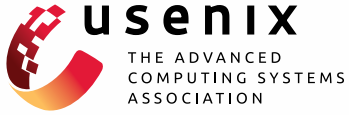


Towards Detecting Target Link Flooding Attack

Lei Xue, ***The Hong Kong Polytechnic University;*** **Xiapu Luo,** ***The Hong Kong Polytechnic University Shenzen Research Institute;*** **Edmond W. W. Chan and Xian Zhan,** ***The Hong Kong Polytechnic University***

https://www.usenix.org/conference/lisa14/conference-program/presentation/xue

This paper is included in the Proceedings of the 28th Large Installation System Administration Conference (LISA14).

November 9–14, 2014 • Seattle, WA

ISBN 978-1-931971-17-1

Open access to the Proceedings of the 28th Large Installation System Administration Conference (LISA14) is sponsored by USENIX

Towards Detecting Target Link Flooding Attack

Lei Xue[†], *Xiapu Luo*[†‡*], *Edmond W. W. Chan*[†], *and Xian Zhan*[†]
Department of Computing, The Hong Kong Polytechnic University[†]
The Hong Kong Polytechnic University Shenzhen Research Institute[‡]
{*cslxue,csxluo*}*@comp.polyu.edu.hk,* {*edmond0chan,chichoxian*}*@gmail.com*

Abstract

A new class of target link flooding attacks (LFA) can cut off the Internet connections of a target area without being detected because they employ legitimate flows to congest selected links. Although new mechanisms for defending against LFA have been proposed, the deployment issues limit their usages since they require modifying routers. In this paper, we propose *LinkScope*, a novel system that employs both the end-to-end and the hop-by-hop network measurement techniques to capture abnormal path performance degradation for detecting LFA and then correlate the performance data and traceroute data to infer the target links or areas. Although the idea is simple, we tackle a number of challenging issues, such as conducting large-scale Internet measurement through noncooperative measurement, assessing the performance on asymmetric Internet paths, and detecting LFA. We have implemented *LinkScope* with 7174 lines of C codes and the extensive evaluation in a testbed and the Internet show that *LinkScope* can quickly detect LFA with high accuracy and low false positive rate.

Keywords: Network Security, Target Link Flooding Attack, Noncooperative Internet Measurement

1 Introduction

DDoS attacks remain one of the major threats to the Internet and recent years have witnessed a significant increase in the number and the size of DDoS attacks [1, 2], not to mention the 300 Gbps direct flooding attacks on Spamhaus and the record-breaking 400 Gbps NTP reflection attack on CloudFlare. However, it is not difficult to detect such bandwidth DDoS attacks, because the attack traffic usually reaches the victim and has difference from legitimate traffic [3].

Recent research discovered a new class of target link flooding attacks (LFA) that can effectively cut off the Internet connections of a target area (or guard area) *without* being detected [4, 5]. More precisely, an attacker first selects persistent links that connect the target area to the Internet and have high flow density, and then instructs bots to generate legitimate traffic between themselves and public servers for congesting those links [5]. If the paths among bots cover the target area, an attacker can also send traffic among themselves to clog the network [4].

It is difficult to detect LFA because (1) the target links are selected by an attacker. Since the target links may be located in an AS different from that containing the target area and the attack traffic will not reach the target area, the victim may not even know he/she is under attack; (2) each bot sends low-rate protocol-conforming traffic to public servers, thus rendering signature-based detection systems useless; (3) bots can change their traffic patterns to evade the detection based on abnormal traffic patterns. Although a few router-based approaches have been proposed to defend against such attacks [6–8], their effectiveness may be limited because they cannot be widely deployed to the Internet immediately. Note that LFA has been used by attackers to flood selected links of four major Internet exchange points in Europe and Asia [6].

Therefore, it is desirable to have a practical system that can help victims detect LFA and locate the links under attack whenever possible so that victims may ask help from upstream providers to mitigate the effect of LFA. We fill this gap by proposing and implementing a system, named *LinkScope*, which employs both end-to-end and hop-by-hop network measurement techniques to achieve this goal. The design of *LinkScope* exploits the nature of LFA including (1) it causes severe congestion on persistent links. Note that light congestion cannot disconnect the target area from the Internet; (2) although the congestion duration will be much shorter than that caused by traditional bandwidth DDoS, the congestion period caused by LFA should not be too short. Otherwise, it cannot cause severe damage to the victim; (3) to cut off the Internet connections of a target area, LFA has to

*The corresponding author.

continuously clog important links. Otherwise, the victim can still access the Internet. *LinkScope* actively collects samples of network path performance metrics and uses abnormal performance degradation to detect LFA.

Although the basic idea is simple, our major contributions lie in tackling a number of challenging issues to realize a practical detection system, including:

1. Since the target links are selected by an attacker, a user has to monitor as many paths as possible. However, the majority of existing network measurement systems have limited scalability because they require installing measurement tools on both ends of each path [9]. We solve this issue from two aspects. First, we design *LinkScope* as a noncooperative measurement tool that only needs the installation on one end of a path. Therefore it can cover much more paths than existing systems. Second, we strategically select important paths for measurement.

2. Due to the prevalence of asymmetric routes [10], we equip *LinkScope* with the capability to differentiate the performance metrics on the forward path (i.e., from the host where *LinkScope* is running to a remote host) and that on the reverse path. It empowers a user to infer which path(s) is under attack.

3. Although network failures may also lead to abnormal path metrics, they will not result in the same effect on all path metrics as that caused by LFA. For example, LFA will cause temporal instead of persistent congestion. By learning the normal profiles of a set of path metrics, *LinkScope* can detect LFA, differentiate it from network failures, and identify different attack patterns.

4. By conducting hop-by-hop measurement, *LinkScope* locates the target link or the target area on the forward path. Although *LinkScope* may not locate the target link on the reverse path in the absence of reverse traceroute data, we will explore possible solutions, such as reverse traceroute, looking glass, etc, in future work.

We have implemented *LinkScope* with 7174 lines of C codes and to the best of our knowledge *LinkScope* is the first system that can conduct both end-to-end and hop-by-hop noncooperative measurement. The extensive evaluations in a testbed and the Internet show that *LinkScope* can quickly detect LFA with high accuracy and low false positive rate.

The rest of this paper is organized as follows. Section 2 describes *LinkScope*'s methodology and Section 3 details the design and implementation of *LinkScope*. The evaluation results obtained from a testbed and the Internet are reported in Section 4. After introducing related work in Section 5, we conclude the paper with future work in Section 6.

2 Methodology

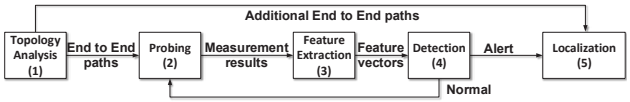


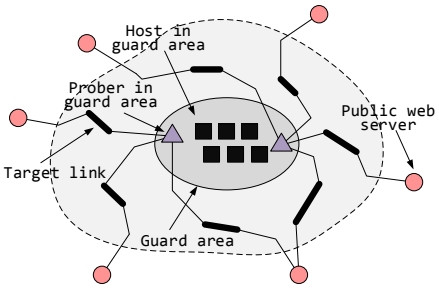
Figure 1: Major steps for detecting LFA and locating target links/areas.

Fig. 1 illustrates the major steps in our methodology for detecting LFA and locating target links/areas whenever possible. The first step, detailed in Section 2.1, involves identifying potential target links and enumerating a set of end-to-end paths that cover potential target links. Depending on the available resource, we conduct noncooperative Internet measurement on selected paths and Section 2.2 describes the measurement method and the corresponding performance metrics. Section 2.3 elaborates on the third and the fourth steps where the feature extraction algorithm turns raw measurement results into feature vectors that will be fed into the detection module for determining the existence of LFA. If there is no attack, the system will continue the measurement. Otherwise, the localization mechanism, introduced in Section 2.4, will be activated for inferring the links or areas under attack.

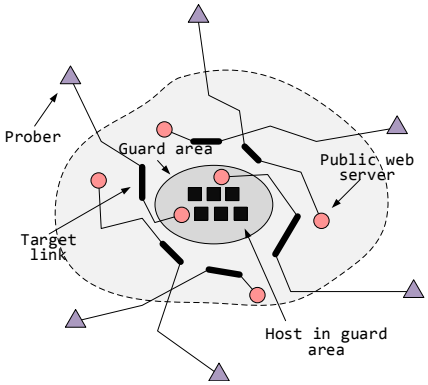
2.1 Topology Analysis

Adopting the noncooperative measurement approach, *LinkScope* only needs to be installed on one end of an Internet path, which is named as a prober. The current implementation of *LinkScope* can use almost any web server as the other end.

