Optimal Filter Assignment Policy Against Distributed Denial-of-Service Attack

Rajorshi Biswas and Jie Wu
Department of Computer and Information Sciences
Temple University, Philadelphia, PA, USA
{rajorshi, jiewu}@temple.edu

Abstract—A distributed denial-of-service (DDoS) attack is a cyber-attack in which attackers from different locations send out many requests to exhaust the capacity of a server. Current DDoS attack protection services filter out the DDoS attack packets in the middle of the path from the attacker to the servers. Some of the DDoS protection systems filter them out at the victim server. As a result, unnecessary attack traffic congests the network and wastes bandwidth. This can be minimized if we block them as early as possible. In this paper, we propose a DDoS attack protection system by using the filter router. The victim needs to wisely select and send filters to a subset of filter routers to minimize attack traffic and blockage of legitimate users (LUs). Many filters can easily minimize the attack traffic and blockage of LUs, but it is costly to the victim. So, we formulate two problems with different settings for selecting filter routers given a constraint on the number of filters. We propose dynamic programming solutions for both problems. Both problems consider the blockage of all attack traffic before it reaches the victim. We conduct extensive simulation to support our solutions.

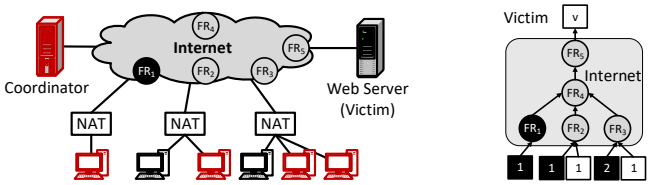
Index Terms—*botnet, DDoS defense, DDoS, flooding attack, filter router, network security, filter assignment*

1 Introduction

A denial-of-service attack (DoS attack) is a cyber-attack in which the attacker seeks to make a machine (e.g., web server) or network resource temporarily unavailable to its users. DoS attacks are considered a federal crime under the Computer Fraud and Abuse Act with penalties that include years of imprisonment [1]. The Computer Crime and Intellectual Property Section of the US Department of Justice handles cases of DoS attacks. Therefore, detecting DoS attacks and identifying attackers have been important issues in Network Forensics. Moreover, DoS attacks are increasing day by day in both number and size; CloudFlare [2] recently reported a 400 Gbps massive DoS attack that took place in their servers.

There are several types of DoS attacks such as SYN Floods, Malformed Packets, UDP Floods, Amplification Attacks, and Distributed Attacks [3]. In a SYN Flood attack, the perpetrator sends many SYN messages to set up TCP connection. The server replies ACK and waits for the client's ACK, but the attacker does not reply ACK and the connection remains half-open till timeout. The objective of a SYN flood is to simply fill up the limited slots that the target system has available for half-open connections. In some cases, it's easy to detect a SYN Flood attack if a lot of SYN requests come from an address in short interval. Detection is harder, however, when the attacker spoofs IP address, SYNs come from multiple addresses, and arrival time varies. In a UDP Flood attack, the purpose would likely be to consume all available network bandwidth. Attackers send a large amount of spoofed requests with large useless payloads. The application wastes CPU cycles trying to determine the meaning of the garbage.

The objective of the DDoS attack is to generate a lot of packets from different locations to exhaust the incoming/outgoing bandwidth of the victim (e.g., web server). A coordinator would send commands to workers, who continue to send requests to the target. The workers are known as bots and the network of workers is known as botnet. As users also send requests through the NAT, it is difficult for the victim to differentiate between the bot requests and user requests. Fig. 1(a) shows the DDoS attack model by a botnet.



(a) DDoS attack by bots. (b) Topology.

Fig. 1: DDoS attack and traffic topology.

The existing works which are based on DDoS detection at the router level increase router computation overhead. The works which are based on filtering at the victim increase the network overhead. The routers that detect DDoS traffic based on some generalized characteristics cannot detect DDoS traffic better than the victim. Besides, the characteristics of DDoS traffic are different for different victims. An effective method of preventing DDoS attacks is to use filter routers (FRs) in the network infrastructure. FRs are special types of routers that are capable of packet marking and receiving filter tasks. Marking a packet means appending the FR's IP address to the packets it forwards. A FR does not mark all the packet it forwards, rather it probabilistically selects some packets to mark. The task of receiving filter refers to receiving a filter from a web server. A web server can block all or part of the traffic destined to it. The FR-based system does not increase the router computation overhead

and network overhead. The victim can accurately differentiate between the attack and legitimate traffic because it knows the characteristics of the user traffic very well.

The complete method of using FRs is a four-phase process. In the first phase, the FRs probabilistically mark forwarded packets by appending their own IP addresses. In the second phase, the victim discovers the traffic topology from the marking of the packets. The marked packet contains the FR's IP addresses and the sequence of the IP addresses implies their relative positions. After collecting a sufficient number of marked packets the victim discovers the traffic topology. Fig. 1(b) shows the traffic topology which is discovered by the victim v. In the third phase, the victim constructs filters, then finds and selects some FRs to assign the filters. In the last phase, the FRs evict unused filters from their storage.

Packet marking is used by the victim to find the traffic topology. There is a tradeoff between topology construction time and router overhead. When the probability of marking is low, the router overhead is low and it takes a long time to construct the topology. When it is high, the router overhead is high and the topology construction is quick. After topology construction, the victim generates filters and selects a subset of the FRs to assign them. There are two types of filters: source-based and destination-based. Using a source-based filter, a FR can allow/block traffic from specific sources that are destined for the victim. This type of filter is vulnerable to IP spoofing attacks. Although destination-based filters can prevent IP spoofing, they are more restrictive. Using a destination-based filter, a FR can block all traffic, including LU traffic destined to the victim. A good filter assignment can stop the attack traffic close to its source, which reduces network overhead for attack traffic. A bad filter assignment can let the attack traffic travel a longer way and produce unnecessary network overhead. It is challenging to find an optimal filter assignment when the victim has a limited number of FRs that can be selected. A FR may get filters from multiple victims. It has limitations on storage and computation power. Therefore, it evicts filters which are no longer used.

In this paper, we focus on finding the optimal filter assignment considering that the victim has already constructed the traffic topology. We formulate two problems and propose solutions for them. In the first problem, a limited number of source-based filters are assigned to the FRs. For example, if the victim can assign 2 filters, it can select $\{FR_1, FR_2\}$, $\{FR_1, FR_5\}$, $\{FR_2, FR_4\}$ or another pair of FRs (see Fig. 1). If the victim selects the first pair of FRs, then no attack traffic can get into the network, which is highly expected. If the second pair is selected, then the attack traffic will travel through the (FR_2, FR_4) and (FR_4, FR_5) links. The amount of attack traffic in each link is not the same. It is challenging to find a filter assignment for which the total amount of attack traffic is the minimum. We propose an optimal solution for this problem by using dynamic programming. In the second problem, a limited number of destination-based filters are assigned to the FRs. A destination-based filter blocks every packet at the FR that is destined to the victim. If the victim selects the third pair, then all the legitimate users (LUs) will be blocked and the attack traffic will travel through the (FR_1, FR_4) link. It is also challenging to find a filter assignment so that the total attack traffic and the number of blocked LUs are both minimized. We propose another dynamic programming solution for this problem. Our main contributions are the following:

1) We formulate two problems for finding filter assignments with a limited number of filters and provide optimal solutions using dynamic programming.
2) We conduct extensive simulations with synthetic and real topology datasets and present simulation results to support our model.

The remainder of this paper is arranged as follows: In Section 2, we discuss some related works and their limitations. In Section 3, we present the system model for preventing DDoS attacks. In Section 4, we define the first problem and propose a dynamic programming solutions. In Section 5, we define the second problem and propose another dynamic programming solution. In Section 6, we present some simulation results that strengthen our proposed solutions. Finally, Section 7 concludes our paper.

2 Related Works

There exist many statistical methods, including correlation, entropy, covariance, divergence, cross-correlation, and information gain to detect anomalous DDoS requests [4]. A rank correlation-based method, Rank Correlation-based Detection (RCD), is proposed in [5]. An information theoretical approach using Kolmogorov complexity is used for the detection of DDoS attacks in [6]. A novel DDoS detection mechanism is proposed based on Artificial Neural Networks in [7]. A spiking neural network-based intrusion detection system is proposed in [8]. The neural network evolves over time to adopt with the behavior of the inputs. The spiking neural network cube learns to activate the neuron based on input data using unsupervised learning. The spiking neurons are trained to generate the output module using supervised learning. In [9], the authors present several DDoS mitigation approaches including prevention of JavaScript bot code injection and PHP-sensor-based scheme to identify cross-site-scripting attacks. In [10], the authors propose a novel flow-table sharing approach to protect against the table overloading DDoS attacks in SDN-based networks. There are other methods of detecting DDoS attacks, including [11–17].

