doi:10.1093/comjnl/bxl026

An Efficient Distributed Algorithm to Identify and Traceback DDoS Traffic

T. Y. WONG*, K. T. LAW, JOHN C. S. LUI AND M. H. WONG

Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong
**Corresponding author: tywong@cse.cuhk.edu.hk*

Distributed denial-of-service attack is one of the most pressing security problems that the Internet community needs to address. Two major requirements for effective traceback are (i) to quickly and accurately locate potential attackers and (ii) to filter attack packets so that a host can resume the normal service to legitimate clients. Most of the existing IP traceback techniques focus on tracking the location of attackers after-the-fact. In this work, we provide an efficient methodology for locating potential attackers who employ the flood-based attack. We propose a *distributed algorithm* so that a set of routers can correctly (in a distributed sense) gather statistics in a coordinated fashion and that a victim site can deduce the local traffic intensities of all these participating routers. We prove the correctness of our distributed algorithm, and given the collected statistics, we provide a method for the victim site to locate attackers who sent out dominating flows of packets. The proposed distributed traceback methodology can also complement and leverage on the existing ICMP traceback so that a more efficient and accurate traceback can be obtained. We carry out simulations to illustrate that the proposed methodology can locate the attackers in a short period of time. Moreover, the applications as well as the limitations of the proposed methodology are covered. We believe this work also provides the theoretical foundation on how to correctly and accurately perform distributed measurement and traffic estimation on the Internet.

Keywords: Security and protection, programmable routers, distributed algorithm

Received 30 November 2004; revised 20 February 2006

1. INTRODUCTION

The emergence of the Internet as a pervasive form of communication has led to the recent enormous deployment of e-business and information distribution services. However, the success of the Internet also attracts malicious attackers to abuse system resources and exposes the inherent security problems. Distributed denial-of-service (DDoS) attack is one of the most pressing problems on the Internet. In recent years, well-known commercial sites such as Yahoo, Amazon and eBay were being attacked and were out of service for many hours due to the DDoS attack on February 2000 [1]. Since then, DDoS attacks have increased in size, frequency, sophistication and severity.

Two basic requirements to counter DDoS attack are *traceback property* and *filtering property*. Traceback property refers to the capability to determine the possible locations of the attackers. Filtering property refers to the capability to eliminate the attacking traffic from the victim site so that regular services can be maintained. If the attackers are using non-spoofed packets in their attacks, there is no traceback problem because the source addresses in the packets are valid and can be traced easily. However, the attackers often use fake or spoofed IP source addresses in their attack packets. Real attacks, like the TCP SYN attack, are using spoofed packets in order to bring about the desired damage. Also, compromised machines are precious resources for the hackers, and hackers like to protect these resources so others will not discover that these hosts are compromised. Therefore, IP spoofing is often used. Moreover, because of the stateless nature of the Internet, it is a difficult task to determine or to trace the source of these attackers' packets and thereby locate the potential locations of these attackers.

In this paper, we present a *distributed* approach to effectively traceback the location of potential *flood-based* attack sources. Our proposed approach is twofold. Firstly, our approach is grounded on the programmable router architecture [2, 3, 4, 5] wherein participating routers can collaboratively collect traffic statistics to a victim site. The statistical information will be forwarded to a victim site (or any processing node in a distributed system). The victim site can then

(i) construct the attack graph, which depicts the network paths taken by all received packets at the victim, and

(ii) accurately determine the magnitudes or intensities of the local traffic, which was generated from the local administrative domain of each participating router.

Secondly, upon determining the intensity of *local traffic* of each participating router, the victim site can then determine the amount of traffics from each participating router which arrived at the victim site within a measurement interval. Based on this information, the victim site can determine a subset of attacking routers whose workload consume a large percentage of the victim's resource.

The contributions of our work are as follows:

- An effective distributed traceback methodology to determine the local traffics of participating routers.
- The local traffics of all routers are determined at the same logical time without requiring any global clock or global synchronization from each participating router.
- The victim can efficiently locate the attackers who contribute large attack flows.

The rest of the paper is organized as follows. In Section 2, we formally present the traceback problem and present the network setting wherein we perform the distributed traceback. We also give an example to illustrate why one needs a distributed algorithm to carefully record the local state of each participating router so as to achieve the notion of correctness. In Section 3, we present the distributed algorithm to correctly record the state of each participating router. In Section 4, we present the criteria and the algorithm to effectively determine the subset of routers whose local traffics consume a large portion of the victim's resource within a given time interval. In Section 5, we carry out *NS*-2 simulation to illustrate the effectiveness of our distributed traceback methodology. Implementation issues are discussed in Section 6. In Section 7, we introduce possible applications and limitations of our proposed approach. Related work is given in Section 8, and the conclusion is given in Section 9.

2. OVERVIEW AND PROBLEM DEFINITION

In this section, we first present the overview of our approach, then we present a network model and some of its important components. We define three important concepts, namely, the *transit traffic*, the *local traffic* and the *outgoing traffic* of a participating router. We also illustrate why one needs a distributed algorithm to correctly perform the traceback under a DDoS attack.

2.1. Overview

One way to eliminate the detrimental effect of the flood-based DDoS attack is to trace the location of the attacker and to filter out all the malicious packets leaving that host. Since the attacker is sending a huge amount of packets compared with those of the normal users, one can easily notice the large portion of traffic from the attacker on the victim side through a *traffic intensity measuring mechanism*.

However, this approach is infeasible since the attackers are usually spoofing the source address of the malicious packets. One can hardly measure the traffic intensity of a particular host based on the source addresses of the outgoing packets. Alternatively, we suggest measuring the intensity of the outgoing traffic towards the victim on the routers. Certainly, this scheme neither measures the traffic intensity of an individual user nor traceback to a particular attacker. Nevertheless, it aims to identify a number of routers which have high volume of outgoing traffic towards the victim site. This indicates that the origins of the attack are from the *domains* of those routers.

In order to measure and collect the traffic intensities from the routers that are participating in the DDoS attack traceback mechanism, we propose a novel approach by applying the snapshot algorithm suggested in [6]. The snapshot approach provides means to coordinate all the participating routers in the traffic measurement and the data collecting procedures. It also provides a way to measure the traffic intensity *correctly*. The advantages of our approach are (i) easy to implement without a large modification of the routers and (ii) fast; the approach requires only a few seconds in measuring the traffic intensities of the router.

2.2. Problem definition

Let us first define our network model. In Figure 1, a directed acyclic graph (DAG) rooted at $\mathcal{V}$ represents a network topology, and the root node $\mathcal{V}$ represents a victim site. The DAG is composed of routers and local area networks (LANs). For the simplicity of illustration, the DAG only shows the network components that are participating in transmitting and forwarding traffics to the victim site $\mathcal{V}$. Let R_i be an upstream router of $\mathcal{V}$ and the DAG a map of all routers which forward traffics to $\mathcal{V}$.

A LAN contains a number of end hosts which include some legitimate clients of $\mathcal{V}$ and possibly some attackers of $\mathcal{V}$. The traffics sending to $\mathcal{V}$ generated by the clients and the attackers are forwarded by routers. For example, in Figure 1, router R_1 serves as a gateway of LAN_0 and LAN_1, and these two LANs are regarded as the '*local administrative domain*' (or '*domain*' in short) of R_1. A router is responsible for forwarding traffics generated from its domain, as well as traffics generated from the domains of its '*upstream routers*'.

For example, in Figure 1, routers R_3, R_4 and R_5 are considered as the upstream routers of R_1. Particularly, routers R_3 and R_4 are regarded as the '*immediate upstream routers*' of R_1. We say that a router is a *leaf* router if it is not connected to any upstream routers, for instance, R_2, R_3 and R_5 in Figure 1. Other routers such as R_1 and R_4 are called *internal* routers. In reality, a router in this model represents the *border router*

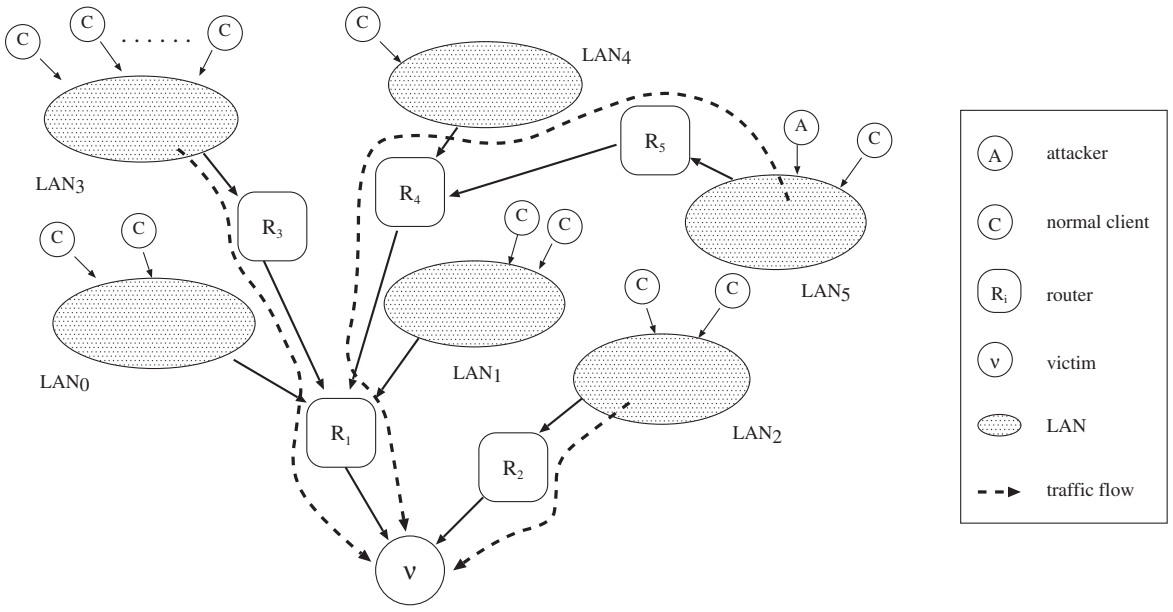

FIGURE 1. An example network topology.

of an ISP or an AS while the corresponding LAN represents the *domain* of the ISP/AS. Throughout this paper, we let $\mathbb{U}(R_i)$ be the set of upstream routers of R_i, $\mathcal{U}(R_i)$ be the set of immediate upstream routers of R_i, $\mathbb{D}(R_i)$ be the set of downstream routers of R_i and $\mathcal{D}(R_i)$ be the set of immediate downstream routers of R_i.

In our work, we classify three types of traffics: they are the *transit traffic* the *local traffic* and the *outgoing traffic*. The transit traffic of R_i is the traffic forwarded from the immediate upstream routers of R_i while the local traffic of R_i represents the traffic generated from the local administrative domain of R_i. Eventually, the outgoing traffic of R_i is the sum of the transit traffic and the local traffic of R_i. To illustrate, let us consider the following example using Figure 1. Part of the traffic to $\mathcal{V}$ was generated in LAN_5, and the packets have to pass through routers R_5, R_4 and R_1 before reaching $\mathcal{V}$. The traffic from LAN_5 is considered as the *transit traffic* of router R_4. On the other hand, the clients in LAN_4 also generate traffic to $\mathcal{V}$ and this traffic is considered as the *local traffic* of R_4. The union of these two streams of traffics generated in LAN_4 and LAN_5 is considered as the outgoing traffics of R_4.

We assume that each router maintains a counter which records the *accumulative* (for the ease of presentation, let us ignore the counter wraparound problem. It can be resolved by a distributed coordinated reset) volume of the outgoing traffic (in units of packet) towards the victim site $\mathcal{V}$. To elaborate the distributed algorithm, we formally define the following concept:

DEFINITION 2.1. *The 'local traffic' of R_i is the number of packets which are destined for the victim site $\mathcal{V}$ and these packets are generated within the local administrative domain of R_i between $[t_1, t_2]$, where $t_2 > t_1$. We denote the 'local traffic' of R_i between the time interval $[t_1, t_2]$ as $L_i(t_1, t_2)$.*

DEFINITION 2.2. *The 'outgoing traffic' counter of R_i at time t records the accumulative number of packets, which are destined for the victim site $\mathcal{V}$ up to time t. We denote the counter value of the 'outgoing traffic' of R_i at time t as $C_i(t)$.*

We define an attacker and his/her behavior as follows. An attacker is a host which is sending *high volume* of traffics towards the victim site within a period of time (usually within seconds) and thereby consumes a large portion of the victim's resource. An attacker may generate any kinds of packets with spoofed source addresses. In an attack scenario, there are usually multiple attackers and they may conspire.

In the following, we introduce our approach in a formal manner. Let $C_i(t)$ be the counter value of the outgoing traffic of router R_i at time t and let $\mathcal{U}(R_i)$ be a set of *immediate upstream routers* of R_i. The accumulative local traffic $N_i(t)$ of router R_i at time t is

$$N_i(t) = \begin{cases} C_i(t) & R_i \text{ is a leaf;} \\ C_i(t) - \sum_{R_j \in \mathcal{U}(R_i)} C_j(t) & \text{otherwise.} \end{cases} \quad (1)$$

Let $L_i(t_1, t_2)$ represent the local traffic generated by the router R_i within the time interval $[t_1, t_2]$:

$$L_i(t_1, t_2) = N_i(t_2) - N_i(t_1). \quad (2)$$

The implication of these Equations (1) and (2) is that, by using the outgoing traffic counters, one can deduce the accumulative local traffic to the victim site $\mathcal{V}$ for every router by Equation (1). Then, by taking the difference of these

TABLE 1. Computation of the accumulative local traffic in Example A by using Equation (1).

	Outgoing traffic counter at time t				
Time	$C_5(t)$	$C_4(t)$	$C_3(t)$	$C_2(t)$	$C_1(t)$
$t = t_1$	20,000	35,000	5000	10,000	65,000
$t = t_2$	50,000	66,000	6000	11,000	99,000

accumulative local traffics between two different time instants, one can obtain the local traffic to a victim $\mathcal{V}$ for every router within the measurement interval by Equation (2).

We describe how the victim can initiate the traceback process in the following. When $\mathcal{V}$ receives a huge amount of traffic that exceeds its pre-defined threshold of traffic loading, $\mathcal{V}$ declares that it is under a DDoS attack and begins the traceback procedure. $\mathcal{V}$ signals all routers[1] to read their outgoing traffic counters. In order to determine the local traffic within a time interval $[t_1, t_2]$, $\mathcal{V}$ needs to send the '*counter reading signals*' to all participating routers *twice*, one at time t_1 and the other at time t_2. Then, each router takes the counter value of its outgoing traffic counter and sends this information back to the victim accordingly. Eventually, after $\mathcal{V}$ has collected these two sets of data from the participating routers, $\mathcal{V}$ computes the local traffic generated from each domain within $[t_1, t_2]$ by Equations (1) and (2). By comparing the intensities of the local traffics among the participating routers, one can determine the locations of the attackers.

To illustrate this traceback process, consider the following simple but illustrative example using the network topology of Figure 1. We call this *Example A*. There is only one attacker located in LAN_5. The attacker launches a DoS attack on the victim site $\mathcal{V}$, and $\mathcal{V}$ initiates the traceback procedure to determine the location of the attacker. For simplicity of illustration, we assume that the initial values of all outgoing traffic counters of the routers are zero (i.e. $C_i(0) = 0$ for all R_i in the network topology). The counter values of all five routers' outgoing traffic are taken at time instants t_1 and t_2. Table 1 depicts the outgoing traffic $C_i(t)$ at time instants t_1 and t_2 for all routers, and the values of all routers' accumulative local traffic $N_i(t)$ at time instants t_1 and t_2.

The local traffic $L_i(t_1, t_2)$ of each router generated within $[t_1, t_2]$ is shown in Table 2 wherein the computation is based on Equation (2). Comparing the intensities of the local traffic of these five routers within the interval $[t_1, t_2]$, one can deduce that the domain of router R_5 is the location of the attacker. Note that this type of distributed counter reading procedure does not require the router to examine the source IP address of each packet.

