple method comes from the fact that congestion variables in one equation of any round of probing is distributed over several equations in the other round of probing. If n variables appear in one equation in the first round of probing, no two (out of the n) variables appear in same equation in second round of probing (Lemma 4) or vice versa. This property helps to solve the equation sets because congestion in any probe path of any round is totally spread over several equations in other round. Theorem 1 shows that if any single probe path is congested with arbitrary number of links, the simple method can identify all the congested links. In Theorem 2, we show that the simple method determines the status of a link with very high probability when the network is less congested.

Lemma 4 *If $\mathcal{P}$ and $\mathcal{P}'$ are probe paths in the first and the second round of probing respectively, $|\mathcal{P} \cap \mathcal{P}'| \leq 1$.*

Proof: See Appexdix A. □

Theorem 1 *If only one probe path $\mathcal{P}$ is shown to be congested in the first round of probing, the simple method successfully identifies each congestion link in $\mathcal{P}$.*

Proof: Let, the congested probe path be $\mathcal{P} = \{l_1, l_2, \ldots, l_k\}$ and X_i is the congestion variable for link $l_i, 1 \leq i \leq k$. X_i appears once in equations for each round of probing. Let, X_m is in equation $X_m + f(S) = 1$ in the second round of probing, where S is a set of congestion variables excluding X_i that appear in the equation. The expression $f(S)$ does not contain any of the variables X_i for $1 \leq i \leq k, i \neq m$ (Lemma 4). From first round of probing, we obtain $f(S) = 0$, because the outcome of all probe paths except $\mathcal{P}$ is zero in this round. Thus, we can determine X_i, which is 1, hence the status of the link l_i, for any $1 \leq i \leq k$. □

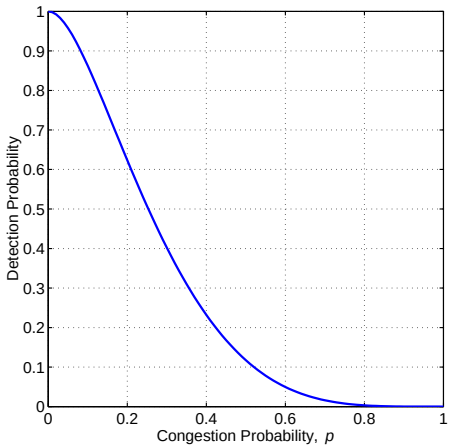


Figure 4: Probability that the simple method determines the status of a link of any arbitrary topology with only $2n$ probes, where n is the number of edge router in the topology. The simple method performs extremely well when less than 20% links of a network are congested. If a network is more than 50% congested, the simple method can not contribute much.

Theorem 2 *Let p be the probability of a link being congested in any arbitrary overlay network. The simple method determines the status of any link of the topology with probability at least* $2(1-p)^4-(1-p)^7+p(1-p)^{12}$.

Proof: Let a particular link l appears in probe paths $\mathcal{P}_1$ and $\mathcal{P}_2$ in first and second round of probing. The status of a link can be either non-congested or congested. We consider both cases separately and then combine the results.

When l is non-congested. The status of l can be determined if the rest of the links in either p_1 or p_2 are non congested. There are also some other cases where the status of l can be determined. Let the length of probe paths $\mathcal{P}_1$ and $\mathcal{P}_2$ are i and k respectively. Since, only common link between paths $\mathcal{P}_1$ and $\mathcal{P}_2$ is l (Lemma 4), the following two events are independent: $Event_1$=all other links in p_1 are non-congested and $Event_2$=all other links in p_2 are non-congested. Thus, for a non-congested link,

$$\begin{aligned} Pr\{\text{status of } l \text{ be determined}\} &\geq Pr\{Event_1\} + Pr\{Event_2\} - Pr\{Event_1\}Pr\{Event_2\} \\ &= (1-p)^{i-1} + (1-p)^{k-1} - (1-p)^{i-1}(1-p)^{k-1} \\ &= (1-p)^{i-1} + (1-p)^{k-1} - (1-p)^{i+k-2} \end{aligned}$$

Now, we estimate the average value of i and k. In our overlay network, the number of links are $2(e+c-1)$ considering both directions of a link. The edge routers are leaves of the topology tree whereas the core routers are the internal nodes of the tree. Number of leaf nodes is always greater than the number of internal nodes. Thus, the number of links is $\leq 2(e+e-1) = 4e$. Number of probe paths in any round (first or second) of probing is e and every link appears exactly once in each round. So, the average length of a path $\leq \frac{4e}{e} = 4$.

Using the average length for the probe paths, i.e., $i = k = 4$,

$Pr\{\text{Status of } l \text{ be determined}\} \geq 2(1-p)^3 - (1-p)^6$.

When l is congested. If l is a congested link, its status can be determined when all other links that appear on the probe path of l are determined to be non-congested. As, the average path length is 4 in the simple method, the probability that a path has all non-congested links is $(1-p)^4$. To determine the status of all 3 links that appear with l in a equation, all 3 of them have to non-congested and determined. Thus, $Pr\{\text{Status of } l \text{ be determined}\} = (1-p)^{12}$.

For any link l (congested or non-congested)

$Pr\{\text{Status of } l \text{ be determined}\} \geq (1-p)[2(1-p)^3 - (1-p)^6] + p[(1-p)^{12}]$

$$= 2(1-p)^4 - (1-p)^7 + p(1-p)^{12}. \qquad \square$$

Figure 4 shows the probability to determine status of a link with probability that a network is congested. This figure shows that simple method determines status of a link with probability more than 0.60 when 20% links of a network are congested. For 30% congestion, the probability is higher than 0.40. The simple method does not help much when 50% links of a network is congested. In that case, we use the advanced method, which is described in this section. This result is validated with the simulation result for two different topologies.

Having congestion on links that affect multiple probe paths might eventually lead to some boolean equations that do not have unique solutions. Thus, the solution of simple method can have congested links and some undecided links. In that case, we can either use a special set of probes that were not used before (discussed in Section 3.3.2), or apply *stripe-based* probing to a part of the tree to determine the exact locations of the congested links. Alternatively, we can even report all the links from the unsolved equations at the end of simple method as congested. These links are referred to as false positive because some non-congested links are reported as congested. The false positive is calculated as a ratio of undecided links labeled as congested by simple method to the total number links in the network. Figure 5 shows false positive for two topologies; Topology 1 shown in Figure 2(b) and Topology 2 shown in Figure 9(b). This figure does not compare the two topologies, instead, it shows the false positive as a percentage of total links with respect to percentage of links that are actually congested. The false positive is a small percentage of overall links of a domain. The number of links that are marked as false positive is very close to the number of actually congested links. The reason we get false positive is that some good (non-congested) links sit on the same probes of congested links and the simple method does not have enough probes to isolate them. Notice that the solution does not have any false negative. We describe the advanced method to find probes that can decide the status of undecided links from simple method.