There are two common strategies to deploy *LinkScope*. Fig. 2(a) shows the first one, named self-initiated measurement, where *LinkScope* runs on hosts within the guard area. By selecting Web servers in different autonomous systems (AS), a user can measure many diverse Internet paths for covering all possible target links. The second scenario, as illustrated in Fig. 2(b), is the cloud-based measurement where *LinkScope* runs on a group hosts outside the guard area (e.g., virtual machines (VM) in different data centers) and measures the paths between themselves and hosts close to the guard area or even hosts within the guard area. Although the latter case is similar to the scenario of utilizing cooperative measurement systems that require the control of both ends



(a) Self-initiated measurement.



(b) Cloud-based measurement.

Figure 2: Deployment strategies of *LinkScope*.

of a path, using *LinkScope* can simplify the deployment, because only one end needs to install *LinkScope*. By running *LinkScope* on hosts in diverse networks and/or selecting web servers in various locations, the paths under measurement may include all possible target links.

Given a guard area, we first construct the network topology between it and its upstream ASes by performing paris-traceroute [11] from a group of hosts (e.g., VM in clouds or looking glasses [12]) to web servers close to or within the guard area, or using systems like Rocketfuel [13]. From the topology, we can identify potential target links following the LFA's strategy that selects persistent links with high flow density [5]. The flow density of a link is defined as the number of Internet paths between bots and public servers in the target area, which include that link.

Given a set of potential target links denoted as $L = \{l_1, l_2, ..., l_M\}$, we select a set of paths for measurement, which is indicated by $P = \{p_1, p_2, ..., p_N\}$. Since there may be more than one path traversing certain target links, we define three rules to guide the path selection:

- For the ease of locating target links, paths that contain one target link will be selected.
- The number of paths sharing the same remote host should be minimized to avoid mutual interference. It is desirable that each path has different remote host.
- Similar to the second rule, the number of paths initialized by one prober should be minimized to avoid self-induced congestion.

2.2 Measurement approaches

As LFA will congest the selected links, it will lead to anomalies in the following path performance metrics, including,

- Packet loss rate, which will increase because the link is clogged;
- Round-trip time (RTT), which may also increase because of the full queue in routers under attack;
- Jitter, which may have large variations when bots send intermittent bursts of packets to congest the link [14], thus leading to variations in the queue length;
- Number of loss pairs [15], which may increase as a pair of probing packets may often see full queues due to LFA;
- Available bandwidth, which will decrease because the target link is congested;
- Packet reordering, which may increase if the router under attack transmits packets through different routes;
- Connection failure rate, which may increase if the target area has been isolated from the Internet due to severe and continuous LFA.

Besides measuring the above metrics, *LinkScope* should also support the following features:

- Conduct the measurements within a legitimate TCP connection to avoid the biases or noises due to network elements that process TCP/UDP packets in a different manner and/or discard all but TCP packets belonging to valid TCP connections.
- Perform both end-to-end and hop-by-hop measurements. The former can quickly detect the anomalies caused by LFA while the latter facilitates localizing the target links/areas.
- Support the measurement of one-way path metrics because of the prevalence of asymmetric routing.

To fulfill these requirements, *LinkScope* contains the following three probing patterns:

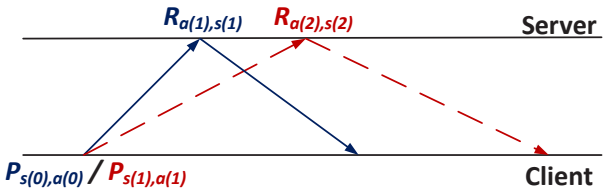


Figure 3: Round trip probing (RTP) pattern.

2.2.1 Round Trip Probing (RTP)

We proposed the Round Trip Probing (RTP) pattern to measure RTT, one-way packet loss, and one-way packet reordering in [16]. As shown in Fig. 3, each RTP measurement involves sending two back-to-back probing packets (i.e., $P_{s(0),a(0)}$ and $P_{s(1),a(1)}$) with customized TCP sequence number (i.e., s(0),s(1)) and acknowledgement number (i.e., a(0) and a(1)) to the remote host. The advertising window of each probing packet is set to 2 maximal segment size (MSS) and therefore each probing packet will elicit one response packet (i.e., $R_{a(1),s(1)}$ and $R_{a(2),s(2)}$). By analyzing the sequence numbers and the acknowledgement numbers in the response packets, we can decide whether there is packet loss/packet reordering occurred on the forward path or the reverse path. If the server supports TCP options like timestamp or SACK, they can ease the detection of forward path packet loss [16]. Moreover, RTT can be measured as the duration from sending $P_{s(0),a(0)}$ to receiving $R_{a(1),s(1)}$.

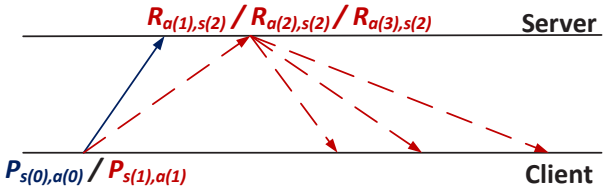


Figure 4: Extended two way probing (eTWP) pattern with $w = 3$.

2.2.2 Extended Two Way Probing (eTWP)

We proposed the original Two Way Probing (TWP) pattern for measuring one-way capacity in [17]. The extended Two Way Probing (eTWP) pattern has similar probing packets as that of TWP. The difference is that eTWP will induce more response packets from the remote host than TWP does. As shown in Fig. 4, TWP(or eTWP) involves sending two back-to-back probing packets (i.e., $P_{s(0),a(0)}$ and $P_{s(1),a(1)}$). The first probing packet uses zero advertising window to prevent the server from sending back responses on the arrival of $P_{s(0),a(0)}$. In TWP, the advertising window in $P_{s(1),a(1)}$ is equal to 2 MSS so that it will trigger two packets from the server [17]. Since a packet train can characterize more loss patterns than a packet pair [18], we enlarge the advertising window in $P_{s(1),a(1)}$ from 2 to w ($w > 2$) in eTWP. Note that increasing w requires *LinkScope* to handle more patterns of response packets.

As the server may dispatch w packets back-to-back if its congestion window allows, we can compute the time gap between the first and the w-th packet, denoted as G_r, and define θ_r to characterize the available bandwidth on the reverse path.

$$\theta_r = \frac{MSS \times (w-1)}{G_r}. \quad (1)$$

Note that θ_r may not be equal to the real available bandwidth [19] but its reduction could indicate congestion [20].

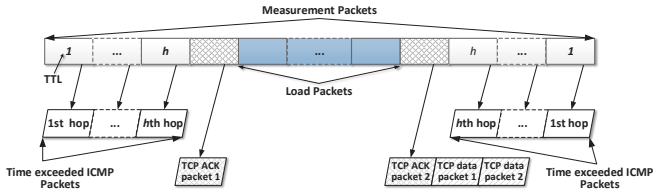
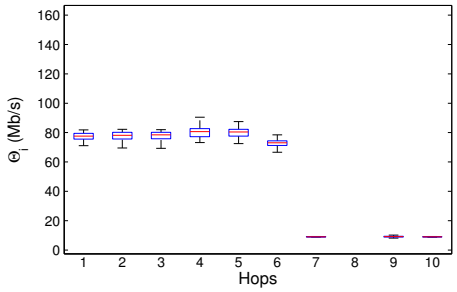


Figure 5: Modified recursive packet train (RPT) pattern.