[1] In reality, only those routers within the attack graph of $\mathcal{V}$ need to participate in the distributed traceback. We will illustrate this in Section 6.

TABLE 2. Computation of $L_i(t_1, t_2)$: the local traffic within $[t_1,t_2]$ in Example A by using Equation (2).

	Local traffic from t_1 to t_2				
	R_5	R_4	R_3	R_2	R_1
$L_i(t_1, t_2)$	**30,000**	1000	1000	1000	2000

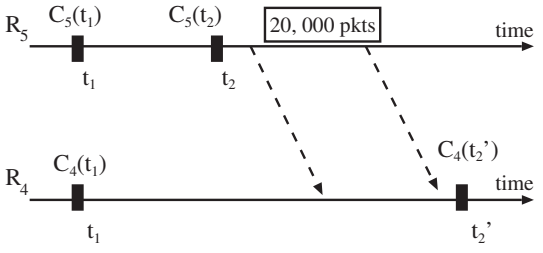

FIGURE 2. Asynchronous reading of outgoing traffic counters in Example B.

By using this traffic measurement, one can easily jump to the conclusion that a DDoS traceback is an easy task. However, we will show that there are some *deficiencies* in this distributed counter reading approach. The major problem is that Equations (1) and (2) are only correct *if the network has a global clock and all routers can perform synchronous reading of their respective outgoing traffic counters*. Let us consider *Example A* again but with asynchronous reading of the counters, and we call this *Example B*.

Figure 2 depicts the scenario of *Example B*. A black rectangle in the figure represents the time instant at which the outgoing traffic counter of a router is read. Since the second outgoing traffic counters for R_4 and R_5 are not read simultaneously, some packets from R_5 sending to $\mathcal{V}$ are not recorded by R_5 but are recorded by R_4. To illustrate it numerically, $C_5(t_2)$ becomes 30,000 instead of 50,000. Thus, $L_5(t_1, t_2)$ becomes 10,000 while $L_4(t_1, t_2')$ becomes 21,000. One can observe that asynchronous reading of the counters will mislead the victim site $\mathcal{V}$ to conclude that the domain of router R_4 is the location of the attacker. In the next section, we present a complete distributed algorithm to precisely measure the local traffics of all routers without a global clock. The derived local traffic intensities can help us to determine the locations of the attackers.

3. DISTRIBUTED ALGORITHM

In this section, we present the complete distributed algorithm to measure the local traffic of every router. We first define the notion of '*correctness*' for measuring the local traffic of each participating router and demonstrate how one can effectively achieve the required correctness. Besides the

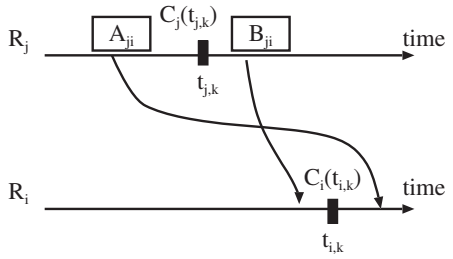

FIGURE 3. Correct accumulative local traffic without clock synchronization.

general proof for the correctness of the proposed distributed algorithm, we also use an example to illustrate the effectiveness of this distributed algorithm.

In the previous section, we have used an example to illustrate that a naive or straightforward manner of reading the outgoing traffic counters can lead to an erroneous conclusion (which concludes that router R_4, instead of R_5, is the location of the attacker). The reason for this erroneous conclusion is that the outgoing traffic counters of the immediate upstream routers of R_4 are not recorded correctly. To illustrate the notion of correctness, Figure 3 illustrates a timing diagram with two routers R_i and R_j, where R_j is an immediate upstream router of R_i. The black rectangle in the figure represents the time instant at which the outgoing traffic counter of a router R_i is read, and we let this time instant be $t_{i,k}$, where $k \in [1, 2]$ represents whether the reading is taken for the first or the second time. We assume that Figure 3 is illustrating the reading of the outgoing traffic counter for the k-th time. Let $C_j(t_{j,k})$ be the value of R_js outgoing traffic counter at time $t_{j,k}$ and let $C_i(t_{i,k})$ be the value of R_is outgoing traffic counter at time $t_{i,k}$. The A_{ji} block in the figure represents a sequence of packets that are sent to $\mathcal{V}$ through R_j before $t_{j,k}$ but are received by R_i after $t_{i,k}$. Correspondingly, the B_{ji} block represents a sequence of packets that are sent to $\mathcal{V}$ by R_j after $t_{j,k}$ but are received by R_i before time $t_{i,k}$. In other words, all packets in A_{ji} are recorded in $C_j(t_{j,k})$ but are not recorded in $C_i(t_{i,k})$; while all packets in B_{ji} are not recorded in $C_j(t_{j,k})$ but are recorded in $C_i(t_{i,k})$. The mis-counting of these packets will lead to an erroneous conclusion.

Let the accumulative local traffic of router R_i at time $t_{i,k}$ be $N_i(t_{i,k})$. The accumulative local traffic of R_i at time $t_{i,k}$ is the *difference* between the outgoing traffic of R_i at $t_{i,k}$ and the transit traffic received by R_i at $t_{i,k}$. Hence

$$N_i(t_{i,k}) = C_i(t_{i,k}) - (\text{transit traffic received by } R_i \text{ at } t_{i,k}).$$

The transit traffic received by R_i at time $t_{i,k}$ is the outgoing traffic sent from all immediate upstream routers of R_i. On one hand, since packets in A_{ji} are not received by R_i at $t_{i,k}$, but are recorded by R_j at $t_{j,k}$, one needs to reduce this traffic workload from $C_j(t_{j,k})$. On the other hand, packets in B_{ji} are received by R_i at $t_{i,k}$, but are not recorded by R_j at $t_{j,k}$. Thus, one needs to include this traffic workload in $C_j(t_{j,k})$. Therefore, the correct accounting of the accumulative local traffic from router R_i is defined as follows:

$$N_i(t_{i,k}) = C_i(t_{i,k}) - \sum_{R_j \in \mathcal{U}(R_i)} (C_j(t_{j,k}) - A_{ji} + B_{ji}).$$

Let us apply the above equation back to *Example B* illustrated in Figure 2. The accumulative local traffic $N_4(t_{4,2})$ becomes

$$\begin{aligned} N_4(t_{4,2}) &= C_4(t_{4,2}) - (C_5(t_{5,2}) - A_{54} + B_{54}) \\ &= 66{,}000 - (30{,}000 - 0 + 20{,}000) \\ &= 16{,}000. \\ \therefore L_4(t_{4,1}, t_{4,2}) &= 16{,}000 - 15{,}000 = 1000. \end{aligned}$$

Hence, one can conclude that the attacker is in the domain of R_5. In the following subsection, we present an efficient distributed algorithm to measure the two sequences of packets A_{ji} and B_{ji} correctly so as to satisfy the correctness criteria stated above.

3.1. Measuring the correct local traffic

We make use of the result in [6] to collect all outgoing traffic counters in a coordinated manner and, at the same time, to determine the values of A_{ji} and B_{ji}. We call this the *snapshot* algorithm. There are three main components in the snapshot algorithm, namely (i) the *marker*, (ii) the *local state* of a participating router or the victim site and (iii) the *channel state*. These three components have the following functionalities under our DDoS application.

(i) *Marker.* The marker is a special packet with a special header or a special header entry. The marker is initially sent by $\mathcal{V}$ to all its neighboring routers. The functionality of the marker is to facilitate all participating routers to record their local states and to derive the corresponding channel states.

(ii) *Local state*. Both the router and the victim site have their own local states. For a participating router, say the router R_i, the local state at time t corresponds to the value of its outgoing traffic counter $C_i(t)$. However, the victim site $\mathcal{V}$ does not have an outgoing traffic counter. Instead, the local state of the victim site $\mathcal{V}$ refers to the accumulative number of packets that $\mathcal{V}$ has received by time t, i.e. the aggregated traffic destined for $\mathcal{V}$ from the domains of all participating routers. We denote this accumulative incoming traffic to $\mathcal{V}$ at time t as $T_{\mathcal{V}}(t)$. To find the aggregated incoming traffic sent to $\mathcal{V}$ within the interval $[t_1, t_2]$, one can perform

$$I_{\mathcal{V}}(t_1, t_2) = T_{\mathcal{V}}(t_2) - T_{\mathcal{V}}(t_1). \quad (3)$$

(iii) *Channel state.* This corresponds to the number of packets that are received by a router after the router

ALGORITHM 1. The snapshot algorithm.

Algorithm initialization:
$\mathcal{V}$ records the value of incoming traffic at t as $T\mathcal{V}(t)$;
For (each link $Link_{vk}$ that connects $\mathcal{V}$ to its neighboring router R_k) {
$\mathcal{V}$ sends a marker along $Link_{vk}$ and starts recording the number of packets received from $Link_{kv}$;
}
Marker-sending and marker-receiving rules:
For the victim site $\mathcal{V}$:
If ($\mathcal{V}$ has received a marker from a $Link_{kv}$ at time t'){
$\mathcal{V}$ stops recording the number of packets received from $Link_{kv}$ and stores the value as the channel state $H_{kv}(t, t')$;
}
For each participating router R_i:
If (R_i has received a marker from $Link_{ji}$ at time $t_{i,k}$ **and** R_i has not recorded its local state) {
R_i records the value of the outgoing traffic counter at time t as $C_i(t)$;
R_i sets the channel state $H_{ji}(t_{i,k}, t_{i,k}{}^j)$ as zero;
For (each link $Link_{ik}$ that connects R_i to its neighboring router R_k) {
R_i sends a marker along $Link_{ik}$;
R_i starts recording the number of packets received from $Link_{ki}$, except $Link_{ji}$;
}
If (R_i has received a marker from $Link_{ji}$ at time $t_{i,k}{}^j$ **and** R_i has already recorded its local state) {
R_i stops recording the number of packets from $Link_{ji}$ and stores the value as $H_{ji}(t_{i,k}, t_{i,k}{}^j)$;
}
Termination:
For each participating router R_i *in the network topology*:
If (R_i has recorded local state and has finished recording channel states for all incoming links) {
R_i sends the snapshot data (i.e. its local state and all its channel states) to $\mathcal{V}$;
R_i terminates;
}
For the victim site $\mathcal{V}$:
If ($\mathcal{V}$ has recorded its incoming traffic and has finished recording all channel states for all its incoming links **and** has received the snapshot data sending from all participating routers) {
$\mathcal{V}$ terminates;
}

records its own local state but before that router receives the marker along that link. Its role will be thoroughly discussed later.

The snapshot algorithm assumes that the packet delivery process is in the order sent (or FIFO). As two adjacent routers are connected by a communication link or in the same LAN, the delivery order of the packets can be preserved under this kind of physical connection. On the other hand, since the channel state is measuring the number of packets forwarding from an upstream router, one can notice that this may not work as the router has to examine the source address of the incoming packets. Nevertheless, the router can rely on which interface a packet is coming from or the router can refer to the level two address (e.g. MAC address in Ethernet) of a packet in order to distinguish which router or channel that packet is coming from. This methodology does not take the risk that the attackers are sending spoof packets because we are only interested in from which upstream router a packet is coming. Also, it is futile for the attacker to spoof the level two source address of a packet because when the packet is routed through a routing device, the level two address is usually altered and set as the hardware address of that routing device.

Before we formally present the snapshot algorithm, let us define the following notations. Let R_i and R_j be two adjacent routers connected by two uni-directional links, namely $Link_{ij}$ and $Link_{ji}$. $Link_{ij}$ carries traffic that is from R_i to R_j while $Link_{ji}$ carries traffic from R_j to R_i. Let the time instant that R_i records its local state be $t_{i,k}$, and let the time instant that R_i receives a marker from $Link_{ji}$ after it has recorded its local state be $t_{i,k}{}^j$ (if the marker arrives before R_i records its local state, then that time instant is $t_{i,k}$. See Algorithm 1 for details). Let $H_{ji}(t_{i,k}, t_{i,k}{}^j)$ be the channel state of $Link_{ji}$, which is the number of packets received by R_j from R_i after $t_{i,k}$ and before $t_{i,k}{}^j$. The following pseudo-code shows the outline of the snapshot algorithm.

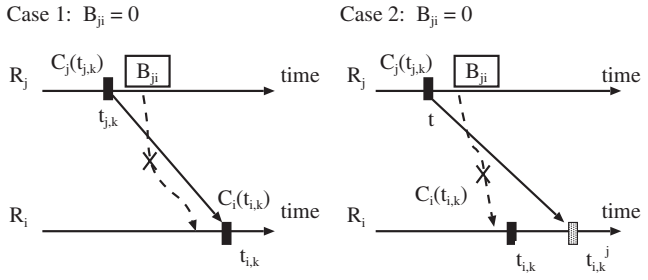

FIGURE 4. $B_{ij} = 0$ under all circumstances.

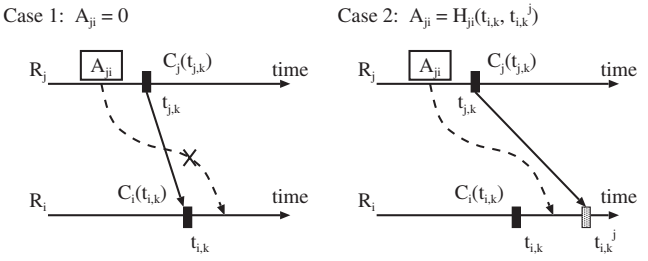

FIGURE 5. A_{ji} is the channel state.

The invocation of the snapshot algorithm occurs when the victim site $\mathcal{V}$ acknowledges that it is under a DDoS attack; for instance, $\mathcal{V}$ finds that the amount of incoming traffic has exceeded a pre-defined threshold. In the beginning of the algorithm, $\mathcal{V}$ sends markers to its neighboring routers along its outgoing links. These routers will then send markers to all its neighboring routers according to the above pseudo-code. Eventually, all participating routers in the attack graph will process the marker for the snapshot algorithm. This is a series of coordinated actions among the routers by the sending and receiving of markers. The *properties* of these actions are (i) it guarantees that $B_{ji} = 0$ and (ii) the measured value $H_{ji}(t_{i,k}, t_{i,k}{}^{j})$ is equivalent to A_{ji}. After a router has finished recording its local and channel states for all its incoming links, it sends these information to $\mathcal{V}$. The algorithm terminates after $\mathcal{V}$ has finished recording its local state and channel states and has received the local states and the channel states from all participating routers. The following lemma proves the mentioned properties of the algorithm.

LEMMA 1. *For any two adjacent routers R_i and R_j, which are connected by the $Link_{ji}$, the snapshot algorithm guarantees that $B_{ji} = 0$ and correctly measures A_{ji} as the channel state of $Link_{ji}$.*

Proof. We first prove $B_{ji} = 0$, then prove that A_{ji} is the channel state $H_{ji}(t_{i,k}, t_{i,k}{}^{j})$ of $Link_{ji}$. In Figures 4 and 5, the black rectangle represents the time instant at which the value of the outgoing traffic counter of a router is recorded. The shaded rectangle represents the time instant at which a marker arrives at R_i after the value of the outgoing traffic counter of R_i is recorded. The dotted line is the transmission of the sequence of packets A_{ji} or B_{ji} from R_j to R_i while the solid line represents the transmission of the marker from R_j to R_i. For both Figures 4 and 5, case 1 corresponds to the scenario that R_i records its local state $C_i(t_{i,k})$ because it receives the marker from R_j along $Link_{ji}$. For case 2, R_i has already recorded its local state $C_i(t_{i,k})$ before the arrival of the marker from R_j along $Link_{ji}$.