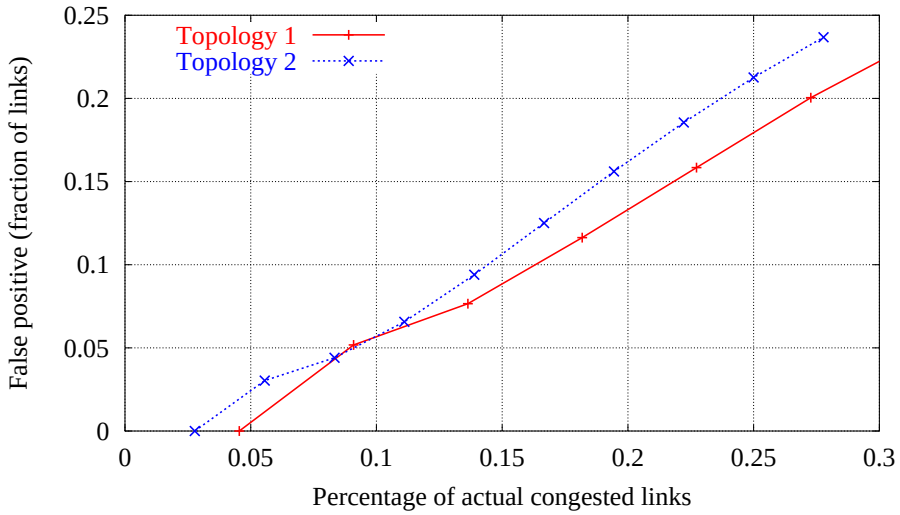


Figure 5: The solution of the simple method can not decide about some links. If those links are considered as congested links, the solution of the simple method provides false positive by declaring some links as congested. The graph is shown for two topologies; Topology 1 shown in Figure 2(b) and Topology 2 shown in Figure 9(b). This figure does not compare the two topologies, instead, it shows the false positive as a percentage of total links with respect to percentage of links that are really congested. The solution does not have any false negative.

We further analyze the simple method when a network has congestion that spreads from one edge router to any other edge routers. In real network, numerous fbws come from different edge routers and make a series of links to be congested. In this case, the simple method performs very well. We observe that for edge-to-edge

congested paths, the simple method does not add any link as false positive. We plot this behavior for both topologies with all possible edge-to-edge paths in Figure 6. On the average, the simple method can isolate more than 50% of the congested links for edge-to-edge congestion scenario. Rest of the cases, the solutions have some equations with more than one variable. We can apply advanced method to be sure about the status of these links. The percentage of identified links is little high for the path length=6 in case of Topology 1, Figure 6. Because this path has more shared links comparing to other paths.

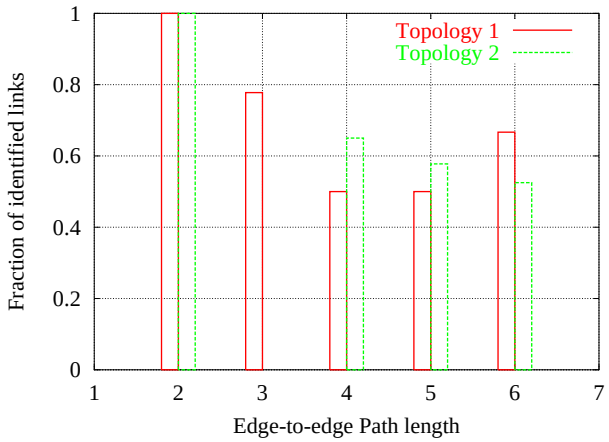


Figure 6: Fraction of identified links by the simple method for all edge-to-edge congested paths in the network. The X-axis shows all paths with a specific length. All solutions for edge-to-edge congestion path does not have any false positive. Topology 2 does not have any path of length 3.

3.3.2 Advanced Method

The solution of the simple method might have some false positive links. As the percentage of false positive links is small comparing to the overall links of the topology, we can proceed to the detection phase with the solution of simple method considering the undecided links as congested. Alternatively, we can refine the solution of the simple method to discard the false positive links. Then the output of the simple method is used as the input of the advanced method. Let the set of equations with undecided variables be $\mathbb{E}$. Now, we traverse the topology tree to find probes that can help to decide about the values for each undecided variable.

The algorithm of the advanced method is shown in Figure 7. For each variable in equation set $\mathbb{E}$, we need to probe the network that helps to decide the variable. Each probe needs one start node and one end node. The algorithm shows functions to find start and end probe nodes. Link direction plays an important role to find these probes. The link direction is defined in Definition 4. For example, in Figure 2, if link $C1 \rightarrow C3$ is congested, the start probe node can be $E2$, $E5$, or $E7$. On the other hand, if link $C3 \rightarrow C1$ is congested, the start probing node can be $E3$, $E4$, or $E5$.

For an undecided link $v_i \rightarrow v_j$, the function *FindNode* looks for leaves descended from node v_i and v_j. First, the algorithm searches for a node in IN on a subtree descended from v_i and then in OUT direction on a subtree descended from v_j. For any node v, the *DecidePath* explores all siblings of v to choose a path in a specified direction. The function avoids previously visited path and known congested path. It marks already visited path so that same path will not be repeated in exploration of an alternate path.

If the network is congested in a way that no solution is possible, the *AdvancedMethod* can not add anything

```
AdvancedMethod()
begin
  Conduct probing from root in the counterclockwise direction. Outcome is an unsolved equation set 𝔼.
  for Each undecided variable X_ij of 𝔼 do
    node1 = FindNode(Tree T, v_i, IN) /*See Definition 4 for description of IN and OUT direction.*/
    node2 = FindNode(Tree T, v_j, OUT)
    if node1 ≠ NULL AND Node2 ≠ NULL then
      Probe(Node1, Node2). Update equation set 𝔼.
    end if
    Stop if no more probe exists
  end for
end

FindNode(Tree T, Node v_i, dir)
begin
  if v_i is leaf then
    return v_i
  end if
  v_k = DecidePath(v_i)
  if v_k = NULL then
    return NULL
  else
    node = FindEndNode(T, v_k, dir)
  end if
end

DecidePath(Node v_i, integer dir)
begin
  𝕍 ← siblings(v_i)
  for Each v of 𝕍 do
    if (dir=IN AND good(v → v_i)) OR (dir=OUT AND good(v_i → v)) then
      return v /*good(L) ⇔ L is neither congested nor visited.*/
    end if
  end for
  return NULL
end
```

Figure 7: Advanced method to obtain probes that help to decide about the status of a congestion variable.

to the simple method. If there is a solution, the *AdvancedMethod* can obtain probes to decide about links because this probe finding is an exhaustive search on the topology tree to find leaf-to-leafs path that are not already congested.