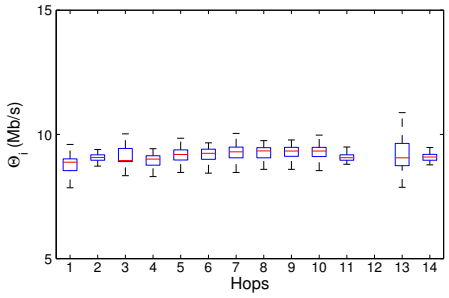
2.2.3 Modified Recursive Packet Train (mRPT)

Hu et al. proposed the original recursive packet train (RPT), which was employed in Pathneck for detecting the location of a network path's bottleneck [20]. The original RPT consists of a group of load packets and a set of TTL-limited measurement packets and Pathneck uses UDP packets to construct RPT. We modify RPT to support end-to-end and hop-by-hop measurements in a TCP connection and remove redundant packets. Fig.5 illustrates the modified RPT, denoted as mRPT, where each rectangle is a probing packet and each parallelogram indicates a response packet triggered by a probing packet. mRPT has h pairs of small measurement packets, whose TTL values are equal to the number in those rectangles. Since a router will send back a time exceeded ICMP packet when a packet's TTL becomes zero, a pair of ICMP packets will be sent back after mRPT passes through a router. We use $G_{I(i)}$ to denote the time gap between the two ICMP packets from the i-th hop. *LinkScope* does not use a fixed number of measurement packets (e.g., 30 in Pathneck [20]) because we do not want them to reach the server and LFA usually targets on links outside the victim's network. Instead, *LinkScope* first determines h by doing a traceroute.

The load packets are customized TCP packets that belong to an established TCP connection and carry an invalid checksum value or a TCP sequence number so that they will be discarded by the server. There are two special packets (i.e., R1 and R2) between the load packets sand the measurement packets. They have the same size as the load packets and work together to accomplish two tasks: (1) each packet triggers the server to send back a TCP ACK packet so that the prober can use the time gap between these two ACK packets, denoted as G_A, to estimate the interval between the head and tail load packet; (2) induce two TCP data packets from the server to start the measurement through RTP [16]. To achieve these goals, *LinkScope* prepares a long HTTP request whose length is equal to two load packets and puts half of it to R1 and the remaining part to R2. To force the server to immediately send back an ACK packet on the arrival of R1 and R2, we first send R2 and then R1, because a TCP server will send back an ACK packet when it receives an out-of-order TCP segment or a segment that fills a gap in the sequence space [21].



(a) Path from Korea to Hong Kong



(b) Path from Taiwan to Hong Kong

Figure 6: θ_i measured on two paths to Hong Kong.

To characterize the per-hop available bandwidth and end-to-end available bandwidth, *LinkScope* defines θ_i (i=1,...,h) and θ_e as follows:

$$\theta_i = \frac{S_L \times (N_L+2) + S_M \times (h-i)}{G_{I(i)}}, i = 1,\ldots,h, \quad (2)$$

$$\theta_e = \frac{S_L \times N_L}{G_A}, \quad (3)$$

where S_L and S_M denote the size of a load packet and that of a measurement packet, respectively. N_L is the number of load packets.

Note that since the packet train structure cannot be controlled after each hop, similar to θ_r, θ_i (or θ_e) may not be an accurate estimate of per-hop available bandwidth (or end-to-end available bandwidth) but their large decrement indicates serious congestion [20]. Since LFA will lead to severe congestion on target links, θ_i of the target link or θ_e on the path covering the target link will be throttled.

Fig.6 shows θ_i on two paths to a web server in our campus, whose last four hops are located in the campus network. Since the last but two hops did not send back ICMP packets, there is no θ_i on that hop. On the path from Korea to Hong Kong, θ_i drops from around 80Mbps to around 9Mbps on the 7th hop. It is because the bandwidth of each host in campus network is limited to 10Mbps. On the path from Taiwan to Hong Kong, θ_i is always around 9Mbps. It may be due to the fact the first hop's available bandwidth is around 9Mbps.

2.3 Anomaly detection

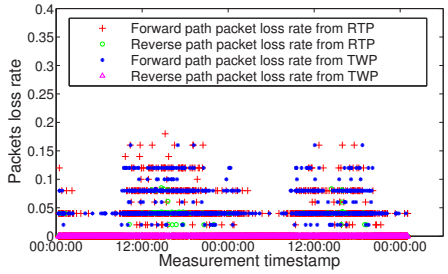
Table 1: Detail metrics measured during one probe.

Direction	Metric	Defination
Forward	θ_e	Characterizing available bandwidth through mRPT.
	R_{RFPL}	Packet loss rate from RTP.
	R_{TFPL}	Packet loss rate from eTWP.
	R_{RFPL2}	Loss pair rate from RTP.
	R_{TFPL2}	Loss pair rate from eTWP.
	R_{RFPR}	Packet reordering rate from RTP.
	R_{TFPR}	Packet reordering rate from eTWP.
Reverse	θ_r	Characterizing available bandwidth through eTWP.
	R_{RRPL}	Packet loss rate from RTP.
	R_{TRPL}	Packet loss rate from eTWP.
	R_{RRPL2}	Loss pair rate from RTP.
	R_{TRPL2}	Loss pair rate from eTWP.
	R_{RRPR}	Packet reordering rate from RTP.
	R_{TRPR}	Packet reordering rate from eTWP.
Round – trip	RTT	Round-trip time.
	J_{RTT}	Round-trip time variation (jitter).
	$Fail_{RTP}$	Connection failure rate in RTP.
	$Fail_{TWP}$	Connection failure rate in eTWP.

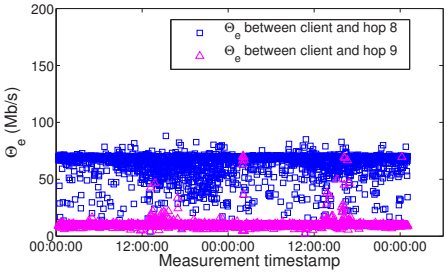
We define two metric vectors in Eqn. (4) and (5), which cover selected performance metrics, for the forward path and the reverse path, respectively. Table 1 lists the meaning of each performance metric.

$$\overrightarrow{F_{forward}} = \{\theta_e, R_{RFPL}, R_{TFPL}, R_{RFPL2}, R_{TFPL2}, R_{RFPR}, R_{TFPR}, RTT, J_{RTT}, Fail_{RTP}, Fail_{TWP}\}^{\mathrm{T}} \quad (4)$$

$$\overrightarrow{F_{reverse}} = \{\theta_r, R_{RRPL}, R_{TRPL}, R_{RRPL2}, R_{TRPL2}, R_{RRPR}, R_{TRPR}, RTT, J_{RTT}, Fail_{RTP}, Fail_{TWP}\}^{\mathrm{T}} \quad (5)$$



(a) Packet loss rate.



(b) θ_e

Figure 7: Performance of a path from Japan to Hong Kong over 48 hours.

LinkScope keeps collecting samples of these metrics and builds a normal profile for each path using the data in the absence of LFA. Since the measurement results show a diurnal pattern, we build the normal profile for each or several hours per day. For example, Fig. 7(a) shows the diurnal pattern of forward path packet loss rate and θ_e on a path from Japan to Hong Kong over 48 hours.

Then, *LinkScope* uses the Mahalanobis distance [22] to quantify the difference between the profile and a new round of measurement results as follows:

$$D_M(\overrightarrow{F}) = \sqrt{(\overrightarrow{F} - \overrightarrow{\lambda})^{\mathrm{T}} \Omega^{-1} (\overrightarrow{F} - \overrightarrow{\lambda})}, \quad (6)$$

where $\overrightarrow{F}$ is the metric vector from a round of measurement results described in Section 3. $\overrightarrow{\lambda}$ denotes the mean metric vector in the profile and Ω is the covariance matrix .

$$\Omega = \frac{1}{n-1} \sum_{i=1}^{n} (\lambda_i - \bar{\lambda})(\lambda_i - \bar{\lambda})^T, \quad (7)$$

where λ_i is the i-th metric in the profile, n is the number of metrics and

$$\bar{\lambda} = \frac{1}{n} \sum_{i=1}^{n} \lambda_i. \quad (8)$$

Finally, *LinkScope* employs the non-parametric cumulative sum (CUSUM) algorithm [23] to capture the abrupt changes in the Mahalanobis distance (i.e., D_M). The non-parametric CUSUM algorithm assumes that the average distance is negative in normal situation and becomes positive when path is under attack. We use D_n to denote the distance measured in n-th probe and turn $\{D_n\}$ into a new sequence $\{X_n\}$ through

$$X_n = D_n - \overline{D_n}, \quad (9)$$

$$\overline{D_n} = mean(D_n) + \alpha std(D_n), \quad (10)$$

where α is an adjustable parameter, $mean(D_n)$ is the mean value of D_n, and $std(D_n)$ is the standard deviation of D_n. The non-parametric CUSUM algorithm defines a new sequence $\{Y_n\}$ by Eqn. (11).

$$Y_n = \begin{cases} (Y_{n-1} + X_n)^+, & n > 0, \\ 0, & n = 0, \end{cases} \quad where \quad x^+ = \begin{cases} x, & x > 0, \\ 0, & otherwise. \end{cases} \quad (11)$$

Since the Mahalanobis distance quantifies the difference between the profile and a new observation, a measurement result showing better network performance may also be regarded as anomalies. To remedy this problem, we only consider the alerts where the measured performance metrics become worse than the normal profile (e.g. smaller θ_e and larger packet loss rate) because of the nature of LFA.