The authors in [18] introduced a model of randomized DDoS attacks with an increasing emulation dictionary where the attackers use the attack definition from the dictionary that contains request patterns similar to those of the LUs. They proposed an inference algorithm for identifying the botnets executing such DDoS attacks. Nowadays, static path identifiers are used for inter-domain routing objects, which helps the attackers to launch DDoS flooding attacks. In [19], the authors present a design dynamic path identification framework that uses path identifier negotiated between the neighboring domains as inter-domain routing objects.

In [20], the authors propose a method, RADAR, to detect and throttle DDoS attacks using adaptive correlation analysis on SDN switches. The system can defend against a wide range of flooding-based DDoS attacks including link flooding, SYN flooding, and UDP-based amplification

TABLE 1: Table of notations

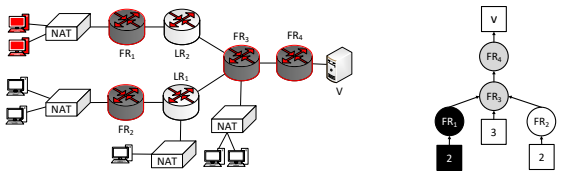
N	Number of nodes
v	Victim
B	Total attack traffic
K	Number of filters
A_1	Contains contamination for Problem 1
A	Contains cost for Problem 2
$C_\delta(i)$	δ th child of node i
Δ	Maximum node degree of the topology
$AL[i]$	Contains attack traffic in subtree rooted by i
$L[i]$	Contains LU traffic in subtree rooted by i
R_1	Contains filter assignment for Problem 1
R_2	Contains blocked traffic for Problem 1
R	Contains filter assignment for Problem 2
G	Graph containing black and gray nodes
g	Assignment set
U	Set of LUs
U_b	Set of blocked LUs
$W_c(g)$	Contamination after applying g filters
$C(g)$	Cost after applying g filters in Problem 2
ω	Weight of contamination in Problem 2

attacks. In [21], the authors propose a new approach which minimizes the resource utilization factor for quick absorption of the attack. In [22], a DDoS protection mechanism called SkyShield, is proposed by taking advantage of the sketch techniques. To identify malicious hosts efficiently, they used the abnormal sketch obtained from the last detection cycle. SkyShield can leverage other techniques including Bloom filters and CAPTCHA. In [23], the authors propose a collaborative DDoS mitigation network system in which one domain helps another domain. A domain can direct excessive traffic to other trusted external domains for DDoS filtering. The filtered clean traffic is then forwarded back to the targeted domain. A three-tier datacenter design is proposed in [24]. In this design, the first two tier of private datacenters filter out the attack traffics and forward the legitimate traffic to the third tier datacenter. Therefore, the attack traffic are being filtered at the destination datacenter which is usually far from the attackers' locations.

Most existing works are mainly concerned about the availability of the server. In fact, the attack traffic may cause huge network congestion and DoS. Therefore, these techniques cannot protect the network from being contaminated by attack traffic. A victim and network component collaboration based system can help in this case. A four-phase DDoS protection system is proposed in [25]. The victim generates filters and sends them to the upstream FRs. FRs send the filters to their upstream FRs and thus the filters propagate to the effective FRs. An adaptive version of PFS is proposed in [26]. The system directly sends filters to the highly capable FRs first, then the filters propagate to the effective FRs. However, these two systems cannot optimally select the FRs when there is a limitation on selecting FRs.

3 System Model

Our system is composed of legacy routers (LRs), network address translators (NATs), filter routers (FRs), attackers, legitimate users (LUs), and a victim (v). Fig. 2 shows the complete system model. In reality, there are multiple victims in a network but for simplicity of explanation, we are considering a single victim. The end users are connected to a FR or a LR through NAT. Nowadays, because of the limited number of public IP addresses, the internet service



(a) Actual network. (b) v's view.

Fig. 2: System model.

providers usually assign private IP addresses to their customers. Therefore, most of the users are connected to the internet through NATs. A FR is a special kind of router which is capable of two functionalities. Firstly, it can do packet marking, which is used to construct traffic topology at the victim. Secondly, it can receive filters from the victim and apply the filters to block the attack traffic according to the filter definitions. There are two types of filters: source-based and destination-based. The source-based filter specifies the blocking of traffic based on the source address. For example, a source-based filter can be understood with: *if source address is X, then discard the packet*. If we use a source-based filter at FR_3 (assume X and Y are the IP addresses of the NATs connected to FR_1 and FR_2, respectively), then FR_3 will discard packets coming from NAT-X, but forward packets from NAT-Y.

The advantage of using source-based filters is that a FR can block the attack traffic by its source IP address and forward legitimate traffic. If the LU and attacker both remain behind the same NAT, then it is impossible to block only attack traffic. The limitation of the source-based filter is that it cannot protect if an attacker spoofs the IP address of a LU. If an attacker creates a packet with Y as the source address, then the packet will not be blocked at FR_3. To protect against DDoS attacks with IP spoofing, we can use destination-based filters. A destination-based filter is: *if the destination address is X, then discard the packet*. For example, if we use a destination-based filter at FR_3 (assume that X is the IP address of v), then all the packets, including legitimate and spoofed attack packets, will be blocked by FR_3. The destination-based filter is more restrictive. When a FR uses it, it blocks all the attack and legitimate traffic destined for the victim. Therefore, spoofed attack traffic cannot penetrate.

In this model, the LRs and the FRs can co-exist without any problem. For example, in Fig. 2(a), there are some LR between the FRs. In the victim's view 2(b) there are no LRs. This is because the LRs do not mark the forwarding packets and their existence is not identified by the victim. The implementation of the FRs can also be done with small changes. The firmware of the routers can be updated to adopt the packet filtering and marking functionalities. Nowadays, many routers and software defined networking switches run on Linux systems. These routers and switches have support for custom developed plugins (for example, Pica8 p-3922). A plugin can be developed to implement the filtering and marking functionalities which will turn a regular a software-defined-networking switch into a FR.

The attackers are usually user devices with compromised programs that can generate traffic destined for a target. The programs are controlled by a master. The master can send attack commands to the programs. This type of

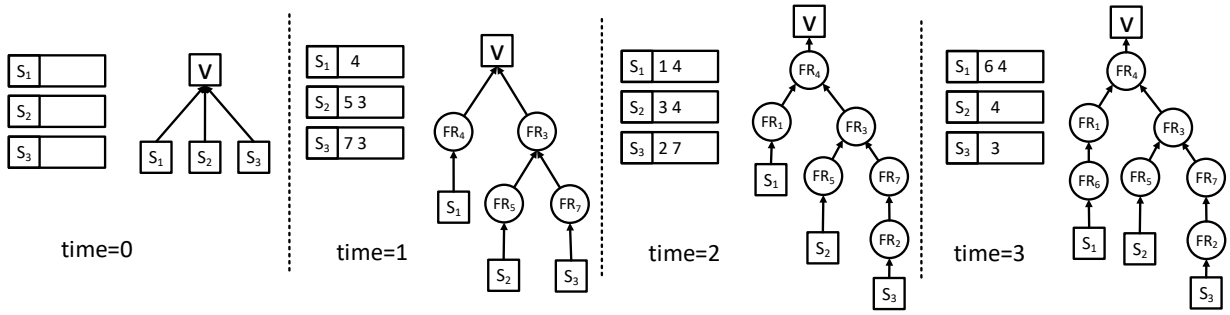


Fig. 3: Formation of topology.

program is called bot, and a network of bots is called a botnet. Although it is hard to differentiate the DDoS traffic from legitimate traffic, there exist several methods based on arrival time, packet size, and packet content for detecting attack packets [4]. In this paper, we are not focusing on the detection of attack packet. The victim finds the source address of attackers using these methods. The victim also knows the packet rate of each attacker. For example, if there are 100 attackers (or LUs) each with 1 Mbps attack traffic (or legitimate traffic) behind a NAT-X (having IP X), then the victim will identify X as an attacker (or LU) with 100 Mbps attack traffic (or legitimate traffic).

The complete protection process consists of four phases.