Analysis of Advanced Method. The number of probes required in advanced method depends on the number of congested links existing in a network. The advanced method starts with the undecided links. When the network is sparely congested or densely congested, the algorithm exits with fewer run and the number of trial for each congestion variable is low. To obtain how many trials we need to identify the status of each link, we need the average length of a probe path d and on how many paths b a link lies on. For an arbitrary overlay network, we calculate the approximated value of d and b in Lemma 6 and Lemma 5 respectively. Using these two values we show that, Theorem 3, the advanced method identifies the status of a link in $O(n)$ probing with a very high probability when the network is 20% congested or less.

Lemma 5 *For an arbitrary overlay network with e edge routers, on the average a link lies on $\frac{e(3e-2)}{8 \ln e}$ edge-to-*

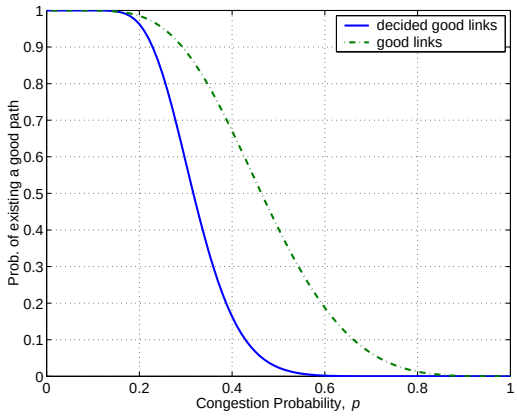


Figure 8: Probability that the advanced method determines the status of a link of topology shown in Figure 2a. The X-axis is the probability that a link to be congested. The Y-axis is the probability that a good path (non-congested) exists for any link. The dotted graph shows the probability that a good path exists. The solid graph shows the probability that a good and decided path (from the first round) exists.

edge paths.

Proof: See Appendix B. □

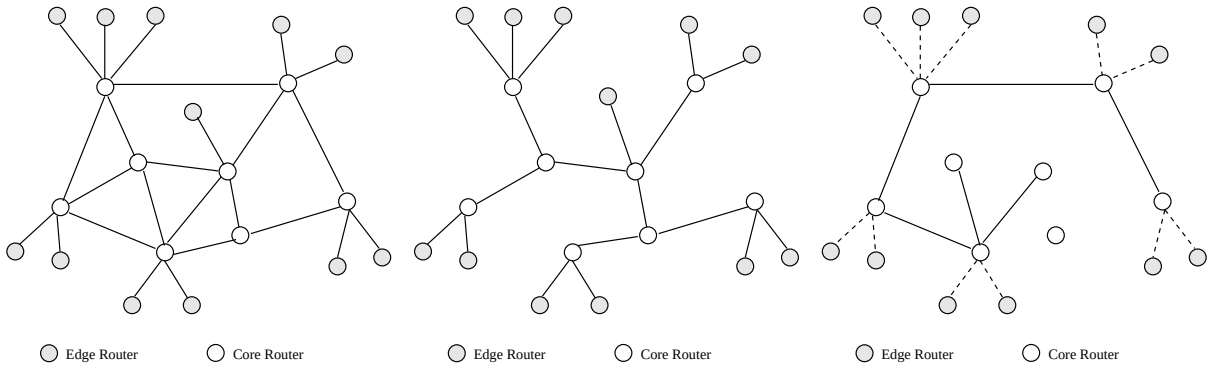
Lemma 6 *For an arbitrary overlay network with e edge routers, the average length of all edge-to-edge paths is $\frac{3e}{2\ln e}$.*

Proof: See Appendix B. □

Theorem 3 *Let p be the probability of a link being congested. The advanced method can detect the status of a link with probability at least $1-(1-(1-p)^d)^b$, where $d=\frac{3e}{2\ln e}$ is the average path length and $b=\frac{e(3e-2)}{8\ln e}$ is the average number of paths a link lies on.*

Proof: The probability that a path of length d is non-congested $(1-p)^d$. The probability of having all b paths congested is $(1-(1-p)^d)^b$. Thus, the probability that at least one non-congested path exists is $1-(1-(1-p)^d)^b$. □

The performance of the advanced method is plot in Figure 8 for Topology 1. This figure shows the probability that a good (non-congested) path exists for any link. Two graphs are shown: one shows the probability that a good path exists and the other shows the probability that a good as well as decided path exists. The advanced method needs only *one probe* on the average to identify the status of the link when the network is less than 20% congested. In this case, the total required probes is $O(n)$. Some links might need more than one, which is not high because a good and decided path exists. If the network is more than 50% congested, the advanced method can not find a good path easily because the path does not exist, and the advanced method terminates quickly. When the network is highly congested, we need to check almost all the fbws. So, we can go to the detection phase instead of wasting time to rule out very few good links.



(a) Original Topology (b) Converted tree topology (c) Rest of the topology

Figure 9: Preprocessing of a general tree topology to apply distributed probing. The original topology is split into tree topologies. Then, the results are aggregated to get overall picture of a network.

3.4 General Network Topology

The simple and the advanced methods are applicable to a network topology with a tree structure only. If there is any loop in the topology or multiple paths from one edge router to another edge router, we need to preprocess the topology before applying the algorithm. First, we split the topology into a spanning tree, and a set of subtrees that may be connected or not. The algorithm is applied to all trees to identify the congested links. We might have multiple probe paths from one edge router to another. In this case, we apply source routing for probe packets to follow the specified route. Some of the subtrees may not be connected to edge routers, i.e., some parts of subtrees may consist of only core routers. To probe those links, we need to connect them to edge routers. We should be careful to connect these internal links with non-congested links. When all subtrees are probed, we need to combine them. As probing any path does not affect other paths, applying our scheme on split tree will not affect each other. We obtain the union of all congested links from each topology as a final set of congested links for the whole topology.