In Figure 4, we show that $B_{ji} = 0$ under all circumstances. When the router R_j records its local state $C_j(t_{j,k})$, it sends markers to all its outgoing channels. Since $Link_{ji}$ is FIFO, all packets in B_{ji} cannot reach R_i before the marker arrives at R_i. Therefore, B_{ji} is equal to zero in both cases.

We now prove that A_{ji} is equivalent to the channel state of $Link_{ji}$. Recall that A_{ji} represents a sequence of packets that are sent to $\mathcal{V}$ by R_j before $t_{j,k}$ but are received by R_i after $t_{i,k}$. When the router R_j records its local state $C_j(t_{j,k})$, it sends markers to all its outgoing channels. There are two cases to consider:

(i) In case 1 of Figure 5, all packets of A_{ji} must not be able to reach R_i after $t_{i,k}$ as $Link_{ji}$ is FIFO. Therefore, $A_{ji} = 0$.

(ii) In case 2 of Figure 5, all packets of A_{ji} reach R_i before $t_{i,k}$ due to the FIFO property of $Link_{ji}$. Since R_i records its local state $C_i(t_{i,k})$ at time $t_{i,k}$ and starts counting the number of packets arrived along $Link_{ji}$, R_i stops counting the number of packets arrived along $Link_{ji}$ when it receives the marker from R_j at t'. The packets arrived between $[t_{i,k}, t_{i,k}{}^{j}]$ is the channel state $H_{ji}(t_{i,k}, t_{i,k}{}^{j})$ of $Link_{ji}$. Based on the result in case 1, the packets of A_{ji} cannot reach R_i after the time $t_{i,k}$. Thus, by the definition of A_{ji}, A_{ji} is equal to channel state $H_{ji}(t_{i,k}, t_{i,k}{}^{j})$ of $Link_{ji}$. □

Finally, by Lemma 1, the equation for calculating the accumulative local traffic $N_i(t_{i,k})$ is as follows:

$$N_i(t_{i,k}) = C_i(t_{i,k}) - \sum_{R_j \in \mathcal{U}(R_i)} (C_j(t_{j,k}) - H_{ji}(t_{i,k}, t_{i,k}{}^{j})). \quad (4)$$

In conclusion, the aim of the traceback algorithm is to measure the local traffic of every router within the time interval $[t_1, t_2]$. The victim site $\mathcal{V}$ initiates the snapshot algorithm *twice*: once at time t_1 and another at time t_2. Based on the local states and the channel states received from all routers, the victim site calculates the consistent local traffic counters by applying Equation (4). In turn, the victim calculates the local traffic intensity of router R_i by Equation (2).

3.2. Illustrating the DDoS traceback algorithm

In this subsection, we illustrate the DDoS traceback algorithm through an example by using the network topology shown in

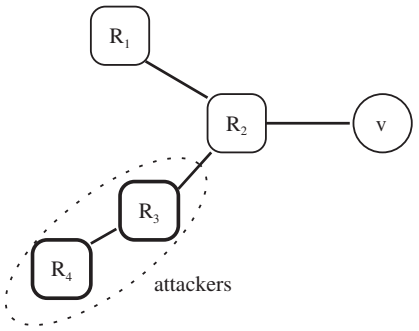

FIGURE 6. A network topology with two attackers who reside in the local domains of R_3 and R_4.

Figure 6. For simplicity, the LANs of the routers are not shown and we assume that all the routers have their own local domains. The attackers are located in the domains of the routers R_3 and R_4 as indicated in the figure.

Figure 7 illustrates how the DDoS traceback algorithm works. A black rectangle represents the time instant that a router R_i records the value of its outgoing traffic counter, and we denote this time instant as $t_{i,k}$ where k represents the k-th instance of the snapshot algorithm, $k \in [1, 2]$. In addition, the corresponding value of the outgoing traffic counter is shown beside the black rectangle. For the victim site $\mathcal{V}$, the black rectangle represents the time instant that the victim site $\mathcal{V}$ records the number of incoming packets. We denote this time instant as t_k where $k \in [1, 2]$. For the simplicity of illustration, we assume that the initial values of the outgoing traffic counters of all routers and the initial value of the accumulative incoming traffic of the victim site $\mathcal{V}$ are zero. On the other hand, a shaded rectangle represents the instant that a router or the victim stops recording a channel state, and we denote this time instant as $t_{i,k}{}^{j}$ (the superscript j means the marker coming from $Link_{ji}$). Similar to the presentation of the local state, the value of the channel state is shown beside the shaded rectangle. This figure also shows the time instant that the domain of a router transmits packets to $\mathcal{V}$. A sequence of normal packets is represented by a white circle in the figure, and each white circle represents 10 packets (shown as n_a, n_b, n_c and n_d in the figure). On the other hand, a sequence of malicious packets from the attackers is represented by a black circle, and each black circle represents 100 packets (m_a and m_b in the figure).

Based on the values shown in Figure 7, one can apply Equation (4) to calculate the accumulative local traffic of the routers and the victim site, and the corresponding results are shown in Table 3. In order to have a clearer illustration, we show the calculation of the accumulative local traffic $N_2(t_{2,1})$ of R_2 as an example. Referring to Figure 6, the immediate upstream routers of R_2 are R_1 and R_3. Then, we apply Equation (4) as follows:

$$\begin{aligned} N_2(t_{2,1}) &= C_2(t_{2,1}) - (C_1(t_{1,1}) - H_{12}(t_{2,1}, t_{2,1}{}^{1})) \\ &\quad - (C_3(t_{3,1}) - H_{32}(t_{2,1}, t_{2,1}{}^{3})) \\ &= 0 - (10 - 10) - (0 - 0) = 0. \end{aligned}$$

Similarly, one can follow the above procedure to calculate the accumulative local traffic of R_1, R_3 and R_4 in both instances of the snapshot algorithm.

After calculating all accumulative local traffic for both instances of the snapshot algorithm, one can apply Equation (2) to obtain the local traffic intensity of every router. The result is shown in Table 4. The most significant property of the DDoS traceback algorithm is that only the packets that are sent within the two instances of the snapshot algorithm will be recorded. We refer to Figure 7 again in order to illustrate the mentioned property. The figure shows that there are totally 240 packets (packet sequences n_a, n_b, n_c, n_d, m_a and m_b) that have been sent towards the victim site $\mathcal{V}$. However, the sequences of packets n_a and m_b are sent before the routers participate in the traceback algorithm and, thus, these packets are not recorded. Therefore, one can conclude that if an attacker at R_i has sent a massive amount of packets within the two instances $t_{i,1}$, $t_{i,2}$ of the snapshot algorithm, he/she will be discovered by the traceback methodology.

To conclude this section, we presented the DDoS traceback algorithm which records the local traffic of the routers correctly without the requirement of a global clock. Also, a proof is given to show the correctness of the algorithm.

4. INTERPRETING THE TRACEBACK RESULT

In the previous section, we presented the DDoS traceback algorithm which enables the victim site $\mathcal{V}$ to correctly compute the local traffic of every participating router. Referring back to the example in Figure 7, one can observe that the local administrative domain of router R_3 is the location of the attacker by comparing the local traffic intensity of Table 4. Since the attacker in R_3 sent the sequence of malicious packets m_a to the victim site $\mathcal{V}$, the calculated local traffic $L_3(t_{3,1}, t_{3,2})$ of R_3 is significantly larger than those of other routers. Nevertheless, from Figure 7, one can notice that another attacker in R_4 had also sent the sequence of malicious packets m_b to $\mathcal{V}$ before $t_{4,1}$, but these malicious packets are not revealed in Table 4 so that one cannot determine whether R_4 is another location of the attacker. The reason is that the sequence of packets m_b is not sent to $\mathcal{V}$ within $[t_{4,1}, t_{4,2}]$. Therefore, the malicious packets m_b, which are sent by R_4 before $t_{4,1}$ and are received by $\mathcal{V}$ within $[t_1, t_2]$, are not recorded as the local traffic of R_4. This leads to a problem relating to local traffic of the router R_i and incoming traffic of the victim site $\mathcal{V}$.

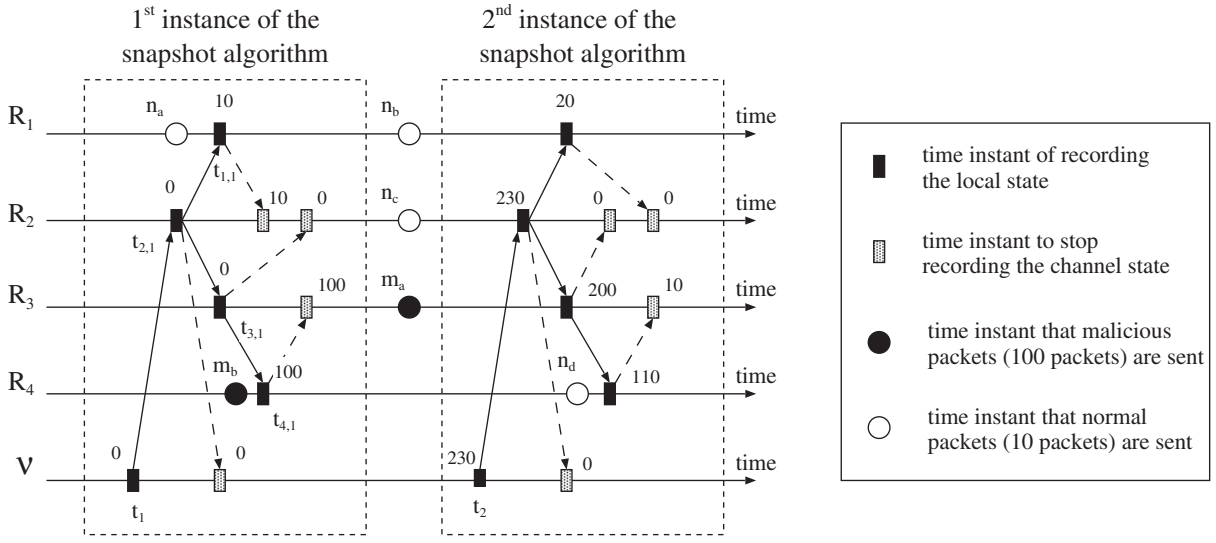

FIGURE 7. A timing diagram that shows the progress of the DDoS traceback algorithm.

TABLE 3. Computation of accumulative local traffic at time $t_{i,1}$ and $t_{i,2}$.

	Accumulative local traffic at time t				
Time	$N_1(t)$	$N_2(t)$	$N_3(t)$	$N_4(t)$	$T_v(t)$
$t = t_{i,1}$	10	0	0	100	0
$t = t_{i,2}$	20	10	100	110	230

TABLE 4. The local traffic intensity only counts the packets in-between the two instances of the snapshot algorithm.

	Local traffic from $t_{i,1}$ to $t_{i,2}$			
	R_1	R_2	R_3	R_4
$L_i(t_{i,1}, t_{i,2})$	10	10	100	10

According to Table 4, the sum of all local traffics is $10 + 10 + 100 + 10 = 130$. But the incoming traffic received by $\mathcal{V}$ within $[t_1, t_2]$ is

$$I_{\mathcal{V}}(t_1, t_2) = T_{\mathcal{V}}(t_2) - T_{\mathcal{V}}(t_1) = 230 - 0 = 230.$$

The total number of packets generated by all routers within the two instances of the snapshot algorithm is not equal to the number of packets received by $\mathcal{V}$ within $[t_1, t_2]$. We call this as the *traffic inequality*. The traffic inequality suggests that the local traffics of the routers may not arrive at $\mathcal{V}$ within $[t_1, t_2]$, and thus one should not only rely on the local traffic of each router to determine the location of the attackers. In the following subsections, we will investigate this problem and we will illustrate a methodology to locate all potential attackers.

4.1. Investigation of the traffic inequality

In this subsection, we present a detailed analysis of the traffic inequality. To start our analysis, we first distinguish packets that are sent from the domain of a router R_i to its downstream routers into three categories based on the time that R_i records its local state. These packets are (i) the *pre-monitoring*, (ii) the *monitoring* and (iii) the *post-monitoring* packets, and they are formally defined in Definition 4.1.

DEFINITION 4.1. *We define pre-monitoring, monitoring and post-monitoring packets with respect to the time that the router R_i records its local state*:

(i) *A packet sent from the local administrative domain of R_i is called a pre-monitoring packet if and only if the packet is sent before R_i records its local state in the first instance of the snapshot algorithm* ($t_{i,1}$).
(ii) *A packet sent from the local administrative domain of R_i is called a monitoring packet if and only if the packet is sent after R_i records its local state in the first instance of the snapshot algorithm* ($t_{i,1}$), *and before R_i records its local state in the second instance of the snapshot algorithm* ($t_{i,2}$).
(iii) *A packet sent from the local administrative domain of R_i is called a post-monitoring packet if and only if the packet is sent after R_i records its local state in the second instance of the snapshot algorithm* ($t_{i,2}$).

Figure 8, which shows a timing diagram of a router R_i, illustrates the three categories of traffics. All packets which are sent before the first instance of the snapshot algorithm are *pre-monitoring* packets. The packets, which are sent between two snapshots, are the *monitoring* packets, and these packets are actually the local traffic of R_i. The packets, which are sent

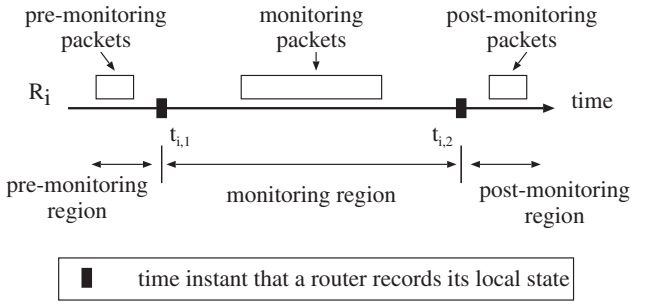

FIGURE 8. Classification of pre-monitoring, monitoring and post-monitoring packets.

after the second instance of the snapshot algorithm, are the *post-monitoring* packets.

Based on the above classification, one can have a better understanding about the snapshot algorithm. Let R_j be the downstream router of R_i. The channel state recorded by R_j in the first snapshot is the pre-monitoring packets from R_i entering the monitoring region of R_j, e.g. n_a and m_b in Figure 7.[2] Similarly, the channel state recorded by R_j in the second snapshot is the monitoring packets from R_i entering the post-monitoring region of R_j, e.g. n_d in Figure 7.

We analyze the effect of the channel states recorded by the routers from the point of view of the victim site. We denote the *aggregated channel states* as the sum of all channel states recorded in an instance of the snapshot algorithm. Let $\delta^{(k)}$ be the numbers of packets in the aggregated channel states of the k-th instances of the snapshot algorithm respectively, i.e.

$$\delta^{(k)} = \sum_{R_i,\, R_j \in \mathcal{G}} H_{ji}(t_{i,k}, t_{i,k}{}^{j}), \text{ where } k \in [1, 2]. \qquad (5)$$

During the first instance of the snapshot algorithm, $\delta^{(1)}$ represents all pre-monitoring packets that are received in the monitoring region of the victim site. Similarly, during the second instance of the snapshot algorithm, $\delta^{(2)}$ represents all monitoring packets which are received in the post-monitoring region of the victim site. Referring to the example in Figure 7, $\delta^{(1)} = n_a + m_b = 110$ and $\delta^{(2)} = n_d = 10$.

As a matter of fact, the monitoring packets sent from router R_i are the local traffic of R_i. If these packets are received only in the monitoring region, i.e. within $[t_1,t_2]$, of the victim site $\mathcal{V}$, the traffic inequality problem will not exist. However, the pre-monitoring packets of the aggregated channel states in the first instance of the snapshot algorithm arrive at $\mathcal{V}$ within $[t_1,t_2]$, therefore, the victim site actually receives both the monitoring and the pre-monitoring packets from all routers within $[t_1,t_2]$. Also, the monitoring packets of the aggregated channel states in the second instance of the snapshot algorithm arrive at $\mathcal{V}$ after t_2. Hence, the victim site does not receive all monitoring packets from the routers.