2.4 Locating the target link

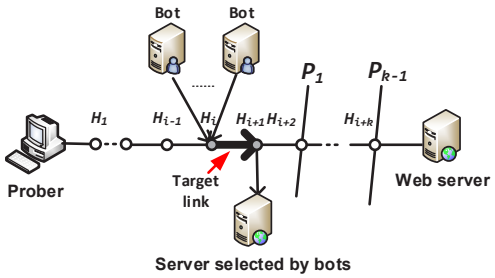


Figure 8: Locating the target links.

When performance anomaly is detected on a forward path, *LinkScope* tries to locate the target link through two steps. We use an example shown in Fig.8 to illustrate the steps, where bots send traffic to the server selected by bots in order to congest the link between H_i and H_{i+1}.

First, based on the hop-by-hop measurement results from mRPT, *LinkScope* knows that the path from H_1 to H_{i-1} is not under attack. Second, according to the topology analysis, *LinkScope* will perform measurement on other paths that cover the hops after H_i, such as P_1 going through H_{i+1} and P_{k-1} covering H_{i+k}. If one new path (e.g. the one covering H_{i+j}) does not have poor performance like the original path, then the target link is in the area from H_i to H_{i+j-1}. The rational behind this approach comes from the nature of LFA that congests a selected link so that all paths including that link will suffer from similar performance degradation. By contrast, other paths will not have similar patterns.

Since the paths identified in Section 2.1 may not cover all hops on the original path, we propose the following steps to look for new paths.

1. For a hop, H_k, we utilize high-speed scanning tools such as Zmap [24] to look for web servers in the same subnet as H_k, which can be determined through systems like traceNET [25]. If a web server is found, *LinkScope* performs traceroute to this web server and checks whether the path to the server goes through H_k.

2. We look for web servers located in the same AS as H_k and then check whether the paths to those web servers go through H_k.

3. We look for web servers located in the buddy prefix [26]as H_k and then check whether the paths to those web servers go through H_k.

4. If no such path can be found, we check next hop.

We acknowledge that this method may not be applied to reverse paths because it is difficult to get the traceroute on the reverse path (i.e., from the remote host to the prober). In future work, we will explore two possible approaches to tackle this issue. First, the victim may combine both self-initiated measurement and cloud-based measurement using hosts under control after anomalies are detected. Second, using reverse traceroute [27] or looking glass [28] may be able to obtain the traceroute on the reverse path.

3 *LinkScope*

In this section, we detail the design of *LinkScope* whose architecture is illustrated in Fig. 9. We have implemented *LinkScope* with 7174 lines of C codes and the extensive evaluation results obtained in a testbed and the Internet are reported in Section 4.

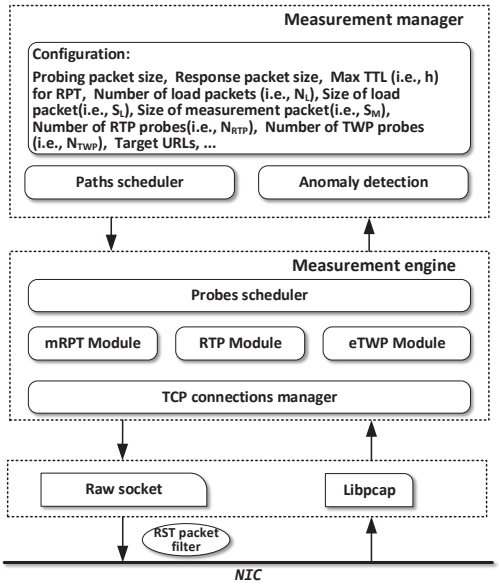


Figure 9: The architecture of *LinkScope*.

3.1 Measurement Manager

The original designs of RTP, TWP, and RPT are not limited to specific application layer protocol. We use HTTP as the driving protocol because tremendous number of web servers are publicly available for the measurement. In future work, we will explore other protocols.

We also realize a tool named *WebChecker* to collect basic information about the path and the remote server. It runs Paris-traceroute [11] to determine the number of hops between a prober and the server, and then sets h so that the measurement packet in mRPT can reach the network perimeter of the server.

WebChecker also enumerates suitable web objects in a web server and output a set of URLs. It prefers to fetching static web objects (e.g., figure, pdf, etc.) starting from the front page of a web site and regards a web object as a suitable one if its size is not less than 10K bytes. Furthermore, similar to TBIT [29], *WebChecker* will check whether the web server supports TCP options, including Timestamp, Selective Acknowledgment(SACK), and HTTP header options such as Range [30]. These options may simplify the process of *LinkScope* and enhance its capability. For example, if the server supports MSS, *LinkScope* can control the size of response packets. Supporting Timestamp and SACK can ease the detection of forward path packet loss [16].

The paths scheduler in *LinkScope* manages a set of probing processes, each of which conducts the measurement for a path. To avoid self-induced congestion, the path scheduler will determine when the measurement for a certain path will be launched and how long a path will

be measured. Currently, each path will be measured for 10 minutes. The probing packet size, the response packet size, and the load packet size are set to 1500 bytes. The number of load packets is 20 and the size of measurement packet is 60 bytes. The number of RTP probes and the number of TWP probes are equal to 30. All these parameters can be configured by a user.

The collected measurement results will be sent to the anomaly detection module for detecting LFA.

3.2 Measurement Engine

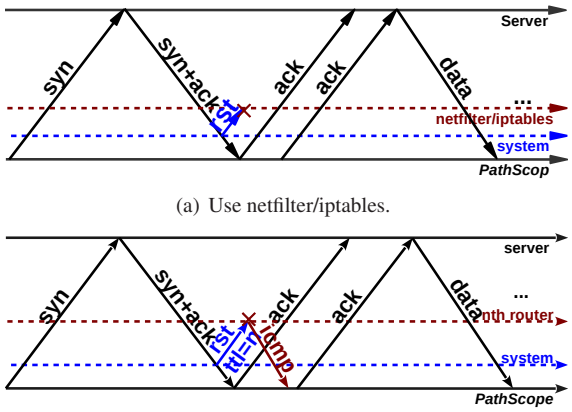
In the measurement engine, the probes scheduler manages the measurements on a path. A round of measurement consists of one probe based on the mRPT pattern, N_{RTP} probes based on the RTP pattern, and N_{TWP} probes based on the eTWP pattern. A probe consists of sending the probing packets and processing the response packets. After finishing a round of measurement, the probes scheduler will deliver the parsed measurement results to the anomaly detection module and schedule a new round of measurement.

The mRPT, RTP, and eTWP modules are in charge of preparing the probing packets and handling the response packets according to the corresponding patterns. Before conducting measurement based on mRPT, *LinkScope* sets each measurement packet's IPID to its TTL. Since each pair of measurement packets will trigger two ICMP packets, *LinkScope* inspects the ICMP packet's payload, which contains the IP header and the first 8 bytes of the original packet's data, for matching it to the measurement packet.

It is worth noting that in each round of measurement for a path all probes are performed within *one* TCP connection. Such approach can mitigate the negative effect due to firewall and instable routes, because stateful firewall will drop packets that do not belong to any established TCP connection and load balancer may employ the five tuple of <src IP, src Port, dst IP, dst Port, Protocol> to select routes.

The TCP connections manager will establish and maintain TCP connections. If the server supports TCP options like MSS, Timestamp, and SACK, the TCP connections manager will use MSS option to control the size of response packet (i.e., the server will use the minimal value between its MSS and the MSS announced by *LinkScope*). It will also put the SACK-permitted option and TCP timestamp option into the TCP SYN packet sent from *LinkScope* to the server.

Since *LinkScope* needs to control the establishment of TCP connections and customize probing packets (e.g., sequence number, acknowledgement number, advertising window), all packets are sent through raw socket. Moreover, *LinkScope* uses the libpcap library to capture all response packets and then parses them for computing performance metrics.