In Figure 9, the general topology (Figure 9a) is split into two trees. The first one (Figure 9b) is a spanning tree for the general topology. The other one (Figure 9c) is a tree where two core routers are not connected to any edge router. We need to add links to these core routers so that we can access this link from edge routers. When probing on Figure 9b is done we select some good links to connect these core routers with the edge routers. At the end, all results can be combined together to reflect the overall status of the topology. The topology preprocessing is done infrequently only when a network is setup, and when any link or router is added.

3.5 Limitations of Distributed Monitoring

There are some limitations for the distributed monitoring approach. For example, in Figure 3a, if both $X_{2,3}$ and $X_{2,4}$ are congested, we can not decide about $X_{2,3}$. Because we need at least one non-congested outgoing link from core router 2 to decide about the link $X_{1,2}$. The argument is the same for $X_{2,1}$ when both $X_{3,2}$ and $X_{4,2}$ are congested. If all links have the same bandwidth, we can report all three links as congested. Even if $X_{1,2}$ ($X_{2,1}$) has the combined capacity of the two outgoing (incoming) links, the argument is still valid. In such case,

the algorithm will report one non-congested link as congested, which is a close approximation of actual result. Each terminal core has at least two edge routers (by Lemma 3). As long as any non-congested $core \rightarrow edge$ link exists, our method can provide partial solution.

The worst case is when all links from edge routers to core routers are congested in a network domain. In this case, outcome of all probes will be congested. The solution of the simple and advanced method is *all links are congested*. This solution is useful because it is very likely that the whole network is congested when all $edge \rightarrow core$ links are congested. Thus, we can also go to the detection phase considering the whole network is congested. This is also true when all $core \rightarrow edge$ links are congested. If some combinations of $E \rightarrow C$ or $C \rightarrow E$ are not congested, we can use them to provide a partial solution for the network.

4 Detecting Violations and Attacks

4.1 Violation Detection

Violation detection is the second phase of our monitoring process. When delay, loss, and bandwidth consumption exceed the pre-defined thresholds, the monitor decides whether the network experiences a possible SLA violation. The monitor knows the existing traffic classes and the acceptable SLA parameters per class. For each service class, we obtain bounds on each SLA parameter that is be used as a threshold. A high delay is an indication of abnormal behavior inside a network domain. If there is any loss for the guaranteed traffic class, and if the loss ratios for other traffic classes exceed certain levels, an SLA violation is flagged. This loss can be caused by some flows consuming bandwidths above their SLA_{bw}. Bandwidth theft is checked by comparing the total bandwidth obtained by a user against the user's SLA_{bw}. The misbehaving flows are controlled at the ingress routers.

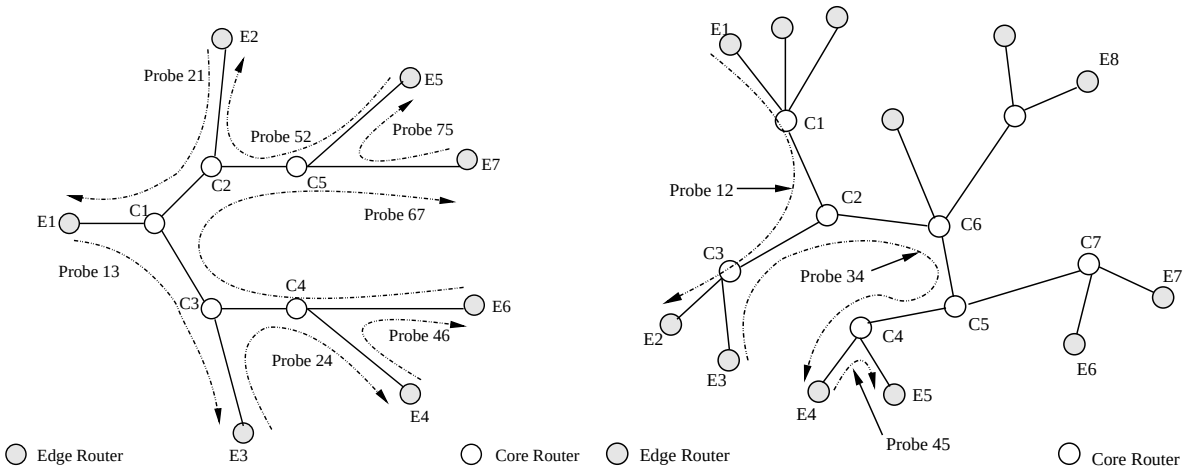
4.2 Detecting DoS Attacks

To detect DoS attacks, set of links L with high loss are identified. For each congested link, $l(v_i, v_j) \in L$, the tree is divided into two subtrees: one formed by leaves descendant from v_i and the other from the leaves descendant from v_j. The former subtree has egress routers as leaves through which high aggregate bandwidth flows are leaving. If many exiting flows have the same destination IP prefix, either this is a DoS attack or they are going to a popular site [21]. Decision is taken by consulting the destination entity. In case of an attack, we control it by triggering filters at the ingress routers, which are leaves of the subtree descendant from v_i and feeding flows to the congested link. For each violation, the monitor takes action such as throttling a particular user's traffic using a flow control mechanism.

A scenario of detecting and controlling DoS attack is now illustrated using Figure 10a. Suppose, the victim's domain $\mathcal{D}$ is connected to the edge router $E6$. The monitor observes that links $C3 \rightarrow C4$ and link $C4 \rightarrow E6$ are congested for a specified time duration Δt sec. From both congested links, we obtain the egress router $E6$ through which most of these flows are leaving. The destination IP prefix matching at $E6$ reveals that an excess amount of traffic is heading towards the domain $\mathcal{D}$ connected to $E6$. To control the attack, the monitor needs to identify the ingress routers through which the suspected flows are entering into the domain. The algorithm to identify these ingress routers is discussed in next subsection.

4.3 Flow Aggregation and Filtering

An important question is how to identify ingress routers through which the fbws are entering into the domain. To identify the fbw aggregation, we use delay probes and assign an ID to each router. An ingress router puts its ID on the delay probe packet. The egress router knows from which ingress routers the packets are coming. For example, in Figure 10a, say egress router $E6$ is receiving fbws from $E1$, $E2$, $E3$, and $E5$. These fbws aggregate during their trip to $E6$, and makes the link $C4 \rightarrow E6$ congested. We traverse the path backwards from the egress router to the ingress routers through the congested link to obtain the entry points of the fbws that are causing attacks. In this example, all edge routers can feed the congested links and they all will be candidates for activating fi lters. Knowing the ingress routers and congested links, we fi gure out the entering routers for the fbws that are causing the attacks.