1) **Packet marking by FRs:** The process of probabilistically appending the FR's own IP address in a special field of packet header.
2) **Construct traffic topology:** The process of a victim constructing the topology from the appended IP addresses of packets it receives.
3) **Construct filter and assign to FRs:** After analyzing the received packets over a period, the victim can identify the IP addresses of the attacker. Then, it needs to find some FRs and send some filters to block the attack traffic.
4) **Evict unused filter from FR:** A FR has limited storage. When a filter is not used for a period of time, that FR evicts the filter from its storage.

In **phase 1**, the FR probabilistically marks the packets it forwards by appending its own IP address. The algorithm is simple; the router picks a random number and if the number is less than the marking probability, then it decides to mark the packet. Marking a packet means appending the FR's IP address to the header of it. The marking probability is the probability of a packet being appended by the FR's IP address. For example, if the marking probability of a FR is 0.5, then on average 50% of the packets forwarded by that FR contain the IP address of the router in its header. Let the marking probability be 0.5. Then, the victim v may get packets with $\{FR_1, FR_3\}$, $\{FR_3, FR_4\}$, or $\{FR_1, FR_4\}$. The victim may also get packets with $\{FR_2, FR_3\}$, $\{FR_3, FR_4\}$, or $\{FR_2, FR_4\}$. The $\{FR_1, FR_3\}$ marking indicates that FR_1 remains before the FR_3 along the path from the user.

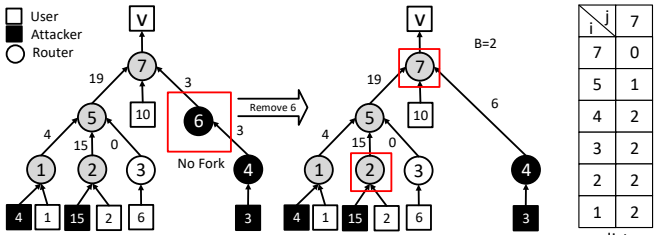
In **phase 2**, the victim constructs paths from all the sources after gathering enough information from the marked packets.The victim can easily form a directly acyclic graph (DAG) combining all the paths. Fig. 3 shows an example of topology formation technique. In the beginning (time=0), there is no knowledge about topology. The victim assumes that all users are directly connected with it because none of the packets are marked by any FRs. At time 1, the victim gets some packets from S_1, S_2, and S_3. These packets are marked by FRs $\{FR_4\}$, $\{FR_5, FR_3\}$, and $\{FR_7, FR_3\}$, respectively. The packet from S_1 indicates that FR_4 remains between S_1 and v ($S_1 \rightarrow FR_4 \rightarrow v$). Similarly, the packets from S_2 indicate that FR_3 remains before v and FR_5 remains before FR_3 ($S_2 \rightarrow FR_5 \rightarrow FR_3 \rightarrow v$). The packet from S_3 indicates that FR_3 remains before v and FR_7 remains before FR_3 ($S_3 \rightarrow FR_7 \rightarrow FR_3 \rightarrow v$). Combining all these information we can infer that FR_3 might be a parent of both FR_5 and FR_7. The victim might be connected with FR_3 and FR_4. We can also infer that S_1, S_2, and S_3 might be connected to the FR_4, FR_5, and FR_7. At time 3, more packets arrive at v and the topology changed a little bit. FR_3 is found to be a child of FR_4. A new FR_1 is discovered at this time. At time 4, the topology changed with an addition of FR. We can see that the more v observes, the more knowledge it gathers about topology, and the closer it gets to the actual topology. Thus the victim can formulate a DAG of the traffic topology.

For simplicity, we will consider a tree instead of a DAG. We consider that bots and LUs are behind the NAT of the internet service provider. We color the bots/attackers as black and LUs as white. The FRs which forward the end users' traffic first are called *entry nodes*. $\{FR_1, FR_2\}$ are the entry nodes in Fig. 2. The FRs are colored as black, white, or gray. A black (or white) FR means it only forwards messages from attackers/bots (or LUs). A gray FR forwards packets from both LUs and attackers.

In **phase 3**, some of the FRs in the traffic topology are selected to assign filters. The traffic topology is simplified by removing nodes with no forks. A node with at least two children is called a fork node. Non-fork nodes are not efficient for assigning filters. Instead, selecting the child node reduces attack traffic in the network. Therefore, an optimal filter assignment policy should select a set of FRs (g) from the set of gray and black nodes (G) with minimal blockage of legitimate traffic and contamination by attack traffic, while ensuring that no attack traffic can reach the victim. We define contamination as the total attack traffic in the network. For example, if the attack traffic is blocked at FR_3, then the total contamination is 2 (all attackers' packet rates are 1). We denote the contamination in a network for the g filter assignment set by W_c.

$$W_c(g) = \sum_{a=1}^{A} \alpha_a \times d_a, \; d_a = \min_{\forall_{n \in \text{PRED}(n) \cap g}} \text{dist}(a, n). \qquad (1)$$

Here, PRED(n) is the set of predecessors of n, α_a is



(a) Topology for Problem 1. (b) dist.

Fig. 4: Topologies for problems 1 and 2.

the traffic load of attacker a, dist(a, n) is the number of hops between a and n. A is the total number of attackers. Therefore, W_c is the total attack traffic load for selecting $|g|$ FRs out of $|G|$ FRs. If U is the set of LUs, then the number of blocked LUs for the filter assignment g is denoted by $U_b(g)$.

$$U_b(g) = \{u : u \in U \text{ and } \text{PRED}(u) \cap g \neq \emptyset\}. \tag{2}$$

The best way to minimize blockage of LU and contamination is to block immediately after the attacker. In reality, there are a huge number of attackers and the victim needs to select a huge number of FRs to block them, which is not possible. So, a victim has budget K for selecting a number of FRs. Therefore, $|g|$ should be less than or equal to K.

In **phase 4**, unused filters are removed from the FRs. As the FRs have limited capacity and computation power, it is necessary to reduce the workload by removing the filters. Otherwise, the attacker can flood the FRs by sending useless filters. This types of filters are evicted soon because they are most likely not being used.

The filter sent by a user (or victim) is only applicable to the packets which are destined for that user (or victim). However, an attacker can spoof the IP of the victim and send wrong filters to FRs. This spoofed filter can be detected using a simple handshake protocol. The spoofing attacker will not be able to handshake with the spoofed IP address. However, we are focusing on finding an optimal filter assignment policy, which is discussed in the next section.

4 Source-based Filter Assignment

In this section, we formulate the problem of assigning the source-based filters to the FRs so that the contamination is the minimum.

4.1 Problem Definition

Problem 1. *Find a filter assignment so that the contamination is the minimum by ensuring that all the attack traffic is blocked before reaching the victim.*

In this problem, source-based filters are used. The contamination is defined by the total amount of attack traffic in the network. The problem can be expressed as the following:

$$\begin{aligned} &\text{minimize} && W_c(g) \\ &\text{subject to} && |g| \leq K, \forall g \subset G, v \notin G. \end{aligned} \tag{3}$$

The victim v will be white ($v \notin G$) if all attacker traffic is blocked before reaching it. We propose an optimal solution using dynamic programming.

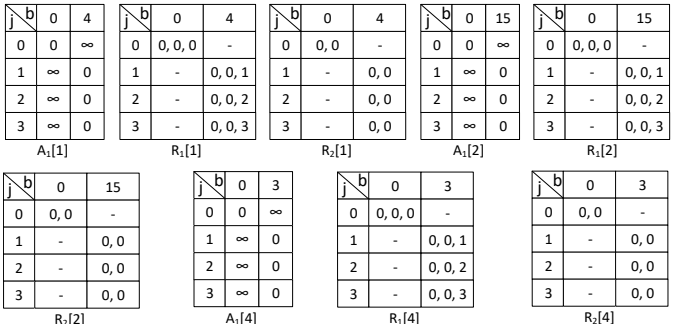


Fig. 5: Tables A_1, R_1, and R_2 for leaf nodes.

4.2 An Optimal Solution

To solve the problem, we define the following two problems.

1) $P_1(i, j, b)$: Find and return optimal contamination in the tree rooted by i for j number of filters by blocking b attack traffic. The optimal contaminations, filter assignments, and blocked attack traffics are stored in $A_1[i, j]$, $R_1[i, j]$, and $R_2[i, j]$ to reuse in dynamic programming.
2) $P_2(i, j)$: Find and return optimal contamination in the tree rooted by i for j number of filters by ensuring blockage of all attack traffic. This is the problem defined in Problem 1. $P_2(i, j) = P_1(i, j, B)$ where B is the total attack traffic in the subtree rooted by node i.