(a) Use netfilter/iptables.

(b) Modify TTL of RST packet

Figure 10: RST packet filter.

3.3 RST packet filter

Since *LinkScope* constructs all packets by itself and sends them out through raw socket, OS does not know how to handle the response packets and therefore it will reply with an RST packet to the server to close the TCP connections. We employ two approaches to filter out RST packets generated by OS.

As shown in Fig. 10(a), if the system supports netfilter/iptables [31], we use it to drop all RST packets except those generated by *LinkScope*. We differentiate the RST packets from OS and that from *LinkScope* through the IPID value in the IP header because *LinkScope* will set the IPID value of its RST packets to a special value.

Since some hosts do not support netfilter/iptables, such as those Planetlab nodes [32], we propose another method as shown in Fig. 10(b). *LinkScope* first establishes a TCP connection with the web server using stream socket (i.e., SOCK_STREAM), and then uses the function *setsockopt* to set the TTL value in each packet generated by OS to a small value so that it will not reach the web server. Moreover, *LinkScope* utilizes the libpcap library to capture the TCP three-way handshaking packets generated by OS to record the initial sequence numbers selected by the local host and the web server along with other information related to the TCP connection, such as source port, and TCP options. After that, *LinkScope* will create and send probing packets through raw socket with the help of such information.

4 Evaluation

We carry out extensive experiments in a test-bed and the Internet to evaluate *LinkScope*'s functionality and overhead.

4.1 Test bed

Fig. 11 shows the topology of our test bed that connects to the Internet through the campus network. All hosts run Ubuntu system. Host 1 and Host 2 act as attackers and the public server used by attackers, respectively. D-ITG [33] is used to generate traffic for congesting the MikroTik router in red circle. The router serves as the bottleneck with 10Mbps bandwidth. Host 3 is a bridge for emulating packet loss and packet reordering and Host 4 is an NAT-enable router providing port forwarding in order to connect the web server and the LAN to the Internet. In our experiment, LAN denotes the guard area and the web server is a public server that can be accessed by nodes in the Internet. We deploy *LinkScope* on Planetlab nodes and Amazon EC2 instances.

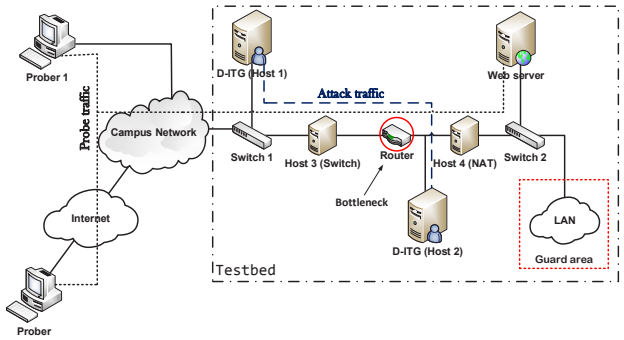


Figure 11: The topology of the testbed.

4.2 Emulated Attacks in the Test bed

To demonstrate that *LinkScope* can capture different kinds of LFA, we emulate four types of LFA in the testbed and use the abnormal changes in θ_e to illustrate the diverse effect due to different attacks. If the attacker floods the bottleneck with high-volume traffic, all TCP connections including the one for measurement are disconnected and θ_e becomes zero all the time. Therefore, we did not show it.

Fig. 12(a) shows θ_e under pulsing LFA where the attacker transmits high-volume bursts of traffic to congest the bottleneck [14]. The attack traffic rate is 1600 packets per second and the packet size is uniformly distributed in the range of [600, 1400] bytes. In the absence of attack, θ_e is close to the available bandwidth. Under the attack, since the bottleneck is severely congested and all connections are broken, θ_e becomes zero.

Fig.12(b) illustrates θ_e under LFA with two attack traffic rates: 400 packets per second and 800 packets per second. An attacker may change the attack traffic rate for evading the detection. We can see that when the attack rate decreases (or increases), θ_e increases (or decreases), meaning that it can capture the changes in the attack traffic rate.

Fig.12(c) represents θ_e under gradual LFA where the attack traffic rate increases from zero to a value larger than the capacity of the bottleneck. It emulates the scenario of DDoS attacks in Internet where the traffic sent from different bots may not reach the bottleneck simultaneously, thus showing the gradual increase in the attack traffic rate. Although the TCP connection for measurement was broken when the attack traffic rate almost reached its maximal value, the decreasing trend of θ_e can be employed to raise an early alarm.

Fig.12(d) demonstrates θ_e when a network element randomly drops packets. It may be due to occasional congestion or the use of random early drop (RED) in routers. We can see that although θ_e varies its values are still close to the available bandwidth.

Since LFA will cause severe intermittent congestion on target links in order to cut off the Internet connections of the guard area, we can use different patterns in performance metrics to distinguish it from other scenarios, such as long-term flooding and cable cut which will disable the Internet connection for quite a long period of time, and even identify different types of attacks, as demonstrated in Fig. 12.

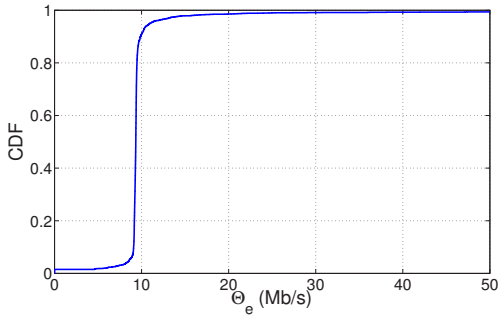
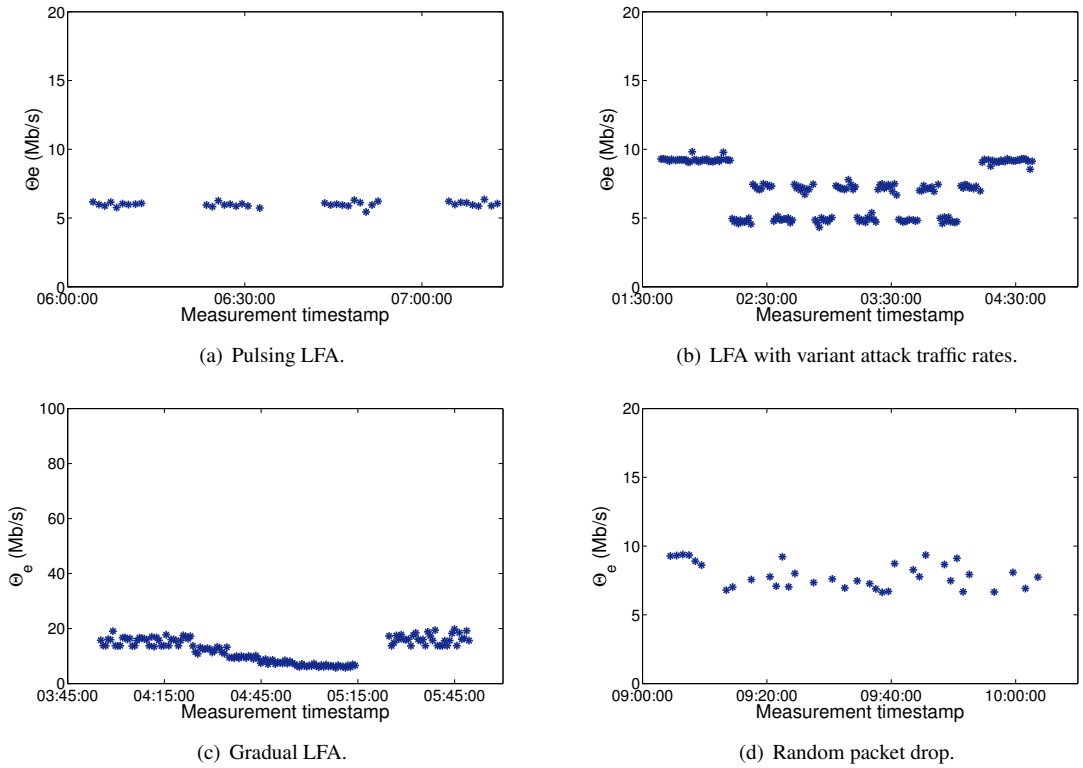


Figure 14: CDF of θ_e on path from Amsterdam to Hong Kong.

4.3 Internet Probing

To evaluate the capability and the stability of *LinkScope*, we run it on Planetlab nodes to measure paths to Hong Kong for two days and paths to Taiwan for seven days.