(a) Topology 1 (b) Topology 2

Figure 10: Topology used to detect service violations using distributed probing. All edge routers are connected to one or multiple domains. All core to core router links are 20 Mbps with 30 ms delay and core to edge router links are 10 Mbps with 20 ms delay. The probes are named with the subscripts of the edge routers.

5 Simulation Results

The performance of our monitoring mechanism is evaluated using simulation. Attacks are simulated to show that the edge routers can detect service violations and attacks due to fbw aggregation towards a downstream domain. First, we provide experiments on parameter measurements to show the algorithms described in Section 3 work properly. Then, we conduct experiments on detecting service violations and attacks.

5.1 Measuring Parameters and Monitoring

We use a network topology shown in Figure 10a, which is similar to the one used in [13, 16] to evaluate stripe-based loss ratio approximations. We want to compare our distributed monitoring with the stripe-based monitoring scheme. Figure 10b is a more complex topology, which is used to show what happen when multiple

attacks happen simultaneously and one changes the behavior of the others. Multiple domains (not shown in the Figure 10) are connected to the edge routers for both topologies to create fbws along all links in the domain. In Topology 1, fbws coming through $E1$, $E2$, $E3$ are destined to edge router $E6$ to make the link $C4 \rightarrow E6$ congested. Many other fbws are created to ensure that all links carry a signifi cant number of fbws.

Interested readers are referred to [16] for detail analysis of our delay and throughput measurement techniques. In this paper, we show how delay pattern changes with excessive traffi c in a domain. We measure delay when the network is properly provisioned or over-provisioned (and thus experiences little loss). When idle, the edge-to-edge delay of $E1 \rightarrow E6$ link is 100 ms; $E1 \rightarrow E7$ delay is 100 ms; and $E5 \rightarrow E4$ delay is 160 ms. When there is an attack, the average delay of the $E1 \rightarrow E6$ link is increased to as high as 180 ms. Figure 11 shows how the delay increases in presence of attacks that inject extra traffi c into the domain. When there is no attack, the edge-to-edge delay is close to the link transmission delay. If the network path $E1 \rightarrow E6$ is lightly loaded, for example with a 30% load, the delay does not go signifi cantly higher than the link transmission delay. Even when the path is 60% loaded (medium load in Figure 11), the edge-to-edge delay of link $E1 \rightarrow E6$ increases by less than 30%. Some instantaneous values of delay go as high as 50% of the link transmission delay but the EWMA does not fluctuate a lot. In this example, the network is properly provisioned, i.e., the fbws do not violate the SLAs. In contrast, an excess traffi c introduced by an attacker increases the edge-to-edge delay inside a network domain. In case of attack, most of the packets of attack traffi c experience a delay 40-70% higher (Figure 11) than the link delay. Delay measurement is thus a good indication of the presence of excess traffi c inside a network domain.

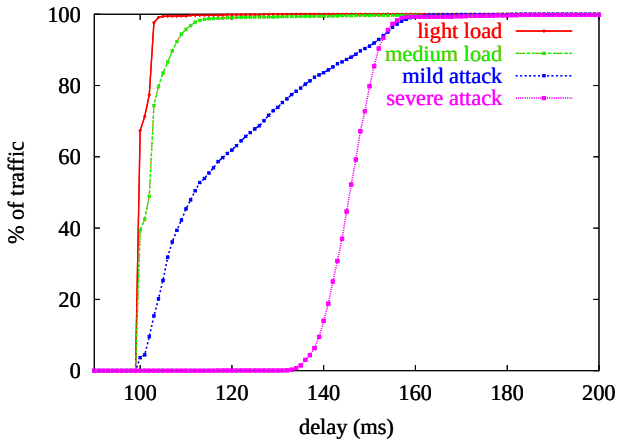
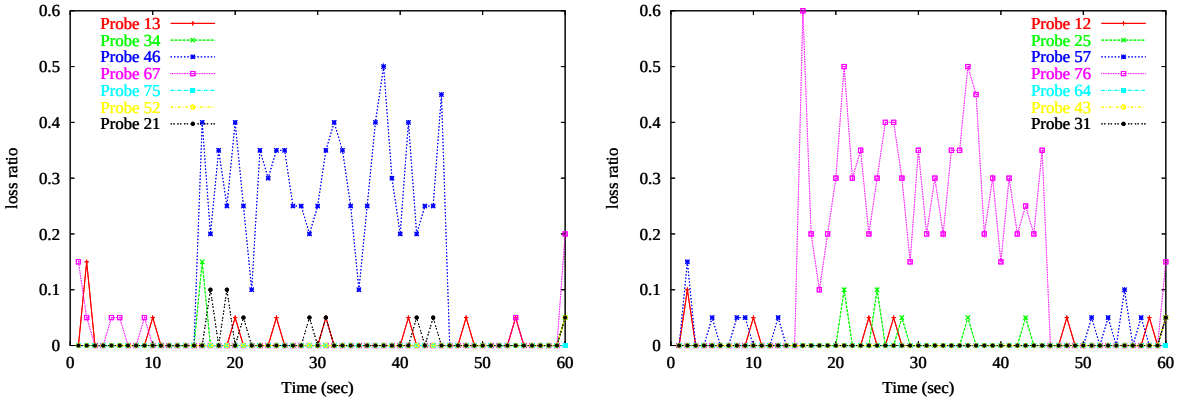


Figure 11: Cumulative distribution function (CDF) of edge-to-edge link delay for link $E1 \rightarrow E6$. The delay changes with network traffi c load.

Now, we demonstrate how the distributed probing to detect congested links in a network domain. Some of the hosts that are connected to domains attached with the edge routers violate SLAs. The inject more traffi c through multiple ingress routers to conduct an attack on link $C4 \rightarrow E6$. The intensity of the attack is increased during the interval from *t=15* seconds to *t=45* seconds. The attack causes around 35% of packet drops except an initial jump at 15 sec.

To identify the congested links, the edge routers probe to their neighbors. Figure 12 shows that *Probe 46*



(a) Counterclockwise probing.

(b) Clockwise probing.