There are two options to assign filters. $P_1(i, j, b)$ is the minimum of the following two options:

Option I: The minimum total contamination, if we assign 1 filter to node i, divide the rest of the $j - 1$ filters into $j_1, j_2, ..., j_\Delta$ parts, assign the parts to the subtrees $c_1(i), c_2(i), ..., c_\Delta(i)$, and block $b_1, b_2, ..., b_\Delta$ attack traffic, respectively. Therefore, the cost will be the sum of the minimum contaminations and the contamination for unblocked attack traffic in $c_1(i), c_2(i), ..., c_\Delta(i)$. In this case, the filter assigned to i blocks all the attack traffic. Therefore, the unblocked attack traffic load is 0 for this option.

$$\begin{aligned} P_1(i, j, b) = \min_{\forall_{j_\delta, b_\delta}} \sum_{\delta=1}^{\Delta} & P_1(c_\delta(i), j_\delta, b_\delta) \\ & + dist[c_\delta(i), i] \times (AL[c_\delta(i)] - b_\delta). \end{aligned} \tag{4}$$

Here, $\forall_\delta\, b_\delta \leq b$ and $\sum_{\delta=1}^{\Delta} j_\delta = j - 1$.

Option II: The minimum total contamination, if we divide the number of filters into $j_1, j_2, ..., j_\Delta$ parts, assign them to the subtrees $c_1(i), c_2(i), ..., c_\Delta$(i), and block $b_1, b_2, ..., b_\Delta$ attack traffic, respectively. Therefore, the contamination for this option will be:

$$\begin{aligned} P_1(i, j, b) = \min_{\forall_{j_\delta, b_\delta}} \sum_{\delta=1}^{\Delta} & P_1(c_\delta(i), j_\delta, b_\delta) \\ & + dist[c_\delta(i), i] \times (AL[c_\delta(i)] - b_\delta). \end{aligned} \tag{5}$$

Here, $\sum_{\delta=1}^{\Delta} b_\delta = b$ and $\sum_{\delta=1}^{\Delta} j_\delta = j$. We take the minimum quantity from the above two options. Fig. 6(b) shows the high level recursion model of the problem.

Let us consider an N node tree with maximum node degree Δ. The nodes are labeled in bottom-up and left-right order. We define A_1 as a $N \times K \times B$ array which contains optimal contamination and unblocked attack loads for every node, budget, and blocked attack traffic. For

j \ b	0	4	15	19
0	19	∞	∞	∞
1	∞	15	4	19
2	∞	15	4	0
3	∞	15	4	0

$A_1[5]$

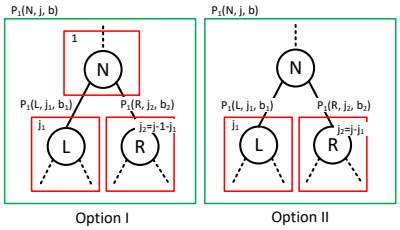
j \ b	0	4	15	19
0	0, 0, 0	-	-	-
1	-	1, 0, 0	0, 1, 0	0, 0, 1
2	-	2, 0, 0	0, 2, 0	1, 1, 0
3	-	3, 0, 0	0, 3, 0	1, 2, 0

$R_1[5]$

j \ b	0	4	15	19
0	0, 0	-	-	-
1	-	4, 0	0, 15	0, 0
2	-	4, 0	0, 15	4, 15
3	-	4, 0	0, 15	4, 15

$R_2[5]$

(a) Tables A_1, R_1, and R_2 for node 5.



(b) Recursion model for Problem 1.

j \ b	0	3	4	7	15	18	19	22
0	43	∞	∞	∞	∞	∞	∞	∞
1	∞	38	36	∞	14	∞	25	43
2	∞	38	36	30	14	30	6	14
3	∞	38	36	30	14	30	6	0

$A_1[7]$

j \ b	0	3	4	7	15	18	19	22
0	0, 0, 0	-	-	-	-	-	-	-
1	-	0, 1, 0	1, 0, 0	-	1, 0, 0	-	1, 0, 0	0, 0, 1
2	-	0, 2, 0	2, 0, 0	1, 1, 0	2, 0, 0	1, 1, 0	2, 0, 0	1, 0, 1
3	-	0, 3, 0	3, 0, 0	2, 1, 0	3, 0, 0	2, 1, 0	3, 0, 0	2, 1, 0

$R_1[7]$

j \ b	0	3	4	7	15	18	19	22
0	0, 0	-	-	-	-	-	-	-
1	-	0, 3	4, 0	-	15, 0	-	19,0	0, 0
2	-	0, 3	4, 0	4, 3	15, 0	15, 3	19,0	15, 0
3	-	0, 3	4, 0	4, 3	15, 0	15, 3	19,0	19, 3

$R_1[7]$

(c) Tables A_1, R_1, and R_2 for node 7.

Fig. 6: Recursion model and complete values of A_1, R_1, and R_2

example, $A_1[i, j, b]$ contains optimal contamination in the subtree rooted by i of budget j by blocking b attack traffic.

We define AL as a $1 \times N$ array which contains the traffic loads of attackers in subtree rooted by every node. $AL[i]$ is the attack traffic load of the attacker in subtree rooted by node i. We define R_1 as an $N \times K \times B \times (\Delta+1)$ array which contains the number of filters assigned to node i and its subtrees for every node, budget, and blocked attack traffic. For example, $R_1[i, j, b, 1]$, $R_1[i, j, b, 2]$, and $R_1[i, j, b, \Delta + 1]$ are the number of filters to the first subtree, the second subtree, and node i of subtree rooted by i for budget j by blocking b attack traffic. R_1 is the assignment according to P_1. We define R_2 as an $N \times K \times B \times \Delta$ array which contains the blocked attack traffic at optimal contamination to its subtrees for every node, budget, and blocked attack traffic. We use a double linked tree data structure. Each node contains a pointers to its parent, an array of pointer to children with distance, and the color of the node. The complete algorithm is shown in Alg. 1.

4.3 An Example

Let us consider the tree in Fig. 4(a). Firstly, we need to simplify the tree. There is only one node (node 6) without a fork. We remove node 6 and make 4 the child of its parent 7. The new distance to 4 from 7 increases by the distance of the deleted link. The deletion of a node can be done in constant time. Finding out all non-forked nodes takes $O(N)$ time. Therefore, the simplification can be done in $O(N)$ time. Next, we calculate the distance $(dist)$ of every node from the root. This calculation takes $O(N)$ as it needs to traverse the whole tree once again. A part of the $dist[i, j]$ is shown in Fig. 4(b). Next, we compute $A_1[i, j]$ and $R_1[i, j]$ for $i = 1, ..., 7$ and $j = 0, 1, ..., 3$.

Calculation for Leaf Nodes (Nodes 1, 2, and 4): $A_1[1, 0, 0]$ is 0 because if no filter is assigned to node 1, then no attack traffic is blocked, and there is no contamination in subtree rooted by 1. The attack traffic in subtree rooted by 1 does not travel through any links (there is no link). Therefore, there is no contamination in the subtree. $R_1[1, 0, 0] = [0, 0, 0]$, which means no filter is assigned to the left subtree, the right subtree, or itself. The blocked attack traffic load $R_2[1, 0, 0]$ is $[0, 0]$, as no traffic is blocked from the left and right subtrees. $A_1[1, 0, 4]$ is ∞ because if no filter is assigned to node 1, then we cannot block 4 attack traffic. Therefore, this assignment is not possible. $R_1[1, 0, 4] = [-]$ and $R_2[1, 0, 4] = [-]$, which means this option is not possible. Similarly, we can calculate the $A_1[1, j, 0]$, $R_1[1, j, 0]$, and $R_2[1, j, 0]$ where the j ranges between 1 and 3 are ∞, $[-]$, and $[-]$. $A_1[1, 1, 4]$ is 0 because, if a filter is assigned to node 1, then 4 attack traffic is blocked, and there is no contamination in the subtree rooted by 1. $R_1[1, 1, 4] = [0, 0, 1]$, which means a filter is assigned to node 1 and no filter is assigned to the left or right subtrees. The blocked attack traffic load $R_2[1, 1, 4]$ is $[0, 0]$ as no traffic is blocked from the left and right subtrees. Similarly, we can calculate the $A_1[1, j, 4]$, $R_1[1, j, 4]$, and $R_2[1, j, 4]$ where the j ranges between 1 and 3 are 0, $[0, 0, j]$ and $[0, 0]$.