Let the local traffic of R_i be $L_i(t_{i,1}, t_{i,2})$, where $i \in [1, \ldots, n]$, and let $I_{\mathcal{V}}(t_1, t_2)$ be the incoming traffic of the victim site $\mathcal{V}$ within $[t_1,t_2]$. According to the above observation, we have the following equation relating $I_{\mathcal{V}}(t_1, t_2), L_i(t_{i,1}, t_{i,2})$ of $R_i, \delta^{(1)}$, and $\delta^{(2)}$:

$$I_{\mathcal{V}}(t_1, t_2) = \sum_{R_i \in \mathcal{G}} L_i(t_{i,1}, t_{i,2}) + \delta^{(1)} - \delta^{(2)}. \qquad (6)$$

The interpretation of Equation (6) is as follows. $I_{\mathcal{V}}(t_1, t_2)$ is composed of the monitoring packets from all the routers and the pre-monitoring packets of the aggregated channel states $\delta^{(1)}$. Thus, $I_{\mathcal{V}}(t_1, t_2)$ is the sum of $\sum_{R_i \in \mathcal{G}} L_i(t_{i,1}, t_{i,2})$ and $\delta^{(1)}$. However, the monitoring packets in the aggregated channel states $\delta^{(2)}$ are received by the victim site $\mathcal{V}$ after t_2. Therefore, $\delta^{(2)}$ is subtracted from the above sum. Referring to the example in Figure 7, we have the following result by using Equation (6):

$$\sum_{R_i \in \mathcal{G}} L_i(t_{i,1}, t_{i,2}) + \delta^{(1)} - \delta^{(2)} = 130 + 110 - 10 = 230,$$

The above value is exactly equal to the value of incoming traffic $I_{\mathcal{V}}(t_1, t_2)$ of $\mathcal{V}$. To summarize, the traffic inequality is compensated by Equation (6).

4.2. Calculating bounds for the number of packets arrived at the victim site

Recalling from the previous subsection, we have two important observations:

(i) The packets arrived at the victim site within $[t_1, t_2]$ not only include the monitoring packets from the routers, but also include the pre-monitoring packets of the aggregated channel state $\delta^{(1)}$ and
(ii) The monitoring packets of the aggregated channel state $\delta^{(2)}$ arrive at the victim site after t_2.

These observations imply that one cannot directly use the local traffics $L_i(t_{i,1}, t_{i,2})$ of the participating routers to find the locations of the attackers. To overcome this problem, one has to identify the originating domains of the pre-monitoring packets as well as the monitoring packets in the channel states.

Consider the illustration in Figure 9. which contains the same topology as in Figure 6 as well as a timing diagram that shows the time-lines of R_2, R_3 and R_4. The channel state of $Link_{3\ 2}$ is measured by the router R_2, and it contains the sequence of packets m_y. However, the packet sequence m_x may also be included in the channel state of $Link_{3\ 2}$ since m_x may arrive at router R_3 before R_3 records its local state. Thus, the packets in the channel state of $Link_{3\ 2}$ may be sent from

[2]Note these pre-monitoring packets may also enter the post-monitoring region of R_j but these packets will not affect our analysis. It is because it will be canceled out in the calculation of the local traffic.

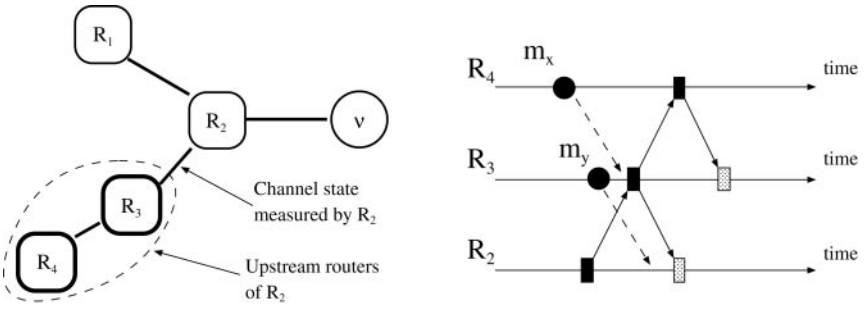

FIGURE 9. The channel state of $Link_{32}$ contains pre-monitoring (monitoring) packets from both R_3 and R_4 in the first (second) instance of the snapshot algorithm.

R_2's upstream routers R_3 and R_4. In summary, let $R_j \in \mathcal{U}(R_i)$, and let R_i and R_j be connected by $Link_{ji}$. A non-empty channel state, $H_{ji}(t_{i,k}, t_{i,k}{}^j)$, of $Link_{ji}$, is composed of the pre-monitoring (monitoring) packets from the upstream routers of R_i in the first (second) instance of the snapshot algorithm, and these packets are from $Link_{ji}$.

Based on the above observation, one cannot determine the originating domains of the packets in the channel states. This implies that one cannot calculate an exact number of packets that have arrived at the victim site within $[t_1, t_2]$ from each participating router. However, we provide a methodology to determine the upper and the lower bounds of these packets. Let R_r be an upstream router of R_q. Let $H_{pq}(t_{q,k}, t_{q,k}{}^p)|_{R_r}$ represent the *exact* number of packets, which are sent from the domain of R_r, and contribute to the channel state $H_{pq}(t_{q,k}, t_{q,k}{}^p)$ of $Link_{pq}$. Hence, the channel state of $Link_{pq}$ is the sum of $H_{pq}(t_{q,k}, t_{q,k}{}^p)|_{R_r}$ for all upstream routers R_r of R_q, and the corresponding equation is as follows:

$$H_{pq}(t_{q,k}, t_{q,k}{}^p) = \sum_{R_r \in \mathbb{U}(R_q)} H_{pq}(t_{q,k}, t_{q,k}{}^p)|_{R_r}. \tag{7}$$

Also, recall from Section 1 that $\mathbb{D}(R_i)$ represents a set of downstream routers of R_i. The number of monitoring packets that are generated by the domain of R_i and arrive at $\mathcal{V}$ after t_2 is

$$\sum_{R_p, R_q \in \mathbb{D}(R_i)} H_{pq}(t_{q,2}, t_{q,2}{}^p)|_{R_i}. \tag{8}$$

Equation (8) represents the number of monitoring packets which are sent from the domain of R_i and are recorded as the channel states of the downstream router of R_i in the second instance of the snapshot algorithm. $L_i(t_{i,1}, t_{i,2})$ represents the number of monitoring packets sent by R_i within the snapshot interval. Since the packets in Equation (8) are the monitoring packets that are not received at $\mathcal{V}$ within $[t_1, t_2]$, the number of monitoring packets which are sent from R_i and are received by $\mathcal{V}$ in $[t_1, t_2]$ is

$$L_i(t_{i,1}, t_{i,2}) - \sum_{R_p, R_q \in \mathbb{D}(R_i)} H_{pq}(t_{q,2}, t_{q,2}{}^p)|_{R_i}. \tag{9}$$

Let $L_i^*(t_{i,1}, t_{i,2})$ be the number of packets which are sent from R_i and are received by the victim site $\mathcal{V}$ within $[t_1, t_2]$ (the real local traffic). These packets are composed of two components: (i) the monitoring packets sent from R_i and arrived at $\mathcal{V}$ in $[t_{i,1}, t_{i,2}]$, which is given by Equation (9), and (ii) the pre-monitoring packets sent from R_i and arrived at $\mathcal{V}$ within $[t_1, t_2]$, and these packets are given as follows:

$$\sum_{R_p, R_q \in \mathbb{D}(R_i)} H_{pq}(t_{q,1}, t_{q,1}{}^p)|_{R_i}. \tag{10}$$

Thus, by Equations (9) and (10), the real local traffic $L_i^*(t_{i,1}, t_{i,2})$ of R_i is represented as follows:

$$L_i^*(t_{i,1}, t_{i,2}) = L_i(t_{i,1}, t_{i,2}) - \sum_{R_p, R_q \in \mathbb{D}(R_i)} H_{pq}(t_{q,2}, t_{q,2}{}^p)|_{R_i} + \sum_{R_p, R_q \in \mathbb{D}(R_i)} H_{pq}(t_{q,1}, t_{q,1}{}^p)|_{R_i}. \tag{11}$$

Note that it is possible that the pre-monitoring packets may arrive at $\mathcal{V}$ after t_2 probably because the interval $[t_1, t_2]$ is not long enough. However, those packets will not affect the correctness of the calculation of the real local traffic $L_i^*(t_{i,1}, t_{i,2})$ because these packets will be recorded in both Equations (8) and (10). As shown in Equation (11), these packets will be canceled out.

Let $Upper(L_i^*)$ and $Lower(L_i^*)$ be the upper and lower bounds of the real local traffic $L_i^*(t_{i,1}, t_{i,2})$ respectively. To find the bounds of $L_i^*(t_{i,1}, t_{i,2})$ in Equation (11), one can observe that

$$H_{pq}(t_{q,k}, t_{q,k}{}^p) \geq H_{pq}(t_{q,k}, t_{q,k}{}^p)|_{R_i}.$$

Therefore, $Upper(L_i^*)$ and $Lower(L_i^*)$ are

$$L_i^*(t_{i,1}, t_{i,2}) \leq L_i(t_{i,1}, t_{i,2}) + \sum_{R_p, R_q \in \mathbb{D}(R_i)} H_{pq}(t_{q,1}, t_{q,1}{}^p), \tag{12}$$

$$L_i^*(t_{i,1}, t_{i,2}) \geq L_i(t_{i,1}, t_{i,2}) - \sum_{R_p, R_q \in \mathbb{D}(R_i)} H_{pq}(t_{q,2}, t_{q,2}{}^p). \tag{13}$$

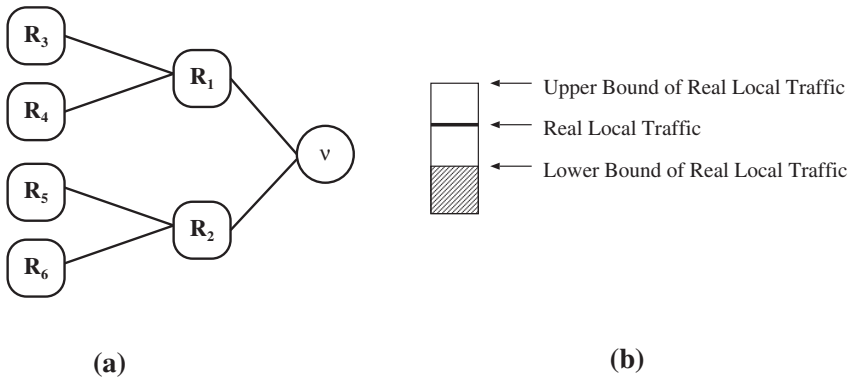

FIGURE 10. (a) Network topology and (b) legend for simulations A and B.

Referring to the example in Figure 7, $Upper(L_4^*)$ and $Lower(L_4^*)$ are

$$Upper(L_4^*) = L_i(t_{4,1}, t_{4,2}) + H_{43}(t_{3,1}, {t_{3,1}}^4) = 110$$
$$Lower(L_4^*) = L_i(t_{4,1}, t_{4,2}) - H_{43}(t_{3,2}, {t_{3,2}}^4) = 0.$$

Since the attacker in R_4 sends the sequence of malicious packets m_b to $\mathcal{V}$, $Upper(L_4^*)$ is significantly higher than the others. This suggests that the domain of R_4 is also a possible location of the attackers. In the next section, we present the simulation results that show the effectiveness of our distributed methodology.

5. PERFORMANCE EVALUATIONS

In the previous sections, we presented the DDoS traceback algorithm to determine the intensity of local traffic for each participating router. In this section, we carry out two different sets of simulations to demonstrate the effectiveness of our proposed methodology. In the first set of simulations (Simulation A), we use a simple network topology as depicted in Figure 10a to illustrate the correctness and robustness of our algorithm under various factors (e.g. different processes of generating traffic, different attackers' location distributions, ..., etc). For the second set of simulations (Simulation B), we extend the performance study to a large scale realistic Internet topology.

Simulation A (*correctness and robustness of DDoS traceback algorithm*). This set of simulations evaluates the correctness of the proposed DDoS traceback algorithm. For this set of simulations, we use a network topology in Figure 10a which contains six routers. The packets are generated by two methods: (i) constant rate (e.g. an average rate of 100 pkts/s implies that every 0.01 s, a router will generate a new packet to the victim site $\mathcal{V}$), (ii) exponential on/off process (i.e. packets are sent at a fixed rate during the 'on' periods, and no packet will be sent during the 'off' periods). Both the on and off periods is taken from an exponential distribution. The average duration for the on period and the off period is set to 100 ms in this set of simulations. The bandwidth and the delay of each link are set to 100 Mbps and 50 ms respectively.

Simulation A.1 (*Bounds for the local traffic*). In this simulation, there is one attacker who is located at the domain of R_3. The attack traffic rate of R_3 is set as a constant rate of 500 pkts/s while the normal traffic rates for all other routers are set to a constant rate of 100 pkts/s. The victim site $\mathcal{V}$ initiates the DDoS traceback algorithm to determine the location of the attackers. Figure 10b illustrates the legend for various graphs in our simulations. Figure 11 shows the upper bound of real local traffic $Upper(L_i^*)$, the lower bound of real local traffic $Lower(L_i^*)$, as well as the real local traffic L_i^* for all six routers in four different measurement intervals. The snapshot time interval of four cases are 1, 2, 3 and 4 s respectively. The lower bound and upper bound of real local traffic are computed based on Equations (12) and (13). The real local traffic L_i^* is the number of packets sent from router R_i and received by the victim site $\mathcal{V}$ within the snapshot time interval. Note that the real local traffic L_i^* is only provided in the simulation environment. In Figure 11, one can observe that

(i) The real local traffic L_i^* is between $Upper(L_i^*)$ and $Lower(L_i^*)$, which means that our DDoS traceback algorithm can successfully bound the exact number of packets sent from router R_i and received by the victim site $\mathcal{V}$ in the snapshot time interval.
(ii) The difference between the bounds of the real local traffic will reduce if we increase the duration of the measurement interval.
(iii) The lower bound of real local traffic of the attack domain R_3 is significantly higher than the upper bound of real local traffic of other routers. This implies we can locate the source of the attack traffic.
(iv) Lastly, we observe that the measurement interval can be very short (e.g. 4 s) and one can quickly determine the domain of R_3 is the location of the attacker.