Figure 12: Probe outcome both for counterclockwise and clockwise direction. Probe 46 in (a) and Probe 57 in (b) have high losses, which means that link $C4 \rightarrow E6$ is congested.

in counterclockwise direction and *probe 57* in clockwise direction experience high losses. Other probes do not face high losses, that is, most of the internal links are not congested. It is important to note that *Probe 46* experiences high loss but *Probe 64*, which is in the opposite direction to *Probe 46*, faces very small amount of loss. This verifies that link loss in both directions of a link can be very different, based on the traffic load on each direction. Using algorithm specified in Section 3, we detect that link $C4 \rightarrow E6$ is the only congested link in the domain.

All points in the Figure 12 are calculated by taking averages of samples over one second time period. If we take the average over a longer time period, we can avoid this high fluctuations of loss. Figure 13 shows that taking averages over a longer time period reduces the chance of considering a non-congested link as congested. We observe that taking averages over a longer time period helps more in reducing the fluctuations than in increasing the number of probes per second. The actual loss for this congested link is very high (Figure 14), which verifies that the distributed probing is able to detect links with high losses.

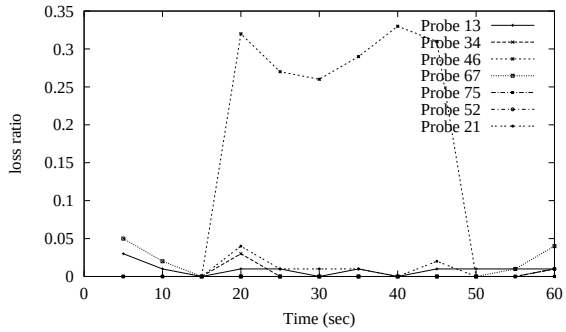


Figure 13: Probe outcome using 5-second averages for the same experiments shown in Figure 12a.

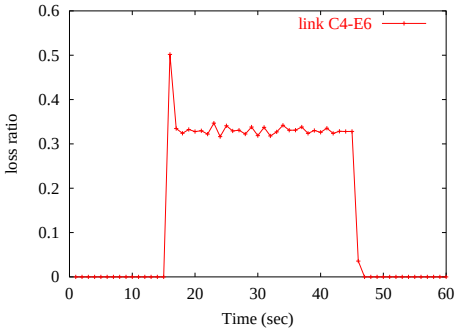


Figure 14: Actual loss in link $C4 \rightarrow E6$. Other links have low losses. This verifies that our monitoring scheme detects the congestion properly.

5.2 Local Vs. Global Congestion

In this section, we address the question what happens if the congestion status is changed during our probing. To show an example, we use the Topology 2 (Figure 10b). This topology is more complex and we simulate congestion in such a way that congestion in one area might affect the congestion of another area. Two attacks are simulated in this case. The first attack (Attack 1) is due to excessive flows coming from different edge routers and makes the $C4 \rightarrow E5$ link congested. All of the probes in the first round are good except "Probe 45". This attack continues up to time $T = 50$ sec (Figure 15) At time 50 sec, we have another attack (Attack 2), which is more severe than Attack 1. This attack causes several links on "Probe 34" path congested. It is interesting to note that Attack 2 actually causes Attack 1 to be disappeared from the scenario. Because most of the traffic causes attack on link $C4 \rightarrow E5$ are now dropped earlier in their path due to Attack 1 (Figure 15).

This experiment shows that a local congestion might disappear due a global and severe congestion. The main objective of our work is to pin point a congestion. However, if the congestion is changed while an experiment is being conducted it catches the latest congestion. The simple method can complete two rounds of probing within $10 - 20$ sec. If both rounds of probing are done in parallel, it takes only 10 sec. If a congestion does last for 20 sec, we believe that no action is necessary to alleviate it.

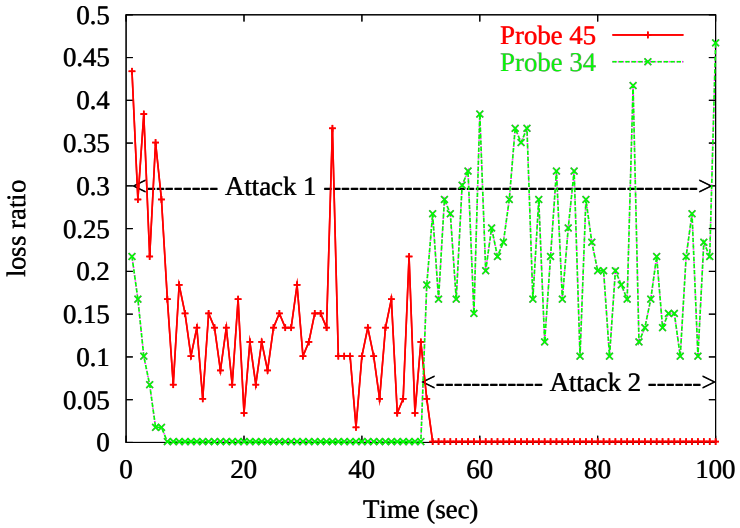


Figure 15: Attack 1 causes link $C4 \rightarrow E5$ congested. However, Attack 2 comes from all different edge routers to E4, which causes the traffic of Attack 1 to drop early. As a result Probe 45 is not congested after 50 sec.

5.3 Detecting Attacks

A major advantage of using the SLA monitor is that it is able to detect denial of service (DoS) and Distributed DoS (DDoS) attacks in a network domain. When the monitor detects an anomaly (a high delay or a high loss), it polls the edge devices to obtain the throughput of existing flows. The QoS egress routers measure the outgoing rate of each flow. Using these rates, the monitor computes the total bandwidth consumption by any particular user. For details of throughput approximation, readers are referred to [16]. This bandwidth obtained by an user

is compared to the SLA_{bw} of that user. If any flow gets very high bandwidth than it should, a DDoS attack is flagged. A DoS attack in a downstream domain can be detected by identifying the congested links, and the egress routers connected to the congested links. Using destination IP address prefix matching [21], we check whether many flows are aggregating towards a specific network or host. Consulting with the destination object, we control these flows at the ingress routers, if necessary.

We demonstrate the detection of *no attack* and *severe attack*. "No attack" means no significant traffic in excess of the capacity. This scenario has little loss inside the network domain. This is the normal case of proper network provisioning and enforcing traffic conditioning at the edge routers. A severe attack injects excessive traffic into the network domain from different ingress points. At each ingress point, the flows do not violate the profiles but overall they do. The intensity of the attack is increased during *t=15* seconds to *t=45* seconds. The severe attack causes packet drops of more than 35%. Figure 16 shows that the edge-to-edge delay is increased more than 100% in presence of severe attack. The outcome of one round of loss probing is shown in Figure 17. The distributed schemes detects high losses in links $E2 \rightarrow C2$, $C1 \rightarrow C3$, $C3 \rightarrow C4$, and $C4 \rightarrow E6$. The link $C4 \rightarrow E6$ has a high loss for a short period of time. Since, some TCP flows adjusted their rates, and it causes the link to be non-congested one again. The egress router for the exiting flows is $E6$, and ingress routers through which flows enter into the domain are $E1$, $E2$, $E3$, $E4$, and $E5$. No traffic came from $E7$.