Calculations of $A_1[i]$, $R_1[i]$, and $R_2[i]$ for any leaf node i are trivial. Figure 5 shows the values of $A_1[i]$, $R_1[i]$, and $R_2[i]$ for the leaf nodes.

Calculation for Node 5 (0 Filters to Block Attack Traffic): For $i = 5$, different filter assignments can block 0, 4, 15, or 19 attack traffic. For $j = 0$, we have one option.

Option II: 0 filters for nodes 5, 1, and 2 ($j_1 = 0$, $j_2 = 0$). The total contamination in this option is $4 + 15 = 19$. The total blocked attack load is $0+0 = 0$. Total unblocked attack traffic from node 1 is 4, which produces contamination of 4. Optimal contamination at the subtree rooted by 1 is 0 for $j = 0$ and $b = 0$. Total unblocked attack traffic from node 2 is 15, which produces contamination of 15. Optimal contamination at the subtree rooted by 2 is also 0 for $j = 0$ and $b = 0$. Therefore, the minimum contamination for $j = 0$ and $b = 0$ is $A_1[5, 0, 0] = 19$. The assignment $R_1[5, 0, 0] = [0, 0, 0]$ and blocked traffic at the optimal contamination is $R_2[5, 0, 0] = [0, 0]$.

We cannot block 4, 15, or 19 attack traffic with 0 filter. Therefore, $A_1[5, 0, 4] = A_1[5, 0, 15] = A_1[5, 0, 19] = \infty$. $R_1[5, 0, b]$ and $R_2[5, 0, b]$ is $[-]$ for any b in $\{4, 15, 19\}$.

Calculation for Node 5 (1 Filter to Block 0 Attack Traffic): For $j = 1$, if we want to block 0 ($b = 0$) attack traffic, then we have two options:

Option I: We have one choice in this option.

- Choice (1): Blocking 0 attack traffic using 0 filter at node 1, blocking 0 attack traffic using 0 filter at 2, and assigning a filter to node 5. $(j_1 = 0, j_2 = 0, b_1 = 0, b_2 = 0)$. This choice is invalid because if we assign a filter to node 5, then 19 attack traffic will be blocked.

Option II: We have two choices in this option.

- Choice (1): Blocking 0 attack traffic using 0 filter at node 1 and blocking 0 attack traffic using 1 filter at 2 $(j_1 = 0, j_2 = 1, b_1 = 0, b_2 = 0)$. Contamination for this choice is $A_1[1,0,0]+dist[5,1](AL[1]-b_1)+A_1[2,1,0]+dist[5,2](AL[2]-b_2) = 0+4+\infty+15 = \infty$.
- Choice (2): Blocking 0 attack traffic using 1 filter at node 1 and blocking 0 attack traffic using 0 filter at 2 $(j_1 = 1, j_2 = 0, b_1 = 0, b_2 = 0)$. Contamination for this choice is $A_1[1,1,0]+dist[5,1](AL[1]-b_1)+A_1[2,0,0]+dist[5,1](AL[2]-b_2) = \infty+4+0+15 = \infty$.

The minimum contamination in option II is ∞. Therefore, the minimum contamination for $j = 1$ and $b = 0$ is ∞. So, $R_1[5,1,0]$ and $R_2[5,1,0]$ are invalid ([−]).

Calculation for Node 5 (1 Filter to Block 4 Attack Traffic): For $j = 1$, if we want to block 4 $(b = 4)$ attack traffic, then we have two options:

Option I: We have few choices in this option. Any choice will assign a filter to node 5, which will block 19 attack traffic. For $b = 4$, this option is invalid.

Option II: We have four choices in this option.

- Choice (1): Blocking 0 attack traffic using 0 filter at node 1 and blocking 4 attack traffic using 1 filter at 2 $(j_1 = 0, b_1 = 0, j_2 = 1, b_2 = 4)$. Contamination for this choice is $A_1[1,0,0]+dist[5,1](AL[1]-0)+A_1[2,1,0]+dist[5,2](AL[2]-4) = \infty+4+\infty+11 = \infty$.
- Choice (2): Blocking 0 attack traffic using 1 filter at node 1 and blocking 0 attack traffic using 0 filter at 2 $(j_1 = 1, b_1 = 0, j_2 = 0, b_2 = 4)$. Contamination for this choice is $A_1[1,1,0]+dist[5,1](AL[1]-0)+A_1[2,0,0]+dist[5,1](AL[2]-4) = \infty+4+\infty+11 = \infty$.
- Choice (3): Blocking 4 attack traffic using 0 filter at node 1 and blocking 0 attack traffic using 1 filter at 2 $(j_1 = 0, b_1 = 4, j_2 = 1, b_2 = 0)$. Contamination for this choice is $A_1[1,0,0]+dist[5,1](AL[1]-4)+A_1[2,1,0]+dist[5,2](AL[2]-0) = \infty+0+\infty+15 = \infty$.
- Choice (4): Blocking 4 attack traffic using 1 filter at node 1 and blocking 0 attack traffic using 0 filter at 2 $(j_1 = 1, , b_1 = 4, j_2 = 0, b_2 = 0)$. Contamination for this choice is $A_1[1,1,0]+dist[5,1](AL[1]-b_1)+A_1[2,0,0]+dist[5,1](AL[2]-b_2) = 0+0+0+15 = 15$.

The minimum contamination in option II is 15. Therefore, the minimum contamination for $j = 1$ and $b = 4$ is $A[5,1,4]$ is 15. So, $R_1[5,1,4]$ and $R_2[5,1,4]$ are $[1,0,0]$ and $[4,0]$, respectively.

Calculation for Node 5 (1 Filter to Block 15 Attack Traffic): For $j = 1$, if we want to block 15 $(b = 15)$ attack traffic, then we also have two options:

Option I: Although we have few choices in this option, but every choice will assign a filter to node 5 which will block 19 attack traffic. For $b = 15$, this option is invalid.

Option II: We have four choices in this option.

- Choice (1): Blocking 0 attack traffic using 0 filter at node 1 and blocking 15 attack traffic using 1 filter at 2 $(j_1 = 0, b_1 = 0, j_2 = 1, b_2 = 15)$. Contamination for this choice is $A_1[1,0,0] + dist[5,1](AL[1]-0) + A_1[2,1,15]+dist[5,2](AL[2]-15) = 0+4+0+0 = 4$.
- Choice (2): Blocking 0 attack traffic using 1 filter at node 1 and blocking 15 attack traffic using 0 filter at 2 $(j_1 = 1, b_1 = 0, j_2 = 0, b_2 = 15)$. Contamination for this choice is $A_1[1,1,0]+dist[5,1](AL[1]-0)+A_1[2,0,15]+dist[5,1](AL[2]-15) = \infty+4+\infty+0 = \infty$.
- Choice (3): Blocking 15 attack traffic using 0 filter at node 1 and blocking 0 attack traffic using 1 filter at 2 $(j_1 = 0, b_1 = 15, j_2 = 1, b_2 = 0)$. Contamination for this choice is $A_1[1,0,15] + dist[5,1](AL[1]-15) + A_1[2,1,0]+dist[5,2](AL[2]-0) = \infty+0+\infty+15 = \infty$.
- Choice (4): Blocking 15 attack traffic using 1 filter at node 1 and blocking 0 attack traffic using 0 filter at 2 $(j_1 = 1, , b_1 = 15, j_2 = 0, b_2 = 0)$. Contamination for this choice is $A_1[1,1,15] + dist[5,1](AL[1]-15) + A_1[2,0,0]+dist[5,1](AL[2]-0) = \infty+0+0+15 = \infty$.

The minimum contamination in option II is 4. Therefore, the minimum contamination ($A_1[5,1,15]$) for $j = 1$ and $b = 15$ is 15. So, $R_1[5,1,15]$ and $R_2[5,1,15]$ are $[0,1,0]$ and $[0,15]$, respectively.

Similarly, we calculate the rest of the values in A_1 and R_1. Figs. 6(a) and 6(c) show the complete value of A_1, R_1, and R_2. $A_1[7,3]$ contains the optimal contaminations for 3 filters for different amount of blocked attack traffic in the topology in Fig. 4(a).