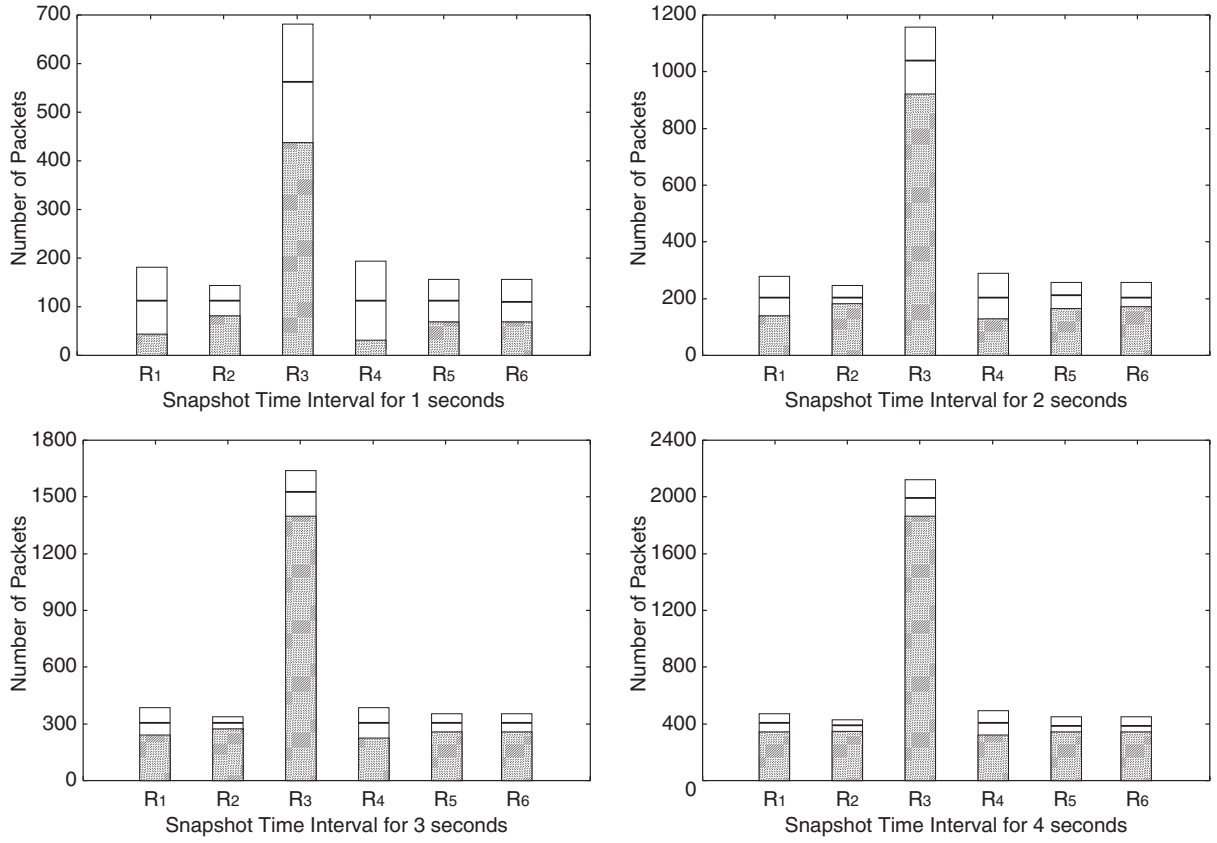

FIGURE 11. *Simulation A.1.* Bounds for the real local traffic under constant traffic rate.

Simulation A..2 (*Exponential on/off process for packet generation*). In this simulation, we consider the packet generation process which is based on an on/off exponential process. We use the same network topology in Figure 10a and repeat the similar simulation as in Simulation A.1. The average duration of the on period and the off period is set to 100 ms in this simulation. Figure 12 illustrates the simulation results. We observe that even if the packet generation process is governed by an on/off process, the algorithm is robust enough to accurately determine the local traffic intensities of all participating routers. Similar to Simulation A.1, the same conclusion can be made for this simulation.

Simulation A.3 (*Multiple attackers*). In this simulation, there are two attack domains and they are located in R_3 and R_5. We repeat the similar simulation as in Simulation A.2. The average on period and off period is set to 100 ms. The local traffic from the attack domains is set as 1000 pkts/s while the local traffic of the normal domain is 100 pkts/s. In Figure 13, we observe that the lower bound of real local traffic of the domains of R_3 and R_5 is *significantly* higher than the upper bound of real local traffic of other domains for all cases. Therefore, our DDoS traceback algorithm can effectively and quickly determine various attackers' locations.

Simulation A.4 (*Varying attackers' location*). In this simulation, we consider different locations for the two attackers and analyze their effect. We repeat the same simulation as in Simulation A.4 but the two attackers are in routers R_2 and R_4. In Figure 14, one can observe that the lower bound of real local traffics of the domains of R_2 and R_4 is *significantly* higher than the upper bound of real local traffics of other domains for all cases. We conclude that our methodology is robust and it is not sensitive to the location distributions of different attackers.

Simulation A.5. (Different attack traffic rates). In this simulation, we investigate the effect of different attack traffic rates on the traceback result. Again, as similar to Simulation A.1, the normal traffic is sending at a rate of 100 pkt/s, and the attacker is in the domain of R_3 sending out packets with a constant bit rate. But, now, we carry out the simulation with different attack traffic rates ranging from 50 pkt/s to 1000 pkt/s at a step of 25 pkts/s. Our aim is to investigate (i) in what percentage the attack traffic contributes to the aggregated traffic received by the victim and (ii) in what range of the attack traffic the traceback methodology is effective in locating the attackers.

In Figure 15, there are five different plots of the measured attack traffic percentage against the attack traffic rate at five

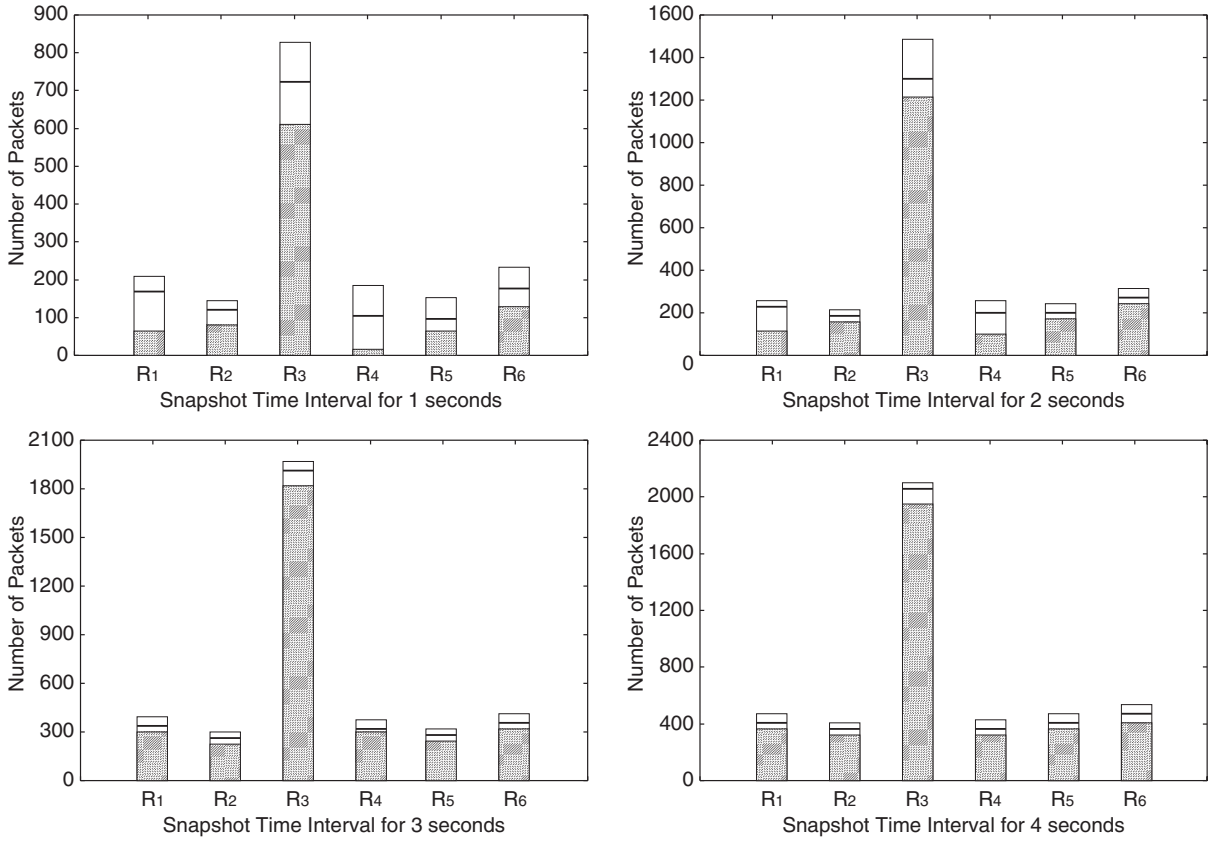

FIGURE 12. Simulation A.2. The real local traffic under exponential on/off process.

different snapshot intervals: 1, 2, 5, 10 and 100 s. The attack traffic percentage is calculated by dividing the local traffic of R_3 by the total number of received packets. Firstly, according to the figure, the percentage of the measured attack traffic decreases as the snapshot interval increases. Nevertheless, the decreasing percentage will eventually converge to a certain value as shown in the plot of 100 s snapshot interval.

We now describe the way in which the victim locates the attackers. We denote a term *filtering threshold* which means if the percentage that a domain's traffic contributes to the aggregated traffic is over that threshold, that domain will be considered as an attacking domain. Eventually, the corresponding network administrator will be notified and starts filtering the large flow. If one sets the threshold to 50%, then, referring to the figure, one can only find attackers with traffic rate greater than or equal to 500 pkts/s, labeled by the coordinates (500, 50.10). For another example, if the threshold is set to be 30%, then the victim can find attackers with the traffic rate greater than 225 pkts/s.

According to this simulation, we can find two main factors affecting the effectiveness in locating the attack traffics.

(i) *The attack traffic to the normal traffic ratio*. According to the simulation results, the methodology may fail to detect the attacker if the attack traffic rate is not large enough.

(ii) *The total number of domains*. If the total number of domains that the traceback methodology is monitoring is large, then even every innocent domain sends a small amount of traffic, this makes the attack flow not significantly dominating and lowers the percentage of the attack flow. Under this situation, this may require the administrator to lower the filtering threshold. For the case that the total number of domains is small, a similar analysis can be applied and it is suggested that the threshold should be a large value in order that the innocent domains will not be misclassified as attacking domains.

In conclusion, the set of results in Simulation A shows the following findings.

(i) As the snapshot interval becomes longer, the upper bound and the lower bound of the local traffic become closer to the measured local traffic.

(ii) The methodology can quickly locate the attackers with dominating large flows, however the efficiency is subject to the magnitude of the attack flow and the total number of domains in the network.

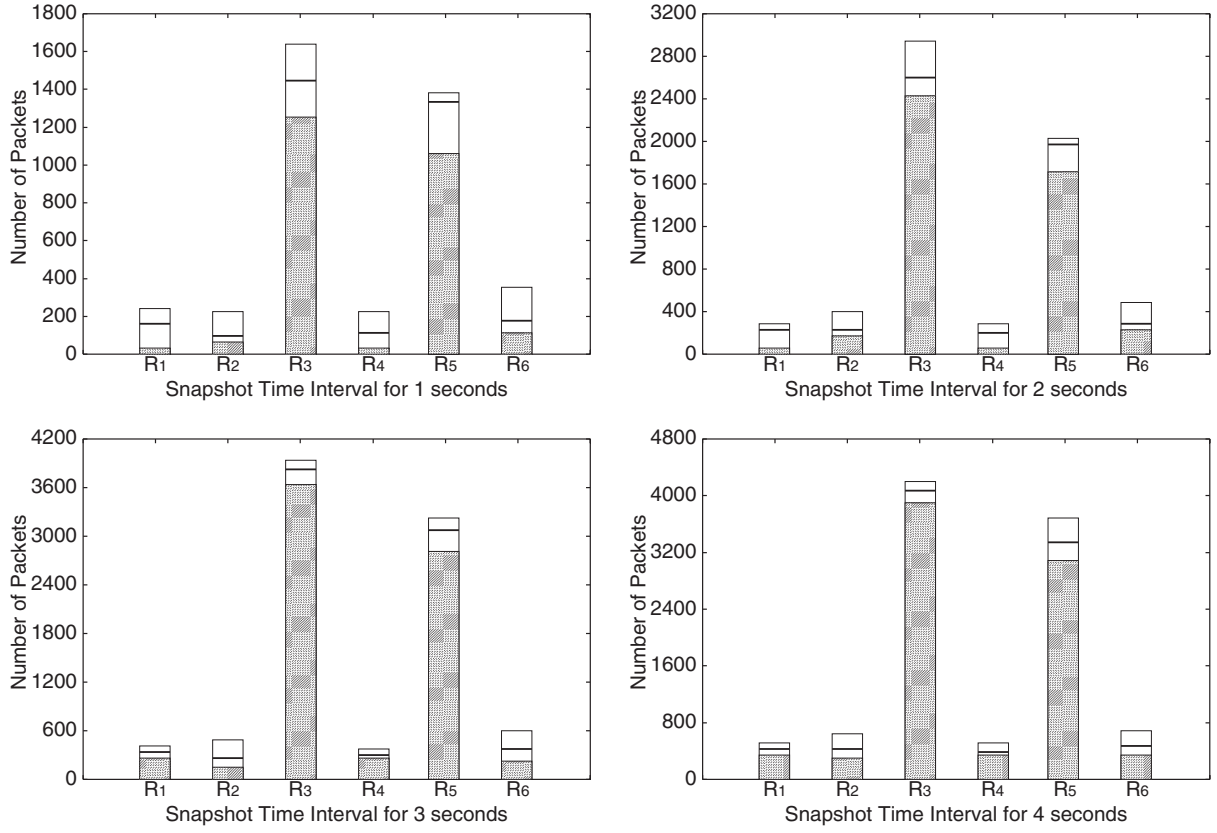

FIGURE 13. *Simulation A.3.* Effect of multiple attackers on the real local traffic bounds.

Simulation B (*simulations on a large-scale realistic Internet topology*). To validate the correctness of the theoretical bounds of the local traffic, we extend the performance study to a large-scale and realistic Internet topology. We use the Internet topology from [7]. The testing dataset in our simulations contains 1000 distinct routers. The source of traceroute is considered as the victim site $\mathcal{V}$ and the traceroute dataset is considered as the map of the upstream routers. We use this dataset and construct a network simulation test-bed based on *NS*-2. There are five classes of attack traffic rates. The attack traffic rates of Classes 1, 2, 3, 4 and 5 are 150, 175, 200, 225 and 250 pkts/s respectively. There are 10 attackers in each class and the attackers are evenly distributed in the different domains of the network. The local traffic rate of normal domain is 10 pkts/s. Packets are generated according to the exponential on/off process. Both the average on/off periods are set to 100 ms in this simulation. The bandwidth and the delay of each link are set to 100 Mbps and 20 ms respectively.

The victim site $\mathcal{V}$ initiates the DDoS traceback algorithm to determine the location of the attackers. Figure 16 shows that the upper bound of real local traffic $Upper(L_i^*)$ the lower bound of real local traffic $Lower(L_i^*)$, and the real local traffic L_i^* for the five classes of attacker as well as for the normal routers. In this simulation, we have four different measurement intervals, they have the duration of 5, 10, 15 and 20 s respectively. The attack domain which has the largest upper bound of real local traffic within its class is selected and its traffic rates are plotted in the figure. From Figure 16, one can observe that L_i^* is in-between $Upper(L_i^*)$ and $Lower(L_i^*)$ for all classes, which means that our algorithm can successfully bound the real local traffics of all participating routers. When the snapshot time interval increases from 5 to 20 s, we observe that the spread of the bounds of the local traffic tends to decrease. Therefore, the estimation of the real local traffic L_i^* becomes more accurate for a longer snapshot time interval. On the other hand, we can see that the lower bound of real local traffic from each of the five attack domains is significantly higher than the upper bound of real local traffic of normal domain. It implies that we can quickly (e.g. with 20 s) and accurately (based on the differences of the bounds) determine the locations of the attackers.