(a) Pulsing LFA.

(b) LFA with variant attack traffic rates.

(c) Gradual LFA.

(d) Random packet drop.

Figure 12: Available bandwidth measured with different attacks from Prober 1 to testbed.

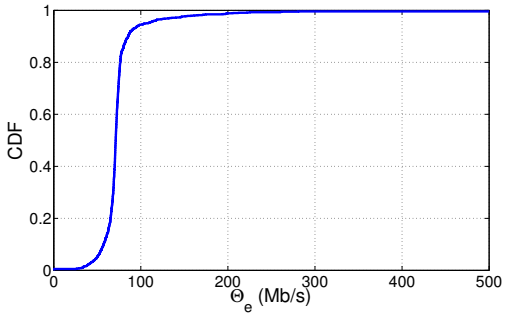


Figure 16: CDF of θ_e on path from Santa Barbara to Taipei.

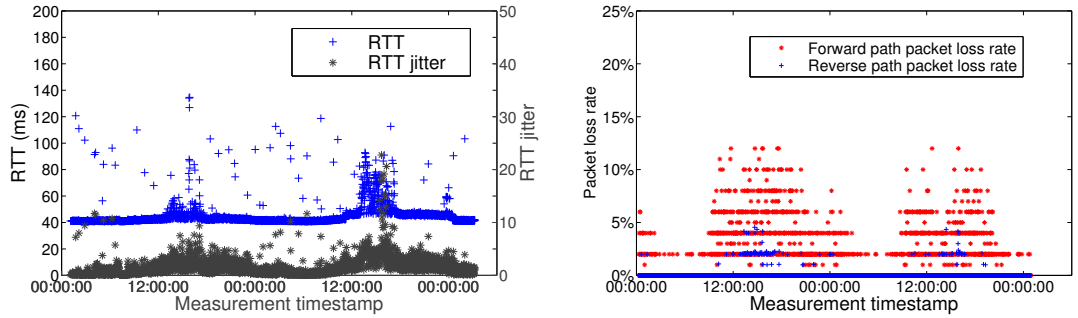
Fig.13 shows the performance metrics measured on the path from Amsterdam to Hong Kong for two days. It demonstrates the diurnal patterns in forward path/reverse path packet loss, RTT, and jitter. The path performance is better and more stable in the period from 00:00 to 12:00 than that during the period from 12:00 to 24:00. The increased loss rate may affect the measurement of θ_e as some measurement results deviate during the period from 12:00 to 24:00 as shown in Fig. 13(d). Fig.14 illustrates the CDF of θ_e on the path from Amsterdam to Hong Kong, where θ_e concentrates on 9 Mb/s.

Fig.15 demonstrates the performance metrics measured on the path from Santa Barbara (US) to Taipei for seven days. This path has stable good performance. For example, RTT is around 150ms and the jitter is less than 10 shown Fig. 15(a). The loss rate is less than 2% and there is no packet reordering. The estimated end-to-end θ_e is around 75Mbps as illustrated in Fig. 15(d) and Fig.16. Since LFA will cause severe congestion during a short duration, it will cause obvious abrupt changes in the performance metrics and get caught by *LinkScope*.

4.4 Detection Performance

We first evaluate *LinkScope*'s false positive rate using Internet measurement results on different paths, and then assess its detection rate using emulated attacks in our test bed.

On the paths to Hong Kong, *LinkScope* conducts measurement once per minute for two days (48 hours). We divide the one-day data (24 hours) into 24 sets (one set per hour), because features are changing over time. We use the data obtained in the first day as the training data and use the remaining data to evaluate *LinkScope*'s false positive rate. Table 2 lists the false positive rates on eight paths to Hong Kong with different α. The first four probers are Amazon EC2 VM and the last four are



(a) RTT and RTT jitter.

(b) Packet loss rate.

(c) Packet reordering rate.

(d) θ_e.

Figure 13: Performance metrics measured on the path from Amsterdam to Hong Kong for two days.

Planetlab nodes. In this experiment, we divide one-day data equally into 24 segments. The false positive rates are all less than 10% and it decreases when α increases, because α serves as a threshold and a larger α may cover more normal scenarios. Moreover, all false positive rates on the path from South Carolina (PL node) to Hong Kong are 0, because the performances of all metrics are very stable during both days. Table 2 shows that all false positive rates are smaller than 6% when α is not less than 30.

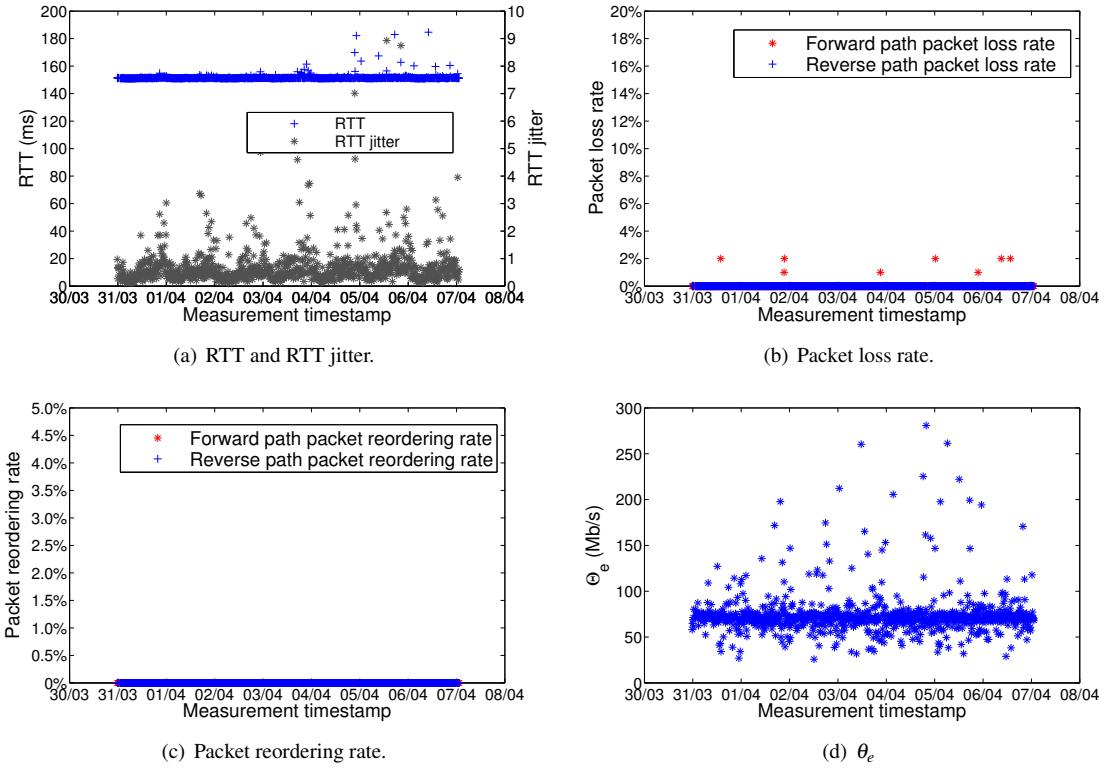
Table 3 shows false positive rate on the paths from five Planetlab nodes and two Amazon EC2 hosts to Taiwan. On these paths, *LinkScope* conducts measurement once per ten minutes for seven days. We also take the data in the first day as the training data and the remaining data for evaluation. Table 3 shows that the increases of α can decrease the false positive rate, except the path from US West to So-net Entertainment where no false positive detected, as a result of the stable performances during the two days.

By inspecting false positive cases, we find that almost all the false positives are due to connection failure. It may happen even without attack. Take the path from Tokyo to Hong Kong as an example, the connection failure rate in two days is 4.06%. This rate varies over time, such as, 0.90% during the period from 00:00 to 12:00 and 7.5% for the period from 12:00-24:00, because the network performance is much more unstable from 12:00 to 00:00 (such as shown in Fig.13). However, in the absence of LFA, the connection failures scatters over time while the connection failures appear continuously in the presence of LFA.

Table 4: Detection rate.