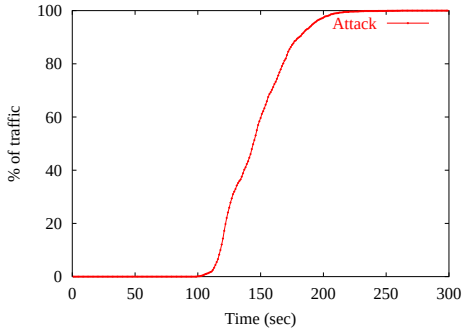


Figure 16: Cumulative distribution function of edge-to-edge delay for link $E1 \rightarrow E6$. High delay indicates presence of severe attack in the domain.

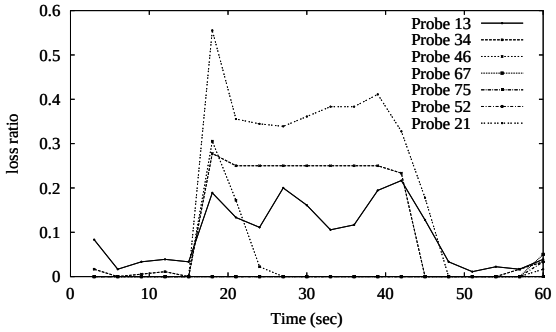


Figure 17: Congestion on multiple probe paths due to severe attack. It indicates multiple links are having high losses.

5.4 Advantages of Distributed Monitoring

The advantages of using *distributed* monitoring are as follows:

1. The simple method of distributed probing requires $O(n)$ probes to identify links with a high loss whereas the stripe-based scheme requires $O(n^2)$ [16], where n is number of edge routers in the domain. The advanced method requires $O(n)$ probes when the network is less than 20% congested, however, it does not exceed $O(n^2)$ in worst case.

2. The distributed scheme is able to detect violations in both directions for any link in the domain, whereas the stripe-based method can detect any violation only if the flow direction of the misbehaving traffic is the

same as the probing direction from the root. To achieve the same result as the distributed monitoring, the stripe-based method needs to probe the whole tree from several different points requiring $O(n^3)$ probes.

3. The distributed scheme can use TCP-based loss measurements (e.g. *Sting* [25]) to detect losses in both directions in one probe cycle.

4. In the stripe based scheme, two leaves/receivers are probed at a time. It takes a long time to complete probing the whole tree. If all leaves are probed simultaneously, in our example, $E1 \rightarrow C1$ link will face huge amount of traffic at that time. On the other hand, the distributed scheme can do parallel probing quite naturally.

6 Conclusions

We have proposed a distributed network monitoring scheme to keep a domain safe from service violations and bandwidth theft attacks. The network is monitored continuously for unusual high delay pattern. When a delay is high, each edge router probes its neighbors in the overlay tree created on top of the physical network. In our monitoring scheme, we do not measure actual loss of all internal links, instead, we identify all congested links with high losses with edge-to-edge measurements. We provide a simple solution to detect congested links. This method requires fewer probes and identifies congested links when the network is sparsely congested. In other cases, the simple method provides a close approximation of congested links instead of the exact number of congested links. To refine the solution of simple method, an advanced method is proposed. Our solution outperforms the existing monitoring mechanism because they require fewer probes to detect attacks in both directions of a link. The number of probes required for the proposed way of monitoring is significantly low (probe traffic is 0.002% of link capacity for an OC3 links). The simulation results indicate that the proposed scheme detects service violations, bandwidth theft attacks, and DoS attacks caused by flow aggregation towards a victim network domain.

Acknowledgments

The authors thank Mohamed Hefeeda and Leszek Lillen for their valuable comments.

References

[1] A. Adams, T. Bu, R. Caceres, N. Duffield, T. Friedman, J. Horowitz, F. Lo Presti, S. B. Moon, V. Paxson, and D. Towsley. The use of end-to-end multicast measurements for characterizing internal network behavior. *IEEE Communications*, 38(5), May 2000.

[2] E. Al-Shaer, H. Abdel-Wahab, and K. Maly. HiFi: a new monitoring architecture for distributed systems management. In *Proc. IEEE 19th International Conference on Distributed Computing Systems (ICDCS'99)*, pages 171–178, Austin, TX, May 1999.

[3] K. G. Anagnostakis, S. Ioannidis, S. Miltchev, J. Ioannidis, Greenwald M., and J. M. Smith. Efficient packet monitoring for network management. In *Proc. IEEE Network Operations and Management Symposium (NOMS)*, Florence, Italy, Apr. 2002.

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

[5] Y. Breitbart, C. Y. Chan, M. Garofalakis, R. Rastogi, and A. Silberschatz. Efficiently monitoring bandwidth and latency in IP networks. In *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001.

[6] T. Bu, N.G. Duffield, F. Lo Presti, and D. Towsley. Network tomography on general topologies. In *ACM SIGMETRICS*, Marina del Rey, CA, 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, Nov. 1997.

[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 *Proc. International Conference on Network Protocols (ICNP)*, pages 37–48, Osaka, Japan, Nov. 2000.

[11] Cisco. Netflow services and applications. http://www.cisco.com/, 2002 May 2000.

[12] M. Dilman and D. Raz. Efficient reactive monitoring. In *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001.

[13] N. G. Duffield, F. Lo Presti, V. Paxson, and D. Towsley. Inferring link loss using striped unicast probes. In *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001.

[14] N.G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley. Network delay tomography from end-to-end unicast measurements. In *Proc. of the 2001 International Workshop on Digital Communications 2001 Evolutionary Trends of the Internet, Italy*, Sept. 2001.

[15] 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.

[16] A. Habib, S. Fahmy, S. R. Avasarala, V. Prabhakar, and Bharat Bhargava. On detecting service violations and bandwidth theft in QoS network domains. *Journal of Computer Communicatons (to appear)*, 2003.

[17] IEPM. Internet End-to-end Performance Monitoring. http://www-iepm.slac.stanford.edu/, 2002.