6. IMPLEMENTATION ISSUES

In previous sections, we have shown that our DDoS traceback algorithm is effective in locating potential attackers

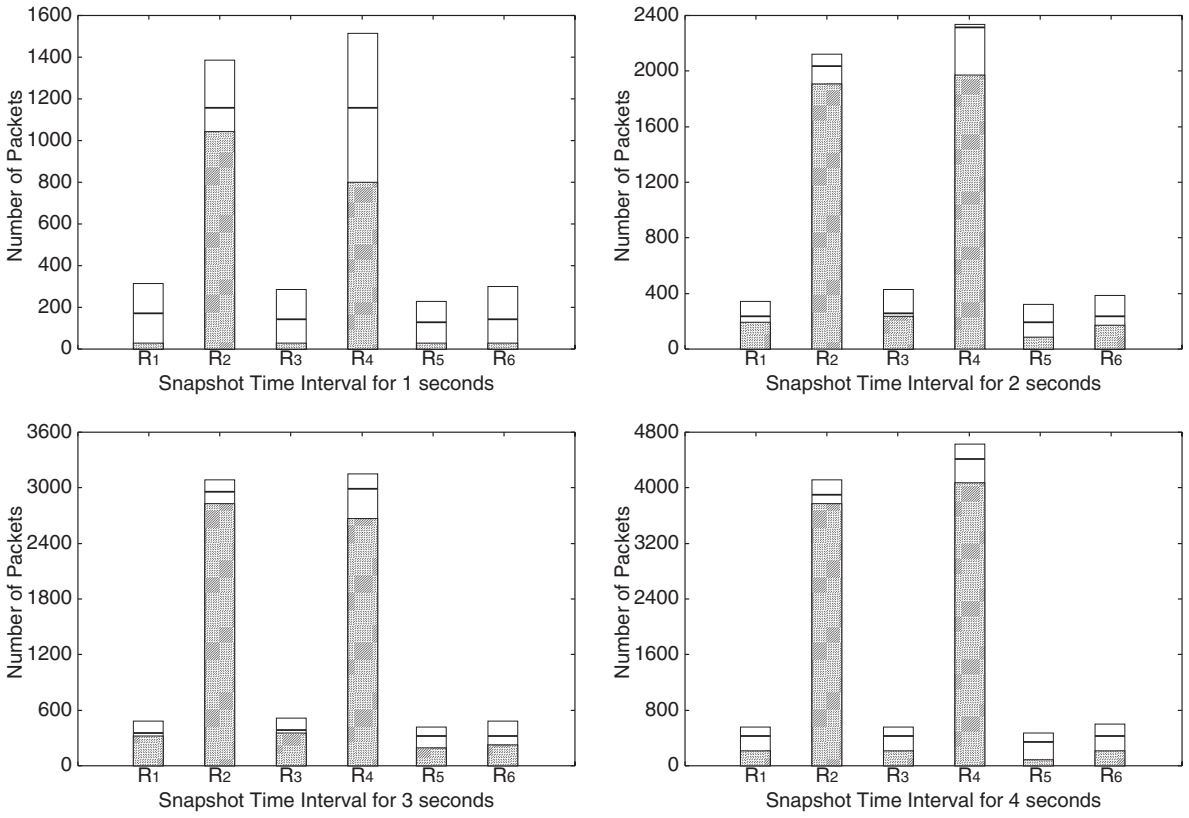

FIGURE 14. Simulation A.4. Effect of new attackers' locations.

and filtering attack packets. The distributed traceback algorithm relies on the assumption that the victim site $\mathcal{V}$ has a map of its upstream routers. In this section, we illustrate this assumption is reasonable and practical. Also, we show that our proposed distributed traceback algorithm leverages on the existing traceback technologies and can complement existing infrastructure such as the ICMP traceback technique [8]. However, we cannot show that the overhead of the proposed method is small on high-end routers. We can only show that the overhead of the traceback service is not significant under a set of experimental Linux routers.

6.1. Topology construction

There are ways to obtain a map of upstream routers for a given victim site. Many network management tools exist for mapping, for example a tool based on traceroute from Lucent Bell Labs [7] and a tool based on ICMP echo requests from CAIDA [9]. In these techniques, the victim site $\mathcal{V}$ sends packet to probe hosts which are $k \geq 1$ hops away. This packet contains a TTL field which is decremented by one for each traverse link. When the TTL reaches zero, the router sends a reply back to $\mathcal{V}$. This form of probing provides the router adjacency information which can help $\mathcal{V}$ to build a map of upstream routers.

Another efficient method to obtain an upstream map is to store the router adjacency or edge information into the packets. Approaches like probabilistic packet marking [10, 11] encode the router adjacency information into the packet header. Other approaches like itrace [8, 12, 13] generate separate ICMP packet with router adjacency information to a victim site $\mathcal{V}$. When $\mathcal{V}$ receives these packets, it extracts the router adjacency information to build a map of upstream routers.

Note that when one invokes our proposed distributed traceback methodology, the traceback region occurs only *within* the map of upstream routers. In other words, only routers within the map need to participate in the distributed traceback. Since it is possible that some leaf routers are at the edge of the map and, at the same time, are connected to some other routers outside the map. In this case, all transit and local traffics of this type of leaf routers will be considered as the local traffics of these leaf routers. One can progressively apply the distributed traceback algorithm to this type of leaf routers to determine the source of the attack. For example, if the local traffic of a leaf router is very high, then one can consider this leaf

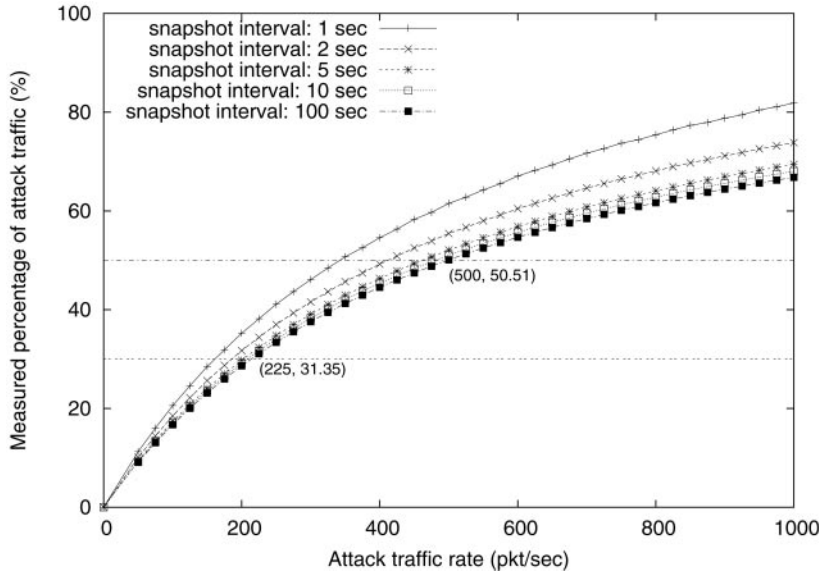

FIGURE 15. Simulation A.5. On different attack traffic rates.

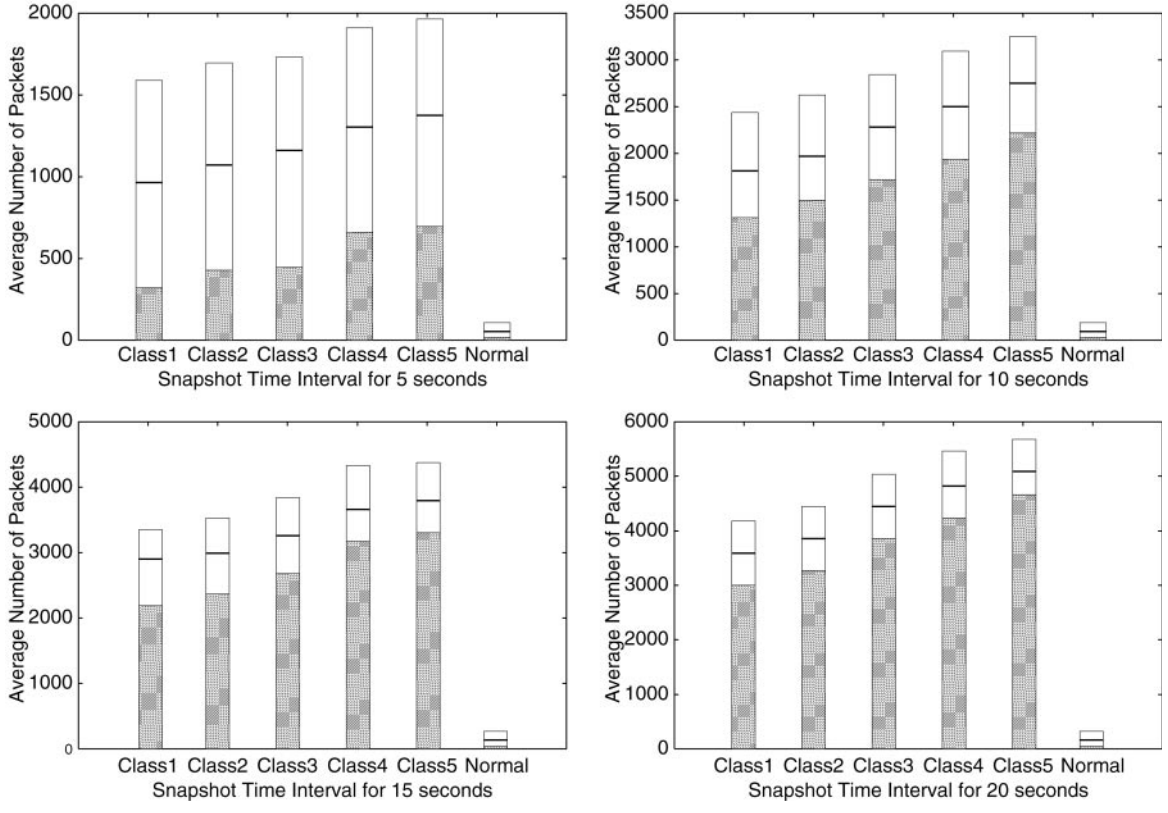

FIGURE 16. Simulation B. Simulation for large scale Internet Topology.

router as a victim site and initiate the distributed traceback algorithm again.

However, it is heavy for an ordinary host to store the map as it is huge in size. Nevertheless, techniques like the Sink Hole [14] can help forwarding traffic to a data processing center hosted by the ISP. The data collection process and the network map storage can then be done in that dedicated host. The only weakness about this approach is that if there

are many hosts requesting the traceback service, the loading of the data processing center will become huge.

6.2. System overhead

The proposed traceback methodology is working on the victim site and the participating routers. Under most of the execution time, the routers have to keep track on traceback data whenever a packet passes through it. The processing of the outgoing counters, the markers and the channel states must incur an inevitable overhead on the router. However, we cannot provide any solid data about this issue. The problem is that nowadays high-end routers, which are deployed worldwide, do not provide any programmable feature for us to modify the router and measure the overhead of our proposed traceback methodology. Nevertheless, on the low-end side, the Linux-based router provides us a possible choice of the programmable routers.

We have implemented a programmable router prototype together with our proposed DDoS traceback algorithm named the *OPERA* [15] by introducing new modules to the *netfilter* [16] in the Linux system. Although the system overhead on high-end routers remains a subject of the future research, we provide the system overhead analysis on low-end routers to show, firstly, the proposed methodology is implementable and is deployable, and, secondly, the proposed methodology does not involve complex computation and hence has a small overhead problem on the low-end routers.

We have implemented an experimental environment called the OPERA. Each router has to install and load the modules provided in OPERA. Nevertheless, it is difficult to carry out experiments to measure the overhead, however, the work done by Harris and Melara [17] can support our claim that the system overhead for a Linux router is not expensive.

The work done by Harris and Melara [17] has carried out experiments to test the firewall in Linux machines, i.e. the *iptables*. We focus on the latency test which shows that the performance degrades as the number of filter rules increases. The experiments carried out in [17] show that when filtering IP addresses, TCP/UDP ports and MAC addresses, the latency for filtering a packet per rule increases linearly and the latencies are approximately 0.12, 0.66 and 0.68 μs/rule respectively.

In the OPERA project, we utilize the iptables, but, we also introduce new functionality by inserting routines into the hook points of the netfilter. When we implement the snapshot algorithm, the inserted routine only handles two events: (i) updates a variable whenever a packet comes in and (ii) responses instantly on incoming markers. The first event can be handled efficiently as it is just a variable update, and the second event requires a routine in matching the source of an incoming marker as well as the injection of a new marker. Nevertheless, these two events are analogous to a filtering routine. Thus, the introduced system overhead will be as light as using the iptables, which is widely used in Linux system nowadays, with only two filtering rules introduced.

On the other hand, for each victim site, OPERA only needs to allocate one set of memory for each outgoing interface and another set of memory for each channel state. Hence, this involves only limited amount of memory usage,[3] and is scalable for several hundreds of registered victim sites.[4] We argue that this traceback service is a privileged service. There will not be many sites paying for this service except those with high popularities. It is peculiar for a router to handle several thousands of victim sites simultaneously. If this does happen, this is a sign of another level of DDoS attack and the attack target is the traceback mechanism itself. This requires a *distributed authentication protocol* among the routers and the victim sites. This will be mentioned in the future work of Section 9.

Further, if there are compromised hosts sending requests of tracing DDoS attacks which are not really happening, our system will be overwhelmed by the malicious hosts. If a host is compromised, the most important point to note is the ability for a router, a victim or a third party (e.g. Certificate Authority) to discover its malicious identity. In most cases, there is no way for any entity to distinguish whether a request is coming from a compromised host or not. If a host is compromised, the compromiser most likely has the private information of that host such as the encryption keys and the authentication secrets, and she is free to invoke the traceback service even though an authentication protocol is implemented between the routers and the clients. Hence, the method to discover the malicious identity of a compromised host is beyond the scope of our traceback system.

6.3. Implementation issue based on ICMP traceback

Our proposed distributed traceback methodology can complement and leverage on the current ICMP traceback [8]. The main idea of ICMP traceback is that, for each router, it samples the forwarding packets with low probability (e.g. 1/20000), and to generate an authenticated ICMP traceback message on to the sampled packets and forwards to the victim site. The ICMP traceback message has information about the routers on which link interfaces packets arrive and depart, as well as the information about its previous and next routers. During a DoS attack, a victim can use these ICMP traceback messages to reconstruct an attack path.

[3] For example, a router with three incoming interfaces and one outgoing interface only needs four variables, and each variable needs 4 bytes (an unsigned long integer) for one set of snapshot data. A total memory usage is only 16 bytes.

[4] There are several hundred Kbytes of memory available in the kernel of Linux.

We view that our proposed distributed traceback methodology and the ICMP traceback infrastructure are complimentary. The ICMP traceback approach encodes a map of upstream routers in the ICMP traceback messages. Therefore, a victim site can use the information in ICMP traceback messages to build a map of upstream routers. A router running the ICMP traceback service has the capability to associate a packet with the input port or MAC address on which a packet arrived. This capability can help the routers to count the incoming and outgoing traffic in our distributed traceback algorithm. One can also use the existing ICMP traceback infrastructure so that routers can send back local state and channel states back the victim site. Another important point is that the ICMP traceback provides an authentication service. This can also be applied to authenticate the victim site, the senders of the marker, local state and channel state information. The main disadvantage of the existing ICMP traceback is, due to the low probability of generating ICMP messages, it requires many attack packets to pass through a router so as to identify the locations of the attackers. On the contrary, our distributed traceback algorithm can trace the location of the attackers in a short period time. Hence, one can achieve a more effective traceback performance by using our proposed distributed traceback algorithm in conjunction with the ICMP traceback service.

6.4. An alternative to aggregate congestion control and push-back

Our proposed methodology can be treated as an alternative to the *aggregate congestion control* [18] (ACC in short, and a brief survey in included in Section 8). The ACC, same as our proposed approach, also requires modifications of the routers and introduces inevitable overheads. The reasons why our approach can be an alternative to the ACC mechanism are as follows:

- The modifications of the routers for ACC approach are much more complex than the modification brought about by our approach, and this implies a heavier overhead for the routers.
- The classification of the aggregates is a not a light burden for the router. For example, the router may have to match the 'characteristics' of every incoming packet against every definition of the known aggregates. As suggested in [18], the classification of aggregates depends on the rules known to the routers. If the number of rules is large, which is quite certain to be true in order to have an effective aggregation detection, the burden will be large. As the overhead of our approach grows with the number of victims while the ACC approach has an inevitable large overhead, our approach can be a better choice.

6.5. Partial deployment

Our traceback scheme is proofed to provide a meaningful traceback result according to the previous analysis and simulations. However, we have assumed that all the routers involved in the traceback are equipped with the traceback ability. In this subsection, we discuss the possibility for our scheme to provide a meaningful traceback result under a *partial deployment environment*, which means that not all routers involved know the traceback protocol. Firstly, we have the following notations. We name a router with the traceback scheme deployed a *deployed router* while a router without the traceback scheme deployed an *undeployed router*.