Training data	path	$\alpha = 10$	$\alpha = 20$	$\alpha = 30$
20 probes	path 1	100.0%	100.0%	100.0%
20 probes	path 2	100.0%	100.0%	100.0%
40 probes	path 1	100.0%	100.0%	100.0%
40 probes	path 2	100.0%	100.0%	100.0%

To evaluate *LinkScope*'s detection rate, we emulate different attacks between Host 1 and Host 2 as shown Fig.11. During the pulsing LFA and gradual LFA, the detection rate are always 100%. Because when the attack traffic rate is much higher than the available bandwidth, the path is congested and none response packets can be received from the destination all the time. Table 4 lists the detection rates when the attack traffic rate is a little higher than the bandwidth (1.2 times of bandwidth). In this case, *LinkScope* can still receive some response packets and compute the measurement results. Table 4 shows that the anomaly detection rates are still



(a) RTT and RTT jitter.

(b) Packet loss rate.

(c) Packet reordering rate.

(d) θ_e

Figure 15: Performance metrics measured on the path from Santa Barbara to Taipei for seven days.

Table 2: False positive rate on paths to Hong Kong.

Prober type	Path	$\alpha = 10$	$\alpha = 20$	$\alpha = 30$	$\alpha = 40$	$\alpha = 50$
EC2	Virginia - Hong Kong	6.32%	5.99%	5.12%	4.33%	3.67%
EC2	Tokyo - Hong Kong	5.88%	4.02%	2.85%	2.07%	1.94%
EC2	Ireland - Hong Kong	8.69%	7.24	5.75%	5.23%	4.58%
EC2	Sao Paulo - Hong Kong	5.80%	2.90%	2.52%	1.55%	1.16%
PL node	Tokyo - Hong Kong	2.19%	1.19%	0.60%	0.60%	0.60%
PL node	Amsterdam - Hong Kong	4.18%	2.61%	1.69%	1.30%	0.91%
PL node	Beijing - Hong Kong	3.54%	2.96%	2.06%	2.67%	1.28
PL node	South Carolina - Hong Kong	0	0	0	0	0

Table 3: False positive rate on paths to Taiwan with different configures.

Prober type	Path	$\alpha = 20$	$\alpha = 30$	$\alpha = 40$
PL node	Boston - Taipei	2.17%	1.45%	0.97%
PL node	Urbana - Taipei	2.18%	1.69%	1.45%
PL node	Turkey - Taipei	2.20%	2.19%	1.21%
PL node	Tokyo - Taipei	1.59%	3.17%	3.17%
PL node	Blacksburg - Taipei	1.76%	1.25%	1.00%
PL node	Tokyo - ChungHwa Telecom	2.32%	1.45%	1.01%
EC2	Sao Paulo - Taichung	6.56%	4.01%	2.15%
EC2	US West - So-net Entertainment	0	0	0

100% though the attacks cannot fully clog the bottleneck.

4.5 System load

To evaluate the system load introduced by *LinkScope*, we use htop [34] to measure the client's and web serv-

Table 5: The CPU utilizations and Load average in the probing client and web server during measurement.

Probing processes	Measurement rate (Hz)	Probing client		Web sever	
		Load average	CPU utilization	Load average	CPU utilization
0	0	0.01	0.3%	0.00	0.5%
1	2	0.06	0.3%	0.00	0.5%
1	10	0.10	0.3%	0.01	0.6%
2	10	0.10	0.4%	0.01	0.6%
10	10	0.11	1.7%	0.02	0.7%
50	10	0.23	2.4%	0.08	0.8%
100	10	0.47	2.7%	0.09	0.8 %

er's average load and average CPU utilization when *LinkScope* runs with different configurations. The client, running Ubuntu 12.04 LTS system, is equipped with Intel 3.4 GHz i7-4770 CPU, 16G memory, and 1Gbps NIC, and the web server is equipped with Intel 2.83 GHz Core(TM)2 Quad CPU and runs Ubuntu 12.04 LTS system and Apache2.

Table 5 lists the results for both the client and the server. The first line represents the load and CPU utilization without *LinkScope* and we ensure that no other routine processes are executed on both machines during the measurement. We can see that even when there are 100 probing process with 10Hz measurement rates, the average loads and average CPU utilizations are still very low on both machines, especially for the web server.

5 Related work

Network anomaly detection can be roughly divided into two categories: performance related anomalies and security related anomalies [35]. The performance related anomalies include transient congestion, file sever failure, broadcast storms and so on, and security related network anomalies are often due to DDoS attacks [36–39] that flood the network to prevent legitimate users from accessing the services. *PachScope* employs various performance metrics to detect a new class of target link flooding attacks (LFA).

Anomaly detection attempts to find patterns in data, which do not conform to expected normal behavior [40]. However, LFA can evade such detection because an attacker instructs bots to generate legitimate traffic to congest target links and the attack traffic will never reach the victim's security detection system. Instead of passively inspecting traffic for discovering anomalies, *LinkScope* conducts noncooperative active measurement to cover as many paths as possible and captures the negative effect of LFA on performance metrics.

Although active network measurement has been employed to detect network faults and connectivity problems [41–48], they cannot be directly used to detect and locate LFA because of two major reasons. First, since LFA will cause temporal instead of persistent congestion, existing systems that assume persistent connection problems cannot be used [41–43]. Second, since LFA avoids causing BGP changes, previous techniques that rely on route changes cannot be employed [44–47]. Moreover, the majority of active network measurement systems require installing software on both ends of a network path, thus having limited scalability. To the best of our knowledge, *LinkScope* is the first system that can conduct both end-to-end and hop-by-hop noncooperative measurement, and takes into account the anomalies caused by LFA.

Router-based approaches have been proposed to defend against LFA and other smart DoS attacks [6–8, 49–52], their effectiveness may be limited because they cannot be widely deployed to the Internet immediately. By contrast, *LinkScope* can be easily deployed because it conducts noncooperative measurement that only requires installation at one end of a network path. Although *LinkScope* cannot defend against LFA, it can be used along with traffic engineering tools to mitigate the effect of LFA.

Existing network tomography techniques cannot be applied to locate the target link, because they have many impractical assumptions (e.g., multicast [53], source routing [54]). Although binary tomography may be used for identifying faulty network links [55], it just provides coarse information [56] and and is not suitable for locating the link targeted by LFA, because it adopts assumptions for network fault (e.g., only one highly congested link in one path [57], faulty links nearest to the source [58]). LFA can easily invalidate them. Moreover, the probers in network tomography create a measurement mesh network [59,60] whereas in our scenarios there is only one or a few probers that may not communicate with each other.

6 Conclusion

In this paper, we propose a novel system, *LinkScope*, to detect a new class of target link-flooding attacks (LFA) and locate the target link or area whenever possible. By exploiting the nature of LFA that causes severe congestion on links that are important to the guard area, *LinkScope* employs both the end-to-end and the hop-by-hop network measurement techniques to capture abrupt performance degradation due to LFA. Then, it correlates the measurement data and the traceroute data to infer the target link or area. After addressing a number of challenging issues, we have implemented *LinkScope* with 7174 lines of C codes and conduct extensive evaluation in a testbed and the Internet. The results show that *LinkScope* can quickly detect LFA with high accuracy and low false positive rate. In future work, we will conduct large-scale and continuous measurements to evaluate *LinkScope* and investigate the optimal deployment of *LinkScope*.

7 Acknowledgment

We thank the reviewers for their comments and suggestions and Paul Krizak, in particular, for shepherding our paper. This work is supported in part by the CCF-Tencent Open Research Fund, the Hong Kong GRF (No. PolyU 5389/13E), the National Natural Science Foundation of China (No. 61202396,60903185), and the Open Fund of Key Lab of Digital Signal and Image Processing of Guangdong Province.

References

[1] Incapsula, "2013-2014 DDoS threat landscape report," 2014.

[2] ARBOR, "DDoS and security reports," http://www.arbornetworks.com/asert/, 2014.

[3] M. Geva, A. Herzberg, and Y. Gev, "Bandwidth distributed denial of service: Attacks and defenses," *IEEE Security and Privacy*, Jan.-Feb. 2014.

[4] A. Studer and A. Perrig, "The coremelt attack," in *Proc. ESORICS*, 2009.

[5] M. Kang, S. Lee, and V. Gligor, "The crossfire attack," in *Proc. IEEE Symp. on Security and Privacy*, 2013.

[6] S. Lee, M. Kang, and V. Gligor, "Codef: collaborative defense against large-scale link-flooding attacks," in *Proc. ACM CoNEXT*, 2013.

[7] S. Lee and V. Gligor, "Floc: Dependable link access for legitimate traffic in flooding attacks," in *Proc. IEEE ICDCS*, 2010.