4.4 Assignment Set Formulation

We generate the filter assignment set using R_1 and R_2. The total amount of attack traffic is 22. Therefore, $R_1[7,3,22]$ contains the assignment at node 7 for budget 3. $R_1[7,3,22] = [2,1,0]$ means the left subtree is assigned 2 filters and the right subtree is assigned 1 filter. Then, we look up $R_2[7,3,22] = [19,3]$ which means 2 and 1 filters are used to block 19 and 3 attack traffic at node 5 and 4, respectively. Next, we need to look up $R_1[5,2,19]$ and $R_1[4,1,3]$. $R_2[5,2,19]$ is $[1,1,0]$, which means left and right subtrees both are assigned 1 filter. We look up $R_2[5,2,19] = [4,15]$ which means 1 and 1 filters are used to block 4 and 15 attack traffic at node 1 and 2, respectively. So, we need to look up $R_1[1,1,4]$ and $R_1[2,1,15]$. $R_1[1,1,4]$ and $R_1[2,1,15]$ are both $[0,0,1]$ which means a filter is assigned to both nodes 1 and 2. Now our assignment set is $\{1,2\}$. Next, we look up $R_1[4,1,3]$. $R_1[4,1,3]$ is $[0,0,1]$, which means a filter is assigned to node 4. Therefore, the final assignment set is $\{1,2,4\}$.

Next, we see another example where we find assignment of budget 2. According to the definition, $R_1[7,2,22]$ contains the assignment at node 7 for a budget of 2 filters. $R_1[7,2,22] = [1,0,1]$ means the left subtree is assigned 1 filter and the right subtree is assigned 0 filters. Node 7 itself is assigned 1 filter. Now, our assignment set is $\{7\}$. Next we need to look up $R_1[5,1,15]$ ($R_2[7,2,22] = [15,0]$). $R_1[5,1,15]$ is $[0,1,0]$, which means right subtree is assigned 1 filter. So, we need to look up $R_1[2,1,15]$ ($R_2[5,1,15] = [0,15]$). $R_1[2,1,15]$ is $[0,0,1]$, which means a filter is assigned to both nodes 2. Therefore, the final assignment set for budget of 2 filters is $\{7,2\}$. The algorithm is shown in Alg. 3.

Algorithm 1 DP Blocking Strategy for Problem 1

Input: The number of filters K, total attack traffic B, and topology tree T.
Output: A set of nodes in T.

Procedure: BLOCK-DP1(K, B, T)
  $N \leftarrow$ number of nodes in T
  for every entry node i **do**
    Initialize $AL[i]$.
    for $j = 0$ to K **do**
      for $b = 0$ to B **do**
        Initialize $A_1[i, j, b]$, $R_1[i, j]$, and $R_2[i, j]$
  CALC-P1(N, K, B)
  return ASSIGNMENT(R_1, R_2, N, K, B)

Algorithm 2 Calculate A_1 and R_1

Procedure: CALC-P1(N, K, B)
  for $i = 1$ to N **do**
    for $j = 0$ to K **do**
      for $b = 0$ to B **do**
        $min \leftarrow= \infty$, $map \leftarrow \emptyset$
        for $\forall j_\delta, b_\delta : \sum_{\delta=1}^{\Delta} j_\delta = j \sum_{\delta=1}^{\Delta} b_\delta = b$ **do**
          $p \leftarrow \sum_{\delta=1}^{\Delta} \{A_1[c_\delta(i), j_\delta, b_\delta] + dist(c_\delta(i), i)(AL[c_\delta(i)] - b_\delta)\}$
          $key \leftarrow [j_1, j_2, ..., j_\Delta, 0, b_1, b_2, ..., b_\Delta]$
          PUT(map, key, p)
        for $\forall j_\delta, b_\delta : \sum_{\delta=1}^{\Delta} j_\delta = j - 1$ **do**
          $p \leftarrow \sum_{\delta=1}^{\Delta} \{A_1[c_\delta(i), j_\delta, b_\delta] + dist(c_\delta(i), i)(AL[c_\delta(i)] - b_\delta)\}$
          $key \leftarrow [j_1, j_2, ..., j_\Delta, 1, b_1, b_2, ..., b_\Delta]$
          PUT(map, key, p)
        $A_1[i, j, b] \leftarrow$ MIN(map)
        $R_1[i, j, b]$, $R_2[i, j, b] \leftarrow$ ARGMIN(map)

Theorem 1. *Complexity and space needed of Alg. 1 are $O(N(KB)^{(\Delta)})$ and $O(NKB\Delta)$.*

Proof. Let us consider that the topology is an N node tree with maximum node degree Δ and the victim has budget of K. To find the partitions $j_1, j_2, ..., j_\Delta$ we need $O(K^{(\Delta-1)})$ time if we use the naive nested iteration approach. Similarly, to find the partitions $b_1, b_2, ..., b_\Delta$ we need $O(B^{(\Delta-1)})$ time. Therefore, the complexity of Alg. 1 is $O(N(KB)^{(\Delta)})$.

For A_1, R_1, and R_2 we need NBK, $NBK(\Delta + 1)$, and $NKB(\Delta)$ space. For AL and $dist$ we need $2N$ space. Therefore, in total we need $O(NKB\Delta)$ space. For a binary tree topology the complexity is $O(N(KB)^2)$ and the space complexity is $O(NKB)$. □

Theorem 2. *Alg. 1 provides optimal solution.*

Proof. Alg. 1 uses a dynamic programming bottom-up strategy to search the optimal assignment. For a one-node tree, if the node color is not "white", then there is no solution for $K = 0$. This is because without any filter, the attack traffic will be forwarded to the downstream routers and finally reach v. For $K \geq 1$ there is only one choice for selecting FR, which is that node. If that node is selected, the optimal contamination is 0. In each step, the Alg. 4 chooses the allocation of filters to itself, the left subtree, or the right subtree which produces the minimum contamination. Therefore, the Alg. 1 provides an optimal filter assignment to the FRs through an exhaustive search. □

Algorithm 3 Find Assignment

Procedure: ASSIGNMENT(R_1, R_2, N, K, B)
  if $B = 0$ **then**
    return $\emptyset$
  $x.i \leftarrow N, x.j \leftarrow K, x.b \leftarrow B$, $g \leftarrow \emptyset$, and $Q \leftarrow \emptyset$.
  ENQUEUE(Q,x).
  while $Q \neq \emptyset$ **do**
    $x \leftarrow$ DEQUEUE(Q)
    if $R_1[x.i, x.j, x.b, \Delta + 1] \neq 0$ **then**
      $K' \leftarrow R_1[x.i, x.j, x.b, i]$
      $B' \leftarrow R_2[x.i, x.j, x.b, i]$
      $g' \leftarrow \bigcup_{\delta=1}^{\Delta}$ ASSIGNMENT($R_1, R_2, c_\delta(x.i), K', B'$)
      $g \leftarrow g \cup g' \cup \{x.i\}$
    else
      for $\delta = 1$ to Δ **do**
        $x'.i \leftarrow c_\delta(x.i)$
        $x'.j \leftarrow R_1[x.i, x.j, x.b, i]$
        $x'.b \leftarrow R_2[x.i, x.j, x.b, i]$
        ENQUEUE(Q, x')
  return g

5 Destination-based Filter Assignment

As we are using destination-based filters for protection against spoofed DDoS attack, we are blocking some LUs. In this section, we formulate another problem of assigning destination-based filters to the FRs so that a weighted sum of the contamination and blocked LUs is the minimum.

5.1 Problem Definition

Problem 2. *Find a filter assignment so that the LU blockage and contamination are the minimum.*

It is always better if the victim can select some FRs within its budget which minimize both the number of blocked LUs and contamination. To formulate the problem we first define the cost ($C(g)$) of filter assignment (g) as the following:

$$C(g) = \omega W_c(g) + (1 - \omega)|U_b(g)|. \tag{6}$$

Here, $\omega = [0, 1]$ is considered a system parameter which determines the priority of contamination and LU blockage. If the LUs of the victim is more important to it than reducing contamination in the network, then it sets a low value of ω. For example, if the victim wants to block the minimum number of LUs neglecting the contamination, then it sets $\omega = 0$. If $\omega = 0$, then the contamination in the network does not have any effect on the cost and filter assignment.

As discussed in Section 3, the source-based filter cannot ensure protection against IP spoofing DDoS attacks. For example, if the attacker attached to node 1 uses the IP address of the users attached to node 3 (see Fig. 4(a)). The filter used at node 1 or 5 would forward the packet. But if the FRs use destination-based filters, then no spoofed attack packet can penetrate. Therefore, the victim would use destination-based filters. The problem can be expressed as the following optimization problem:

$$\begin{aligned} &\text{minimize} && C(g) \\ &\text{subject to} && |g| \leq K, \forall g \subset G, v \notin G. \end{aligned} \tag{7}$$

5.2 An Optimal Solution

To solve the problem, we define the following problem.