Our idea in providing the partial deployment is to treat the local traffic generated from an undeployed router as the local traffic of its nearest downstream deployed router. Thus, if an attacker is located in the domain of an undeployed router, then its downstream deployed router will report a high level of local traffic. This suggests that an attacker is hiding in either the deployed router or its upstream deployed router.

However, the tradeoff in providing the partial deployment is the introduction of a set of strict conditions. The conditions are as follows.

(i) *The last mile router of the victim must be a deployed router*. If the last mile router of the victim is an undeployed router, then the local traffic of the last mile router will become the local traffic of the victim, which is not a reasonable result because the victim should not generate any local traffic.

(ii) *An undeployed router will not process nor drop the marker packet.* The undeployed router is transparent to the traceback protocol, and thus the marker should only be forwarded by the undeployed router.

(iii) *Each deployed router knows whether a router in the Internet map is a deployed router or an undeployed router*. From the viewpoint of a deployed router, in order to send markers to its nearest upstream deployed routers, the deployed router needs to know the location of the nearest upstream deployed routers. In a partial deployment environment, the nearest upstream deployed routers may not be the immediate neighboring upstream router. Therefore, for simplicity, each deployed router is required to know the locations of all the deployed and the undeployed routers.

To illustrate how the partial deployment works, we revisit the example in Figures 6 and 7 in Section 3.2. Figure 17a is a replicated network of Figure 6 except that the router R_3 is now an undeployed attacker.

When the traceback starts, the victim sends a marker to the router R_2. When R_2 receives that marker, it decides the set of routers to which the marker should be sent. According

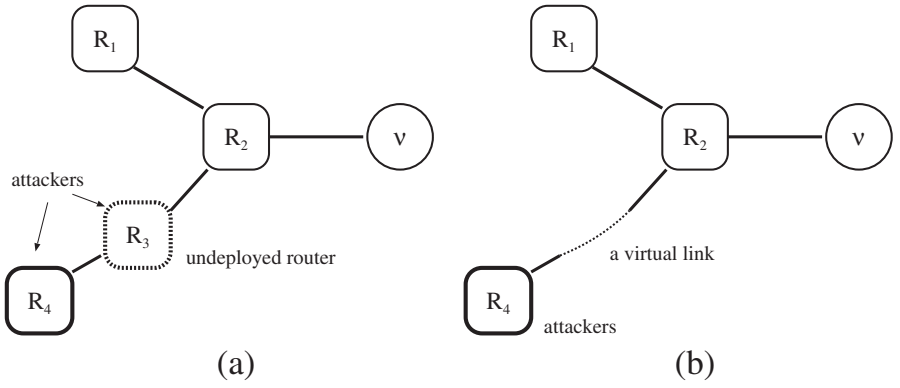

FIGURE 17. (a) The same network example as Figure 6 with attacking domains R_3 and R_4. But the router R_3 is an undeployed router. (b) Logically, a virtual link between the router R_2 and R_4 is formed.

to the third condition (the deployed router knows where the nearest upstream deployed routers are), the router R_2 will find that the nearest upstream deployed routers are R_1 and R_4. Then, R_2 sends two markers destined for R_1 and R_4 accordingly. As the undeployed router R_3 only forwards the marker packet, eventually, all deployed routers will be invoked. Meanwhile, the router R_2 is instructed to record the channel states until the markers from routers R_1 and R_4 arrive. From the viewpoint of router R_2, when it records the channel state, it no longer records the channel state of the physical link $Link_{32}$. Instead, a *virtual link* $Link_{42}$ is established as shown in Figure 17b, and the router R_2 will monitor this virtual link.

Figure 18 shows the timing diagram of the traceback result under the partial deployment environment, and it shows the same scenario as in Figure 7 in Section 3.2 except that the router R_3 is an undeployed router, thus there is no reading recorded by R_3. Also, the timing diagram depicts that the router R_1 is recording the channel state of the virtual link $Link_{42}$. We now calculate the accumulative local traffic of R_2 by using Equation (4).

$$\begin{aligned} N_2(t_{2,1}) &= C_2(t_{2,1}) - (C_1(t_{1,1}) - H_{12}(t_{2,1}, {t_{2,1}}^1)) \\ &\quad - (C_4(t_{4,1}) - H_{42}(t_{4,1}, {t_{4,1}}^4)) \\ &= 0 - (10 - 10) - (100 - 100) = 0. \\ N_2(t_{2,2}) &= C_2(t_{2,2}) - (C_1(t_{1,2}) - H_{12}(t_{2,2}, {t_{2,2}}^1)) \\ &\quad - (C_4(t_{4,2}) - H_{42}(t_{4,2}, {t_{4,2}}^4)) \\ &= 230 - (20 - 0) - (110 - 10) = 110. \end{aligned}$$

The local traffic of R_2 is 110, which is the sum of the local traffic of R_2 and R_3 in the full deployment environment (see Table 4). Hence, the local traffic of the undeployed router R_3 is forwarded to the downstream deployed router R_2.

However, there is one problem about the introduction of the virtual link. To illustrate, let us consider the scenario in

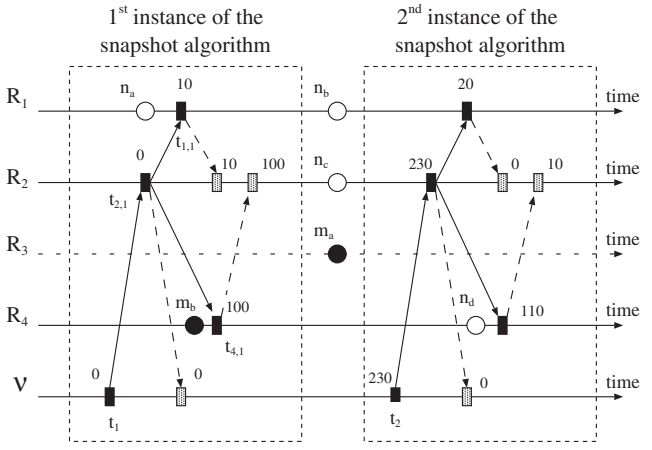

FIGURE 18. A timing diagram that shows the progress of the DDoS traceback algorithm under the partial deployment environment.

Figure 19. According to the snapshot algorithm, the router R_1 will record the channel states of the virtual links $Link_{31}$ and $Link_{41}$. But, the channel states that R_1 is recording is the physical link $Link_{21}$. As R_2 routes packets from R_3 and R_4, the physical channel $Link_{21}$ is a mixture of channel states of $Link_{31}$ and $Link_{41}$. As a result, R_1 is not able to distinguish the two virtual links. This problem is illustrated by the zoom-in part of Figure 20. After the marker from R_3 arrives, the physical channel $Link_{21}$ now only belongs to the virtual link $Link_{41}$.

To remedy the problem, the following approximation is applied: to distribute the mixed channel states into two shares proportional to the measured local states of R_3 and R_4. Mathematically, we have the following. Denote the mixed channel state measured by R_1 on the physical link $Link_{21}$ as $H_{21}(t_{1,1}, {t_{1,1}}^3)$, and denote the channel state measured by R_1 on the physical link after R_1 receives a marker from R_3 as $H_{21}({t_{1,1}}^3, {t_{1,1}}^4)$. Also, denote the local

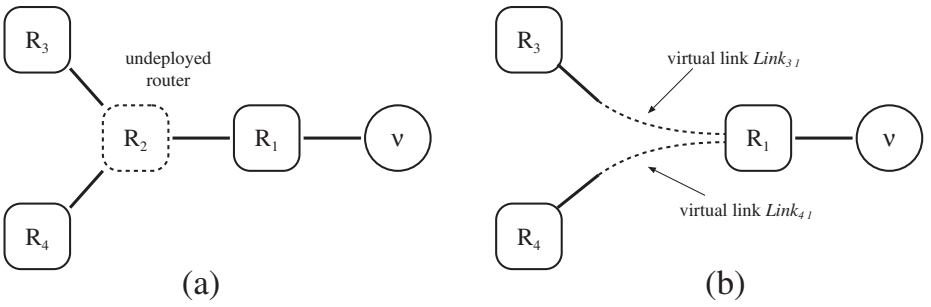

FIGURE 19. (a) In this example network, the router R_2 is an undeployed routers while the others are deployed routers. (b) As the undeployed router is transparent to the traceback protocol, the router R_1 records the channel state of the virtual links $Link_{31}$ and $Link_{41}$.

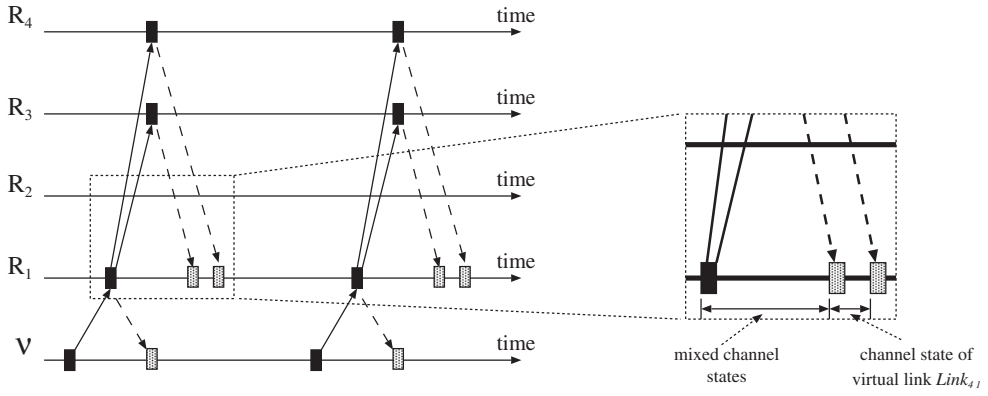

FIGURE 20. The timing diagram under a partial deployment environment. A drawback is that the channel states of the virtual links $Link_{31}$ and $Link_{41}$ become indistinguishable at router R_1.

traffic measured at R_3 and R_4 as $L_3(t_{3,1},t_{3,2})$ and $L_4(t_{4,1},t_{4,2})$ respectively. Then, the channel states $H_{31}(t_{1,1}, {t_{1,1}}^3)$ and $H_{41}(t_{1,1}, {t_{1,1}}^4)$ are given as follows:

$$H_{31}(t_{1,1}, {t_{1,1}}^3) = H_{21}(t_{1,1}, {t_{1,1}}^3) \times \frac{L_3(t_{3,1}, t_{3,2})}{L_3(t_{3,1}, t_{3,2}) + L_4(t_{4,1}, t_{4,2})}; \quad (14)$$

$$H_{41}(t_{1,1}, {t_{1,1}}^4) = H_{21}(t_{1,1}, {t_{1,1}}^3) \times \frac{L_3(t_{3,1}, t_{3,2})}{L_3(t_{3,1}, t_{3,2}) + L_4(t_{4,1}, t_{4,2})} + H_{21}({t_{1,1}}^3, {t_{1,1}}^4). \quad (15)$$

The rationale of this solution is based on the assumption that if the ratio of the local traffic of R_3 to the local traffic of R_4 is at a certain value (say x) within the snapshot interval, then, during the time before and after the snapshot interval, the ratio will be very likely around the value x. Therefore, we distribute the mixed channel state in two shares according to the ratio x. Note that this solution is scalable. Although the undeployed routers form a subnetwork, the scheme still works conditioned that the subnetwork routes packets in a FIFO manner.

7. POSSIBLE APPLICATIONS AND LIMITATIONS

In this section, we discuss the special attack scenarios in which our proposed approach can be useful. Also, we present some of the attack scenarios that our proposed approach cannot be applied.

7.1. Applications

Bursty attack traffic with long duration. Attack traffics can be sent in a long duration in a bursty manner. For example, within 30 min, an attack source sends out spikes of packets for a duration of 10 ms in a period of 1 s. In this case, one can apply our approach for a longer time interval, e.g. 10 s, in order that the victim can capture several spikes from the attack flow, and one can then locate the attackers accordingly.

7.2. Limitations

Global scale attack. In the real attack scenarios like the Code Red worm [19], the worm affects machines by exploiting vulnerabilities in softwares or the operating system. The

number of infected machines were numerous (in the order of 100,000) and infected machines spread across the world causing a worldwide scale attack.

In the Code Red case, every infected machine sends a small amount of packets to the victim and the overall flow is tremendous due to the huge number of infected machines. If one deploys our approach to traceback the locations of the attackers, one can only find tens of thousands of small local traffics sending towards the victim without discovering any dominating flows. This traceback result would only suggest that a worldwide scale attack is launched against the victim.

Reflector attack. The reflector attack [20] also utilizes spoofed packet but it is used in a different manner. The source address of the spoofed packets is set to be the victim's address, and these packets are sent to innocent machines, the *reflectors*. In response, the reflectors send replies to the victim. For example, spoofed HTTP requests are sent to several web servers for htdocs with large size. Thus, a small amount of spoofed requests can result in a large aggregate flow. If one applies our approach, one can only find the reflectors as the attack source and cannot find the 'real' attackers directly. One can locate the 'real' attack sources if one traces attack packets from the reflectors with source address set to be victim's address.

Attack on the infrastructure. There are real attack targets on the Internet infrastructure. For example, the Slammer worm [21] does not contain any malicious payload, but its blitz tactic in scanning for vulnerable machines has already caused hundreds of thousands of machines disconnected from the Internet. Since there is no specific victim in the attack, our approach cannot be applied.

Traceback after-the-fact. If there are attacks such as a very small amount of packets, e.g. one packet can already bring down a victim, our approach is futile in tracing this kind of attackers. Our approach works under flood-based attack and depends on a continuous flow of attack packets in order that the routers can capture the flow within the snapshot interval. Thus, a very small amount of attack packets cannot be detected by our approach.

Low-rate TCP attack. The low-rate TCP attack [22] is a new kind of DoS attack that is carefully orchestrated to exploit the fixed minimum TCP retransmission timeout. Although this kind of attack is a burst type attack, it is carefully orchestrated so as to exploit a defect in a TCP connection. Our approach is not able to detect this kind of attack. Currently, there are already several approaches [23, 24] to solve this problem.

8. RELATED WORK

One major security problem of the IP protocol is the source address can be filled by any user [25]. In DDoS attack, the attackers exploit this vulnerability in order to hide their existences as well as to hinder the authority to trace the attack origins. In the literature, work is done to mitigate effect of the DDoS attack by filtering the malicious packets before they can reach the victim [26, 27, 28, 29]. On the other hand, there are also research work going on in order to traceback the sources of the attack in the presence of spoofed packets [10, 11, 30, 31, 32].

8.1. Packet filtering

One possible way to stop the attackers from spoofing the source address of the malicious packet is the ingress filtering [26]. Under such a filtering mechanism, a router is configured to drop packets that arrive with illegitimate addresses. This requires the participating routers to have the ability to examine every packet that passes through it as well as sufficient knowledge to distinguish between the legitimate packets and the illegitimate packets. The best way to deploy the ingress filtering is at the border of an AS/ISP because it is rather easy for the border routers of the ASes and the ISPs to acquire the range of legitimate addresses.

However, the fatal problem of the ingress filtering is it requires a widespread deployment in order that the mechanism can efficiently remove most malicious packets. Unfortunately, a significant fraction of the ISPs does not implement this approach. Moreover, although the ingress filtering is deployed globally, an attacker can still launch an attack by setting the spoofed address to be a member of the legitimate address range of the AS. On the other hand, under the egress filtering [28], a router is commanded to drop packets that leave routers with illegitimate addresses. However, one can notice that this mechanism bears the same defect of the ingress filtering.