[18] C. Ji and A. Elwalid. Measurement-based network monitoring and inference: Scalability and missing information. *IEEE journal on selected areas in communications*, 20(4):714–725, May 2002.

[19] J. Kim and J. W. Hong. Distributed QoS monitoring and edge-to-edge QoS aggregation to manage end-to-end traffic flows in differentiated services networks. *Journal of Communications and Networks*, XX(Y):1–20, Dec. 2001.

[20] A. Liotta, G. Pavlou, and G. Knight. Exploiting agent mobility for large-scale network monitoring. *IEEE Network*, May/June, 2002.

[21] M. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *ACM Computer Communication Review*, 32(3):62–73, July 2002.

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

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

[24] G. Sager. Security fun with OCxmon and cflowd. Intenet2 working group meeting, Nov. 98.

[25] S. Savage. Sting: a TCP-based network measurement tool. In *Proc. USENIX Symposium on Internet Technologies and Systems (USITS '99)*, Boulder, CO, Oct. 1999.

[26] R. Stone. Centertrack: An IP overlay network for tracking DoS floods. In *Proc. USENIX Security Symposium*, Denver, CO, Aug. 2000.

[27] R. Subramanyan, J. Miguel-Alonso, and J. A. B. Fortes. A scalable SNMP-based distributed monitoring system for heterogeneous network computing. In *Proc. High Performance Networking and Computing Conference (SC 2000)*, Dallas, TX, 2000.

[28] Waldbusser, S. Remote network monitoring management information base. IETF RFC2819, May 2000.

Appendix A

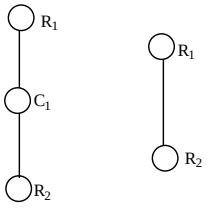


Figure 18: Merging links that do not contribute in distributed probing.

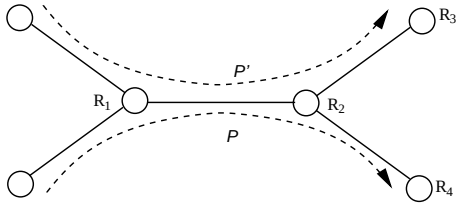


Figure 19: Intersection of probe paths $\mathcal{P}$ and $\mathcal{P}'$. If R_2 is an edge router $R_2 \rightarrow R_3$ and $R_2 \rightarrow R_4$ do not exist.

Proof: Lemma 1. Let the core router c is connected to only two other routers R_1 and R_2. No probe path can be constructed that either includes the link $R_1 \rightarrow C$ and does not include $C \rightarrow R_2$ or vice versa; or includes $R_2 \rightarrow C$ and does not include $C \rightarrow R_1$ or vice versa. The traffic that passes through the link $R_1 \rightarrow C$ also passes through $C \rightarrow R_2$. The traffic that passes through the link $R_2 \rightarrow C$ also passes through $C \rightarrow R_1$. Therefore, for the purpose of probing, a logically equivalent overlay network can be constructed by removing C and replacing the links $R_1 \rightarrow C$ and $C \rightarrow R_2$ with an equivalent single link $R_1 \rightarrow R_2$. $R_2 \rightarrow C$ and $C \rightarrow R_1$ can be replaced by $R_2 \rightarrow R_1$. We say that the link $R_1 \rightarrow R_2$ is congested if and only if at least one of the links $R_1 \rightarrow C$ and $C \rightarrow R_2$ is congested, i.e. the bandwidth of $R_1 \rightarrow R_2$ is the minimum of the bandwidths of $R_1 \rightarrow C$ and $C \rightarrow R_2$. Similarly, the bandwidth of $R_2 \rightarrow R_1$ is the minimum of the bandwidths of $R_2 \rightarrow C$ and $C \rightarrow R_1$. □

Proof: Lemma 4. Let the link $R_1 \rightarrow R_2$ (Figure 19) appears in path $\mathcal{P}$ in the first round of probing and path $\mathcal{P}'$ in the second round of probing. If R_2 is a core router, it is connected to at least two other routers, say R_3 and R_4 (Lemma 2). $\mathcal{P}$ passes through the link $R_2 \rightarrow R_4$ and $\mathcal{P}'$ passes through the link $R_2 \rightarrow R_3$. Since the tree does not have any cycle, $\mathcal{P}$ and $\mathcal{P}'$ never meet again. If R_2 is an edge router, both $\mathcal{P}$ and $\mathcal{P}'$ terminates at R_2. Therefore, $\mathcal{P}$ and $\mathcal{P}'$ can not have any common link in their paths after node R_2. Similarly, it can be shown that $\mathcal{P}$ and $\mathcal{P}'$ can not have common links before they meet at node R_1. That is $|\mathcal{P} \cap \mathcal{P}'| \leq 1$. □

Appendix B

Proof: Lemma 5. To determine the average number of paths a link l lies on, we split the overlay network into two subtrees: T_1 and T_2. The link l lies on an edge-to-edge path whose one end belongs to T_1 and another end belongs to T_2. Let the number of edge routers in T_1 and T_2 be i and $e-i$ respectively. The total possible paths through l is $i(e-i)$. We observe that the probability that T_1 contains i edge routers is, $q_i \propto \frac{1}{i}$, (approximately). i.e. $q_i = \frac{k}{i}$. The average number of paths the link l lies on, $b = \sum_{i=1}^{e/2} q_i.i.(e-i)$.

Now, $\sum_{i=1}^{e/2} q_i = \sum_{i=1}^{e/2} \frac{k}{i} = 1$,i.e. $k = \frac{1}{\ln \frac{e}{2}}$.

Therefor, $b = \sum_{i=1}^{e/2} k(e-i) = \frac{e(3e-2)}{8 \ln \frac{e}{2}} = \frac{e(3e-2)}{8 \ln e - 8 \ln 2} \approx \frac{e(3e-2)}{8 \ln e}$. □

Proof: Lemma 6. There are $e(e-1)$ edge-to-edge paths exist for the advanced method. The number of links in a topology is $\approx 4e$ (see the proof of Theorem 2). The average length of a path $d = b \times \frac{4e}{e(e-1)}$, where $b = \frac{e(3e-2)}{8 \ln e}$ is the average number of paths a link lies on (Lemma 5).

Now, $d = \frac{e(3e-2)}{8 \ln e} \times \frac{4e}{e(e-1)} = \frac{3e-2}{2 \ln e} \times \frac{e}{e-1} \approx \frac{3e}{2 \ln e}$ (for large e). □