1) $P_3(i, j)$: Find and return the optimal cost in the subtree rooted by i by ensuring blockage of all attack traffics before reaching v. The number of LUs, optimal cost,

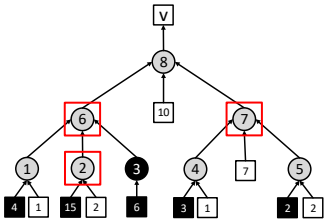


Fig. 7: Topology for Problem 2.

A

i \ j	0	1	2	3
1	∞	0.5	0.5	0.5
2	∞	1	1	2
3	∞	0	0	0
4	∞	1	1	1
5	∞	2	2	2
6	∞	14	6.5	1.5
7	∞	7.5	1.5	1.5
8	∞	36.5	21.5	14

R

i \ j	0	1	2	3
1	0,0,0,0	0,0,0,1	0,0,0,2	0,0,0,3
2	0,0,0,0	0,0,0,1	0,0,0,2	0,0,0,3
3	0,0,0,0	0,0,0,1	0,0,0,2	0,0,0,3
4	0,0,0,0	0,0,0,1	0,0,0,2	0,0,0,3
5	0,0,0,0	0,0,0,1	0,0,0,2	0,0,0,3
6	0,0,0,0	0,0,0,1	0,0,0,2	1,1,1,0
7	0,0,0,0	0,0,0,1	1,1,0,0	2,1,0,0
8	0,0,0,0	0,0,0,1	1,1,0,0	2,1,0,0

L

i	
1	1
2	2
3	0
4	1
5	2
6	3
7	10
8	23

Fig. 8: A, R, and L.

and filter assignments are stored in L, A, and R to reuse in dynamic programming.

The optimal cost of using j destination-based blocking/filter in the subtree rooted by i is the minimum of the following quantity:

Option I: The minimum total weighted cost, if we assign 1 filter to node i and the rest of the filters to some nodes of the subtree rooted by i. Therefore, the cost will be a weighted sum of the minimum contamination and LU in the subtree rooted by i. When we assign a filter to i, then the number of blocked LUs remains constant regardless of other filter assignments. The problem becomes similar to Problem 1. We can apply Alg. 1 to find an optimal assignment in the subtree rooted by i. First, we assume an attacker is attached to i. This assumption confines a filter to i. Then we find an assignment of budget j using Alg. 1. As i will be assigned a filter, the other $j-1$ filters will be assigned to the subtree rooted by i. Then the cost for this option will be:

$$P_3(i,j) = \omega P_2(i,j) + (1-\omega)L[i]. \tag{8}$$

Option II: The minimum total weighted cost, if we divide the number of filters into $j_1, j_2, ..., j_\Delta$ parts and assign them to the subtrees $c_1(i), c_2(i), ..., c_\Delta(i)$, respectively. Therefore, the cost for this option will be:

$$P_3(i,j) = \sum_{\delta=1}^{\Delta} P_3(c_\delta(i), j_\delta). \tag{9}$$

We take the minimum quantity between the above two options. If there are some attackers attached to node i, we do not consider option II. This is because, if we assign all the j filters to its subtree, then the attack traffic from i will reach the victim v, which is not allowed by the constraint of the problem definition.

Let us consider an N node tree with maximum node degree Δ. The nodes are labeled in a bottom-up and left-right order. We define A as an $N \times K$ array which contains the optimal cost for every node and budget. For example, $A[i,j]$ is the optimal cost of budget j on the subtree rooted by node i. We define L as a $1 \times N$ array which contains

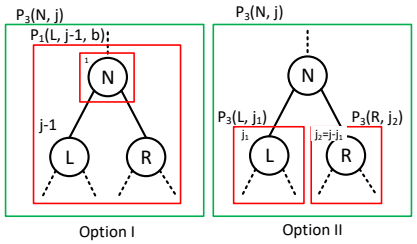


Fig. 9: Recursion model for Problem 2.

the number of LUs in the subtree rooted by every node. $L[i]$ is the number of LUs in the subtree rooted by node i. We also define R as an $N \times K \times (\Delta+1)$ array which contains the number of filters assigned to node i and its subtrees for every node and budget. For example, $R[i,j,1]$, $R[i,j,2]$, and $R[i,j,\Delta+1]$ are the number of filters to the first subtree, the second subtree, and node i of subtree rooted by i for budget j. The complete algorithm is shown in Alg. 4.

5.3 An Example

Let us consider the traffic topology in Fig. 7 and $\omega = 0.5$. We compute the values of $A[i,j]$, $R[i,j]$ and $L[i]$ for $i = 1, ..., 7$ and $j = 0, 1, ..., 3$. We can calculate the values of $L[i]$ by traversing the tree once in a bottom up order.

Calculation for Leaf Nodes (Nodes 1, 2, 3, 4 and 5): The leaf entry nodes are $1, 2, 3, 4,$ and 5. The calculations of A and R are straightforward. For example $A[1,0] = \infty$. This is because without any filter we cannot block all attack traffic. If we assign 1 filter to node 1, then we are blocking one LU, then $A[1,1] = 0.5 \times 0 + 0.5 \times 1 = 0.5$. Similarly, $A[1,2] = 0.5 \times 0 + 0.5 \times 1 = 0.5$. $R[i,1] = [0,0,0,1]$, $R[i,2] = [0,0,0,2]$, and $R[i,3] = [0,0,0,3]$ for $i \in \{1,2,3,4,5\}$.

Calculation for Node 6 Using 0 Filters: For node 6 and $j = 0$, we have one option which is option II.

Option II: We assign 0 filters to node 6. Then, we assign 0 filters to subtrees rooted by node 1, 2, and 3 ($j_1 = 0, j_2 = 0, j_3 = 0$). Therefore, $A[6,0] = A[1,0] + A[2,0] + A[3,0] = \infty$ and $R[6,0] = [0,0,0,0]$.

Calculation for Node 6 Using 1 Filter: For $j = 1$, we have two options for assigning the filter.

Option I: 1 filter for node 6 and no filters for its subtrees ($j_1 = 0, j_2 = 0, j_3 = 0$). If we assign 1 filter to node 6, the number of block LUs is 3 ($L[6] = 3$). The contamination in this option is 25 ($A_1[6,0,0] = 25$) which is calculated using Alg.1. Therefore, for this option, $A[6,1] = 0.5 \times A_1[6,0,0] + 0.5 \times L[6] = 0.5 \times 25 + 0.5 \times 3 = 14$.

Option II: We assign 0 filters to node 6 and rest of the filters to its subtrees. We have three choices to assign filters to its subtrees.

- Choice (1): Assign 1, 0, 0 filters to subtrees rooted by node 1, 2, and 3, respectively ($j_1 = 1, j_2 = 0, j_3 = 0$). Therefore, $A[6,1] = A[1,1] + A[2,0] + A[3,0] = \infty$.
- Choice (2): Assign 0, 1, 0 filters to subtrees rooted by node 1, 2, and 3, respectively ($j_1 = 0, j_2 = 1, j_3 = 0$). Therefore, $A[6,1] = A[1,0] + A[2,1] + A[3,0] = \infty$.
- Choice (3): Assign 0, 0, 1 filters to subtrees rooted by node 1, 2, and 3, respectively ($j_1 = 0, j_2 = 0, j_3 = 1$). Therefore, $A[6,1] = A[1,0] + A[2,0] + A[3,1] = \infty$.

For option II, the minimum cost is ∞. Therefore, option I is the minimum ($A[6,1] = 14$) and $R[6,1] = [0,0,0,1]$.

Algorithm 4 DP Blocking Strategy for Problem 2

Input: The number of filters K, total attack traffic B, and topology tree T.
Output: A set of nodes in T.