Park and Lee have proposed the route-based distributed packet filtering scheme in [27]. For example, let AS 1, AS 2 and AS 3 be three distinct autonomous systems. Under normal situation, *AS 2* receives and routes the packets from *AS 1* at its incoming interface. If an attacker at *AS 1* sends a spoofed packet with its source address belonging to *AS 3*, based on the routing information of *AS 2*, this packet is an abnormal packet and it will then be dropped. The authors analyze the distributed packet filtering scheme on the *power-law*-based Internet. The performance result shows that the main advantage of the proposed scheme is it does not require a global deployment and can still filter a significant fraction of the malicious packets.

Mahajan *et al*. have proposed the ACC in [18]. The suggested method modifies the router's congestion control algorithm, and it is two-folded. Firstly, every router is equipped with the *local ACC* which can (i) identify a congestion, (ii) classify and identify 'bad' traffic aggregates from the input queue of the router, (iii) rate-limit the arrivals in order to ease the congestion and (iv) send push-back messages to upstream routers in case that the congestion

cannot be eased with the local ACC alone. Secondly, the *push-back mechanism* is rather passive than the active local ACC measure. Since the congestion can only be detected on downstream routers, the upstream router will be invoked to launch the local ACC and rate-limits the aggregates specified by the push-back messages.

Chen and Song [33] apply the ACC to mitigate the DDoS attack. The core of the DDoS attack defense is the ability to detect the high bandwidth aggregate which is leverage on the ACC technique. The authors suggest that the defense should be taking place on the edge routers of an ISP, and the edge routers together form a defense perimeter. The perimeter is then responsible for locating the packets belonging to the high bandwidth aggregate. Once those packets are found, the edge routers that located those packets accordingly install a rate-limit filter which drops the packets according to an acceptance rate. The authors have proposed two solutions in locating which edge router(s) is/are accepting the problematic aggregate: one is done by multicasting and the other is done by IP traceback.

Xu and Lee [34] suggest a methodology to sustain the availability of web services under a DDoS attack. The goal of the defense system is to, firstly, defend against attacks using spoofed addresses, and secondly, minimize the system resources consumed by adversary using legitimate addresses. To get rid of the spoofed-address traffic, the authors suggest using the HTTP redirect message. To mitigate the damage brought about by the adversary traffic using legitimate addresses, the system is modeled as a minimax game. The goal is to maximize the small traffic and to penalize the large traffic by suspending the connection.

8.2. IP traceback

Savage *et al*. have proposed the probabilistic packet marking scheme in [10]. Every router, which participates in this scheme, marks the IP header of every packet and passes through it based on a pre-defined probability. If a packet is already marked by an upstream router, the router will not mark that packet again. At the victim site, the victim can reconstruct the packet path (or the attack path) by collecting sufficient number of marked packets from the routers. In [35], the authors have analyzed the time as well as the number of packets that are sufficient to construct the attack graph with certain confidence interval. In [11], the authors have proposed an authentication scheme based on the approach suggested by Savage. The aim of this work is to hinder the malicious parties to alter the marked field in the IP header of the packets. Also, the authors have mentioned that if the victim site knows the map of its upstream routers, it does not need the full IP address in the packet marking. They improved Savage's marking approach by hashing so as to achieve a lower false positive rate and a lower computation overhead. On the other hand, Park and Lee have analyzed this marking approach and pointed out that spoofing of the marking field may impede traceback by the victim site [31]. Attackers may choose the spoofed marking value, source address to hide themselves.

Despite the IP traceback approach proposed by Savage, Dean *et al*. [32] have formulated the traceback problem as a polynomial reconstruction problem. They use algebraic coding theory to encode traceback information in the packet, similar to Savage's approach. On the other hand, Snoeren *et al*. [30] have proposed an efficient hash-based approach to traceback the attackers. Every packet that passes through a router is hashed into the storage device associated with the router. By tracking the storage device of every router, one can derive the traversed path of a single packet. Adler [36] has formulated a new IP traceback scheme that is capable of tracing the attacker with an arbitrary number of encoding bits in the attack packets. According to the analysis, one can apply an IP traceback scheme that uses one encoding bit per packet under a single-attacker environment. However, the lower bound of the number of bits required is greater than one under a multiple-attacker environment. The author also gives the analysis of the lower bound of the number of bits required in [36].

Sung and Xu is the first piece of work that combines an IP traceback approach and an automatic packet filtering approach together [37]. The scheme employs the IP traceback approach, namely the probabilistic packet marking algorithm, to discover the attack path. Then, by setting up a defense perimeter, the attack packets are filtered preferentially at the routers that are far from the victim. By using this scheme, the attack packets can be filtered at a far distance from the victim while the legitimate packets can reach the victim instead of being dropped.

Unlike the probabilistic packet marking algorithm which marks packets randomly, Belenky and Ansari [38] suggest marking every packet that passes through the edge routers of an ISP. Then, by collecting the marked packets, one can know from which edge router the attack traffic is coming from. For more details on the IP traceback schemes, readers can refer to the detailed surveys in [39, 40].

9. CONCLUSION AND FUTURE WORK

9.1. Conclusion

In this paper, we proposed a distributed traceback methodology for DDoS attack such that a victim site can locate attackers who sent dominating flows of attack packets. We illustrated that a router only needs to keep track of (i) the number of packets forwarded to a victim site and (ii) the number of transit packets for all its incoming links during the recording of the router's local state. By providing these two pieces of information, a victim site can accurately

determine the intensity of a router's local traffic. We also presented an efficient algorithm so that a victim site can accurately determine the bounds of the number of packets from each router which arrived during the victim's measurement interval. Based on this information, the victim can determine the locations of attackers with dominating flows no matter the attack packets are spoofed or non-spoofed within a short measurement interval. We carried out simulations to show that the proposed traceback methodology performs efficiently and is able to locate the attackers sending out large flows. Moreover, we discussed the limitations of our traceback methodology. We believe that the proposed distributed traceback methodology can complement and leverage on the existing ICMP traceback so that a more efficient and accurate traceback can be obtained.

9.2. Future work

The proposed approach still requires some extra features including the *distributed authentication* and the *packet dropping calculation*. In our primary approach, we allow any parties to register the traceback service. However, an attacker can exploit this vulnerability by generating arbitrary requests to the routers in order to initiate another level of denial of service attack. Thus, the distributed authentication among the group of participating routers is urgently necessary. One of the potential solutions is to adopt the group key management approach such as [41]. Every victim site before registering the service has to verify its identity. Thus, one can stop attackers from exploiting the traceback approach.

On the other hand, the original snapshot algorithm proposed in [6] assumes the network (or the channels) is reliable, i.e. without packet loss. However, in the DDoS scenarios, the packet loss rate is huge because the routers are usually filled up with malicious packets and is forced to drop further incoming packets. The packet dropping process may ruin the traceback result because the packets that are counted in the upstream routers may not reach the downstream routers and the victim site. A primary approach is to measure the packets dropped at each router's interface and make use of the packet dropping counter to complement the equations in calculating the local traffic.

Lastly, we believe that our proposed approach is not only providing a theoretical foundation in traffic measurement, but also foreseeing that our approach can become a form of network tomography [42]. This requires further research in finding possible applications.

ACKNOWLEDGEMENTS

We thank the editor for coordinating the review process. Also we thank anonymous reviewers for insightful comments and constructive suggestions. The work of John C.S. Lui is supported in part by the Earmarked Grant 4180/03E. The work of M.H. Wong is supported by CUHK 4208/04E.

REFERENCES

[1] CERT Coordinate Center (2000) CERT Advisory CA-2000-01: denial-of-service developments. Available at http://www.cert.org/advisories/CA-2000-01.html.

[2] Taylor, D. E., Lockwood, J. W., Sproull, T. S., Turner, J. S. and Parlour, D. B. (2002) Scalable IP lookup for programmable routers. In *Proc. IEEE Infocom.,* New York, June 23–27, pp. 526–571. IEEE, Washington, DC.

[3] Peterson, L. L., Karlin, S. and Li, K. (1999) OS support for general-purpose routers. In *Proc. Workshop on Hot Topics in Operating Systems,* Rio Rico, AZ, March 29–30, pp. 38–43. IEEE, Washington, DC.

[4] Qie, X., Bavier, A., Peterson, L. and Karlin, S. (2001) Scheduling computations on a software-based router. In *Proc. ACM SIGMETRICS*, Cambridge, MA, June 16–20, pp. 13–24. ACM Press, New York, NY.

[5] Yau, D. K. Y. and Chen, X. (2001) Resource management in software-programmable router operating systems. *IEEE J. Sel. Area. Commun.*, **19**, 488–500.

[6] Chandy, K. M. and Lamport, L. (1985) Distributed snapshots: determining global states of distributed systems. *ACM Trans. Comput. Syst.*, **3**, 63–75.

[7] Internet Mapping Project (1999) Available at http://research.lumeta.com/ches/map/index.html.

[8] Bellovin, S. M.(ed.) (2000) *ICMP Traceback Messages, Internet Draft: draft-bellovin-itrace-00.txt*.

[9] Cooperative Association for Internet Data Analysis. (1997) Available at http://www.caida.org/.

[10] Savage, S., Wetherall, D., Karlin, A. and Anderson, T. (2000) Practical network support for IP traceback. In *Proc. 2000 ACM SIGCOMM Conf.*, Stockholm, Sweden, August 28–September 1, pp. 295–306. ACM Press, New York, NY.

[11] Song, D. X. and Perrig, A. (2001) Advanced and authenticated marking schemes for IP traceback. In *Proc. IEEE INFOCOM '01,* Anchorage, AK, April 22–26, pp. 878–886. IEEE, Washington, DC.

[12] Wu, S. F., Zhang, L., Massey, D. and Mankin, A. (2001) *Intention-Driven ICMP Trace-Back, Internet Draft: draft-wu-itrace-intention-00.txt*.

[13] Mankin, A., Massey, D., Wu, C.-L., Wu, S. F. and Zhang, L. (2001) On design and evaluation of intention-driven ICMP traceback. In *Proc. IEEE Int. Conf. Computer Communications and Networks*, Scottsdale, AZ, October 15–17, pp. 159–165. IEEE, Washington, DC.

[14] Barry, D. (2004) Proactive protection: new techniques and best practices help service providers counter increase in cyber attacks. *Packet: Cisco Systems Users Magazine*, **16**, 64–68.

[15] Chan, B. C., Lau, J. C. and Lui, J. C. (2005) OPERA: an open-source extensible router architecture for adding new network services and protocols. *J. Syst. Softw.*, **78**, 24–36.

[16] The netfiler/iptables projects. (2000) Available at: http://www.netfilter.org.

[17] Harris, J. and Melara, A. J. (2002) Performance analysis of the Linux Firewall in a host. CiNIC—Calpoly intelligent NIC Project. Available at http://www.ee.calpoly.edu/3comproject/.

[18] Mahajan, R., Bellovin, S. M., Floyd, S., Ioannidis, J., Paxson, V. and Shenker, S. (2002) Controlling high bandwidth aggregates in the network. *ACM SIGCOMM Comput. Commun. Rev.*, **32**, 62–73.

[19] Zou, C. C., Gong, W. and Towsley, D. (2002) Code red worm propagation modeling and analysis. In *Proc. 9th ACM Conf. Computer and Communications Security*, Washington, DC, USA, November 18–22, pp. 138–147. ACM Press, New York, NY.

[20] Paxson, V. (2001) An analysis of using reflectors for distributed denial-of-service attacks. *ACM SIGCOMM Comput. Commun. Rev.*, **31**, 38–47.

[21] Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S. and Weaver, N. (2003) Inside the Slammer Worm. *IEEE Secur. Privacy*, **1**, 33–39.

[22] Kuzmanovic, A. and Knightly, E. W. (2003) Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. In *Proc. ACM SIGCOMM 2003*, Karlsruhe, Germany, August 25–29, pp. 75–86. ACM Press, New York, NY.

[23] Shevtekar, A., Anantharam, K. and Ansari, N. (2005) Low rate TCP denial-of-service attack detection at edge routers. *IEEE Commun. Lett.*, **9**, 262–265.

[24] Sun, H., Lui, J. C. S. and Yau, D. K. Y. (2004) Defending against low-rate TCP attack: dynamic detection and protection. In *Proc. IEEE Int. Conf. Network Protocols (ICNP)*, Berlin, Germany, October 5–8, pp. 196–205. IEEE, Washington, DC.

[25] Bellowin, S. M. (1989) Security problems in the TCP/IP protocol suite. *ACM Comput. Commun. Rev.*, **19**, 32–48.

[26] Ferguson, P. and Senie, D. (1998) RFC 2267: network ingress filtering: defeating denial of service attacks which employ IP source address spoofing. Network Working Group, Request for Comments: 2267. Available at http://www.faqs.org/rfcs/rfc2267.html.

[27] Park, K. and Lee, H. (2001) On the effectiveness of route-based packet filtering for distributed DoS Attack prevention in power-law internets. In *Proc. SIGCOMM*, San Diego, CA, August 27–31, pp. 15–26. ACM Press, New York, NY.

[28] Global Incident Analysis Center Egress Filtering v 0.2 2000 Available at http://www.sans.org/y2k/egress.htm.

[29] Yau, D. K. Y., Lui, J. C. S., Liang, F. and Yeung, Y. (2005) Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *IEEE/ACM Trans. Netw.*, **13**, 29–42.

[30] Snoeren, A. C., Partridge, C., Sanchez, L. A., Jones, C. E., Tchakountio, F., Kent, S. T. and Strayer, W. T. (2001) Hash-based IP traceback. In *Proc. ACM SIGCOMM 2001 Conf. Applications, Technologies, Architectures, and Protocols for Computer Communication*. San Diego, CA, August 27–31, pp. 3–14. ACM Press, New York, NY.

[31] Park, K. and Lee, H. (2001) On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack. In *Proc. IEEE INFOCOM '01*, Anchorage, AK, April 22–26, pp. 338–347. IEEE, Washington, DC.

[32] Dean, D., Franklin, M. and Stubblefield, A. (2001) An algebraic approach to IP traceback. In *Proc. Network and Distributed System Security Symp., NDSS '01*, San Diego, CA, February 7–9, pp. 3–12. Internet Society, ISOC, Reston, VA.

[33] Chen, S. and Song, Q. (2005) Perimeter-based defense against high bandwidth DDoS attacks. *IEEE Trans. Paral. Distrib. Syst.*, **16**, 526–537.

[34] Xu, J. and Lee, W. (2003) Sustaining availability of web services under distributed denial of service attacks. *IEEE Trans. Comput.*, **52**, 195–208.

[35] Law, K. T., Lui, J. C. S. and Yau, D. K. Y. (2005) You can run, but you can't hide: an effective methodology to traceback DDoS attackers. *IEEE Trans. Paral. Distrib. Syst.*, **15**, 799–813.

[36] Adler, M. (2005) Trade-offs in probabilistic packet marking for IP traceback. *J. ACM*, **52**, 217–244.

[37] Sung, M. and Xu, J. (2003) IP traceback-based intelligent packet filtering: a novel technique for defending against internet DDoS attacks. *IEEE Trans. Paral. Distrib. Syst.*, **14**, 861–872.

[38] Belenky, A. and Ansari, N. (2003) IP traceback with deterministic packet marking. *IEEE Commun. Lett.*, **7**, 162–164.

[39] Belenky, A. and Ansari, N. (2003) On IP traceback. *IEEE Commun. Mag.*, **41**, 142–153.

[40] Gao, Z. and Ansari, N. (2005) Tracing cyber attacks from the practical perspective. *IEEE Commun. Mag.*, **43**, 123–131.

[41] Alves-Foss, J. (2000) An effcient secure authenticated group key exchange algorithm for large and dynamic groups. In *Proc. 23rd National Information Systems Security Conf.*, October.

[42] Rabbat, M., Nowak, R. and Coates, M. (2004) Multiple source, multiple destination network tomography. In *Proc. IEEE Infocom 2004*. Hong Kong, China, March 7–11, pp. 1628–1639. IEEE, Washington, DC.