[8] A. Athreya, X. Wang, Y. Kim, Y. Tian, and P. Tague, "Resistance is not futile: Detecting ddos attacks without packet inspection," in *Proc. WISA*, 2013.

[9] M. Crovella and B. Krishnamurthy, *Internet Measurement: Infrastructure, Traffic and Applications*. Wiley, 2006.

[10] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker, "On routing asymmetry in the internet," in *Proc. IEEE GLOBECOM*, 2005.

[11] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with Paris traceroute," in *Proc. ACM IMC*, 2006.

[12] A. Khan, T. Kwon, H. Kim, and Y. Choi, "AS-level topology collection through looking glass servers," in *Proc. ACM IMC*, 2013.

[13] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," in *Proc. ACM SIGCOMM*, 2002.

[14] X. Luo and R. Chang, "On a new class of pulsing Denial-of-Service attacks and the defense," in *Proc. NDSS*, 2005.

[15] E. Chan, X. Luo, W. Li, W. Fok, and R. K. Chang, "Measurement of loss pairs in network paths," in *Proc. ACM IMC*, 2010.

[16] X. Luo, E. Chan, and R. Chang, "Design and implementation of TCP data probes for reliable and metric-rich network path monitoring," in *Proc. USENIX ATC*, 2009.

[17] E. Chan, A. Chen, X. Luo, R. Mok, W. Li, and R. Chang, "TRIO: Measuring asymmetric capacity with three minimum round-trip times," in *Proc. ACM CoNEXT*, 2011.

[18] R. Koodli and R. Ravikanth, "One-way loss pattern sample metrics," RFC 3357, Aug. 2002.

[19] J. Sommers, P. Barford, and W. Willinger, "Laboratory-based calibration of available bandwidth estimation tools," *Microprocess. Microsyst.*, vol. 31, no. 4, pp. 222–235, 2007.

[20] N. Hu, L. Li, Z. M. Mao, P. Steenkiste, and J. Wang, "Locating internet bottlenecks: Algorithms, measurements, and implications," in *Proc. ACM SIGCOMM*, 2004.

[21] M. Allman, V. Paxson, and E. Blanton, "Rfc5681: Tcp congestion control," 2009.

[22] B. Everitt, S. Landau, M. Leese, and D. Stahl, *Cluster Analysis*, 5th ed. Wiley, 2011.

[23] B. Brodsky and B. Darkhovsky, *Non-Parametric Statistical Diagnosis Problems and Methods*. Kluwer Academic Publishers, 2000.

[24] Z. Durumeric, E. Wustrow, and J. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *Proc. 22nd USENIX Security Symposium*, 2013, pp. 605–619.

[25] M. Tozal and K. Sarac, "Tracenet: An internet topology data collector," in *Proc. ACM IMC*, 2010.

[26] J. Li, T. Ehrenkranz, and P. Elliott, "Buddyguard: A buddy system for fast and reliable detection of ip prefix anomalies," in *Proc. IEEE ICNP*, 2012.

[27] E. Katz-Bassett, H. Madhyastha, V. Adhikari, C. Scott, J. Sherry, P. Wesep, A. Krishnamurthy, and T. Anderson, "Reverse traceroute," in *Proc. USENIX NSDI*, 2010.

[28] A. Khan, T. Kwon, H. Kim, and Y. Choi, "As-level topology collection through looking glass servers," in *Proc. ACM IMC*, 2013.

[29] J. Padhye and S. Floyd, "Identifying the TCP behavior of web servers," in *Proc. ACM SIGCOMM*, 2001.

[30] Y. Lin, R. Hwang, and F. Baker, *Computer Networks: An Open Source Approach*. McGraw-Hill, 2011.

[31] Netfilter, http://www.netfilter.org.

[32] Planetlab, https://www.planet-lab.org.

[33] A. Dainotti, A. Botta, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, 2012.

[34] htop, http://hisham.hm/htop/.

[35] M. Thottan and C. Ji, "Anomaly detection in ip networks," *IEEE Trans. on Signal Processing*, vol. 51, no. 8, 2003.

[36] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the dos and ddos problems," *ACM Computing Surveys (CSUR)*, vol. 39, no. 1, 2007.

[37] G. Loukas and G. Öke, "Protection against denial of service attacks: a survey," *The Computer Journal*, vol. 53, no. 7, 2010.

[38] S. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, 2013.

[39] M. Bhuyan, H. Kashyap, D. Bhattacharyya, and J. Kalita, "Detecting distributed denial of service attacks: Methods, tools and future directions," *The Computer Journal*, Mar. 2013.

[40] M. Bhuyan, D. Bhattacharyya, and J. Kalita, "Network anomaly detection: Methods, systems and tools," *Communications Surveys & Tutorials*, 2013.

[41] L. Quan, J. Heidemann, and Y. Pradkin, "Trinocular: Understanding internet reliability through adaptive probing," in *Proc. ACM SIGCOMM*, 2013.

[42] Y. Zhang, Z. Mao, and M. Zhang, "Detecting traffic differentiation in backbone ISPs with NetPolice," in *Proc. ACM IMC*, 2009.

[43] E. Katz-Bassett, H. Madhyastha, J. John, A. Krishnamurthy, D. Wetherall, and T. Anderson, "Studying black holes in the internet with hubble," in *Proc. USENIX NSDI*, 2008.

[44] Y. Liu, X. Luo, R. Chang, and J. Su, "Characterizing Inter-Domain Rerouting by Betweenness Centrality after Disruptive Events," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 5, 2013.

[45] W. Fok, X. Luo, R. Mok, W. Li, Y. Liu, E. Chan, and R. Chang, "Monoscope: Automating network faults diagnosis based on active measurements," in *Proc. IFIP/IEEE IM*, 2013.

[46] E. Chan, X. Luo, W. Fok, W. Li, and R. Chang, "Non-cooperative diagnosis of submarine cable faults," in *Proc. PAM*, 2011.

[47] Y. Zhang, Z. Mao, and M. Zhang, "Effective diagnosis of routing disruptions from end systems," in *Proc. USENIX NSDI*, 2008.

[48] X. Luo, L. Xue, C. Shi, Y. Shao, C. Qian, and E. Chan, "On measuring one-way path metrics from a web server," in *Proc. IEEE ICNP*, 2014.

[49] A. Shevtekar and N. Ansari, "A router-based technique to mitigate reduction of quality (roq) attacks," *Computer Networks*, vol. 52, no. 5, 2008.

[50] C. Zhang, Z. Cai, W. Chen, X. Luo, and J. Yin, "Flow level detection and filtering of low-rate DDoS," *Computer Networks*, vol. 56, no. 15, 2012.

[51] C. Chang, S. Lee, B. Lin, and J. Wang, "The taming of the shrew: mitigating low-rate tcp-targeted attack," *IEEE Trans. On Network Service Management*, Mar. 2010.

[52] X. Luo and R. Chang, "Optimizing the pulsing denial-of-service attacks," in *Proc. IEEE DSN*, 2005.

[53] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: recent developments," *Statistical Science*, vol. 19, no. 3, 2004.

[54] L. Ma, T. He, K. Leung, A. Swami, and D. Towsley, "Identifiability of link metrics based on end-to-end path measurements," in *Proc. ACM IMC*, 2013.

[55] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "NetDiagnoser: troubleshooting network unreachabilities using end-to-end probes and routing data," in *Proc. ACM CoNEXT*, 2007.

[56] S. Zarifzadeh, M. Gowdagere, and C. Dovrolis, "Range tomography: combining the practicality of boolean tomography with the resolution of analog tomography," in *Proc. ACM IMC*, 2012.

[57] H. Nguyen and P. Thiran, "The boolean solution to the congested ip link location problem:theory and practice," in *Proc. IEEE INFOCOM*, 2007.

[58] N. Duffield, P. Avenue, and F. Park, "Network tomography of binary network performance characteristics," *IEEE Trans. Information Theory*, vol. 52, no. 12, 2006.

[59] Q. Zheng and G. Cao, "Minimizing probing cost and achieving identifiability in probe based network link monitoring," *IEEE Trans. Computers*, vol. 62, no. 3, 2013.

[60] Y. Zhao, Y. Chen, and D. Bindel, "Towards unbiased end-to-end network diagnosis," *IEEE/ACM Transaction on Networking*, vol. 17, no. 6, 2009.