Procedure: BLOCK-DP2(K, B, T)
  $N \leftarrow$ number of nodes in T
  for every entry node i **do**
    Initialize $AL[i]$ and $L[i]$.
    for $j = 0$ to K **do**
      Initialize $A_1[i,j]$, $A[i,j]$, $R_1[i,j]$, $R_2[i,j]$, and $R[i,j]$
  CALC-P1(N,K,B)
  CALC-P3(N,K,B)
  return ASSIGNMENT-2(R, R_1, R_2, N, K, B)

Algorithm 5 Compute A and R

Procedure: CALC-P3(N, K, B)
  for $i = 1$ to N **do**
    for $j = 0$ to K **do**
      $L[i] \leftarrow \sum_{\delta=1}^{\Delta} L[c_\delta(i)]$
      $min \leftarrow= \infty$
      if i is attached with attacker **then**
        $p \leftarrow \omega A_1[i,j,AL[i]] + (1-\omega)L[i]$
        PUT($map, R_1[i,j], p$)
      else
        for $\forall j_\delta : \sum_{\delta=0}^{\Delta} j_\delta = j$ **do**
          $p \leftarrow \sum_{\delta=1}^{\Delta} A[c_\delta(i), j_\delta]$
          PUT($map, [j_1, j_2, ..., j_\Delta, 0], p$)
      $A[i,j] \leftarrow$ MIN(map)
      $R[i,j] \leftarrow$ ARGMIN(map)

Similarly, we calculate the rest of the entries in A and R. The complete A, L, and R are shown in Fig. 8. According to the definition, $A[8,3]$ contains the cost for budget $K = 3$, which is 14.

5.4 Assignment Set Formulation

From R, we can find out which FRs need to be blocked. $R[8,3,0] = 2$ and $R[8,3,1] = 1$ means 2 and 1 filters are assigned to the first and second subtrees of node 8, respectively. Then, we need to look up $R[6,2]$ and $R[7,1]$. $R[6,2,0] = 0$, $R[6,2,1] = 0$, $R[6,2,2] = 0$, and $R[6,2,3] = 2$, means no filter is assigned to its subtrees and two filters are assigned to itself. Therefore, we need to find the assignment using Alg. 3. According to Alg. 3, $\{6,2\}$ is the assignment. Similarly, we can find that a filter is assigned to node 7. So, the filter assignment is $\{2,6,7\}$ for budget $K = 3$. The complete algorithm is shown in Alg. 6.

Theorem 3. *Complexity and space needed of the DP Blocking Strategy for Problem 2 are $O(N(KB)^{(\Delta-1)})$ and $O(NKB\Delta)$.*

Proof. Let us consider the topology is a N node tree with a maximum node degree Δ and the victim has a budget of K. To find the partitions $j_1, j_2, ..., j_\Delta$, we need $O(K^{(\Delta-1)})$ time if we use the naive nested iteration approach. The algorithm uses Alg. 1. Therefore, the complexity of Alg. 4 is $O(N(KB)^{(\Delta-1)})$.

The additional space needed for A and R are NK and $(\Delta+1)NK$, respectively. The total space needed including space needed for Alg. 1 is an order of $O(NKB\Delta)$.

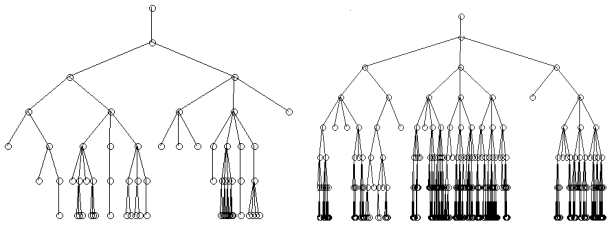
□

Theorem 4. *Alg. 4 provides an optimal solution.*

Proof. Alg. 4 uses a dynamic programming bottom-up strategy to search the optimal assignment. For a one-node tree, if the node color is "black" or "gray", then there is no solution for $K = 0$. This is because without any filter, the attack traffic will be forwarded to the downstream routers. For $K \geq 1$ there is only one choice for selecting FRs, which is that node. If that node is selected, the optimal number of blocked LUs is the number of LUs attached to it. In each step, Alg. 4 chooses the best allocation of filters to itself, the left subtree, or the right subtree. Therefore, Alg. 4 provides an optimal filter assignment to the FRs through an exhaustive search. □

Algorithm 6 Find Assignment 2

Procedure: ASSIGNMENT-2(R, R_1, R_2, N, K, B)
  $x.i \leftarrow N, x.j \leftarrow K$, $g \leftarrow \emptyset$, and $Q \leftarrow \emptyset$.
  ENQUEUE(Q,X).
  while $Q \neq \emptyset$ **do**
    $x \leftarrow$ DEQUEUE(Q)
    if $R[x.i, x.j, \Delta+1] \neq 0$ **then**
      $g \leftarrow g \cup$ ASSIGNMENT(R_1, R_2, N, K, B)
    else
      for $k = 1$ to Δ **do**
        $x'.i \leftarrow c_k(x.i)$, $x'.j \leftarrow R[x.i, x.j, k]$
        ENQUEUE(Q, x_c)
  return g



(a) Topology I. (b) Topology II.

Fig. 10: Randomly generated topologies.

6 Experimental Results

In this section, we present our experimental settings and simulation results.

6.1 Experimental Setting

We conduct the experiments with a custom build Java simulator. The main reason for using a custom built simulator is its scalability. We do not need to analyze transmission time, bandwidth, or packet drop issues. We only need to count the number of legitimate (or attack) received (or blocked) packets. The network topologies we considered contain about $100 - 500$ routers. Using NS3 or other similar simulators for this kind of simulation would take several days. That is why we built our own Java multi-threaded simulator to get the results quickly. The simulation result might be slightly different than the real scenario because of the natural packet drops, failure of the FRs, the variable data rate of LUs/attackers, and change in the routing paths. Therefore, in the real scenario, the victim needs to periodically change the filter assignment. The contamination, blocked/received attack packet, and blocked/received legitimate packets might not be significantly different than the victim's calculation.

We conduct simulations for randomly generated tree topologies and a subset from a real network topology. To generate a random tree, we first generate the desired

TABLE 2: Topology Parameters

	Topology I	Topology II
Number of nodes	66	403
Internal user probability	0.1	0.1
Attacker ratio	0.4	0.4
Max Node Degree	4	20
Data Rate(pack/ms)	[0.1-0.4]	[0.1-0.4]

number of nodes. Then, we randomly pick a root among the nodes. After that a random node from the generated nodes is picked up and added as a child to a random node in the tree. The process continues until all of the generated nodes are added to the tree. We use a randomly generated topology having node degree between $[0-4]$, internal node user probability between $[0.1-0.25]$, and maximum depth of 6. Each entry node color and the number of users or attackers are selected randomly from a uniform distribution. Topology I is a randomly generated tree of 66 nodes and max node degree of 4. Topology II is taken from a subset of the Stanford University AS-733 dataset [27]. The dataset contains $6,474$ nodes and we took a subset (which is a tree) containing 403 nodes. Then we randomly assigned users to the tree with an internal user probability of 0.1. The details are shown in Table 2 and Fig. 10.

We measure the performances of our proposed solutions in terms of contamination (C), Cost (according to Equation 6), blocked LU traffic, the number of blocked attack packets (AB), the number of received attack packets (AR), number of blocked legitimate packets (LB), and the number of received legitimate packets (LR) for the two topologies.

6.2 Simulation Results

We first conduct a simulation with a three level complete binary tree to see the number of packets needed to find the topology. Each node is considered as a FR. Each leaf node is attached with a LU. The root is connected with the victim. Each LU sends packets to the victim with a constant rate. Then we assign the same marking probability to all FRs. Each time when the victim receives a marked packet, it tries to construct the topology form the marking information it got so far. If the constructed topology is correct up-to one level (from the root) then we record the number of total packets (both marked and unmarked) received by the victim. The process continues and we record the total number of packets received by the victim when it succeeds in constructing the topology up to two levels and three levels. We repeat the process by assigning different marking probability. Fig. 11 shows the number of packets by the marking probability. We run the simulation 50 times and take the average. The marking probability ranges between 0.2 and 1. A huge number of packets is needed to construct the complete topology when marking probability is below 0.2. We can observe that the number of packets needed reduces exponentially with the marking probability.

For the following experiments, we use Topology II to observe the performances of both approaches for different budgets. We change the attacker ratio and repeat the experiments. We plot the average and standard deviation of 100 random attacker and LU distributions.

Fig. 12(a) shows the contamination by the number of source-based filters. We vary the number of filters from

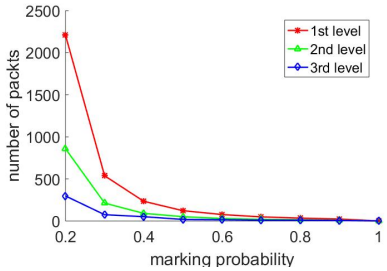
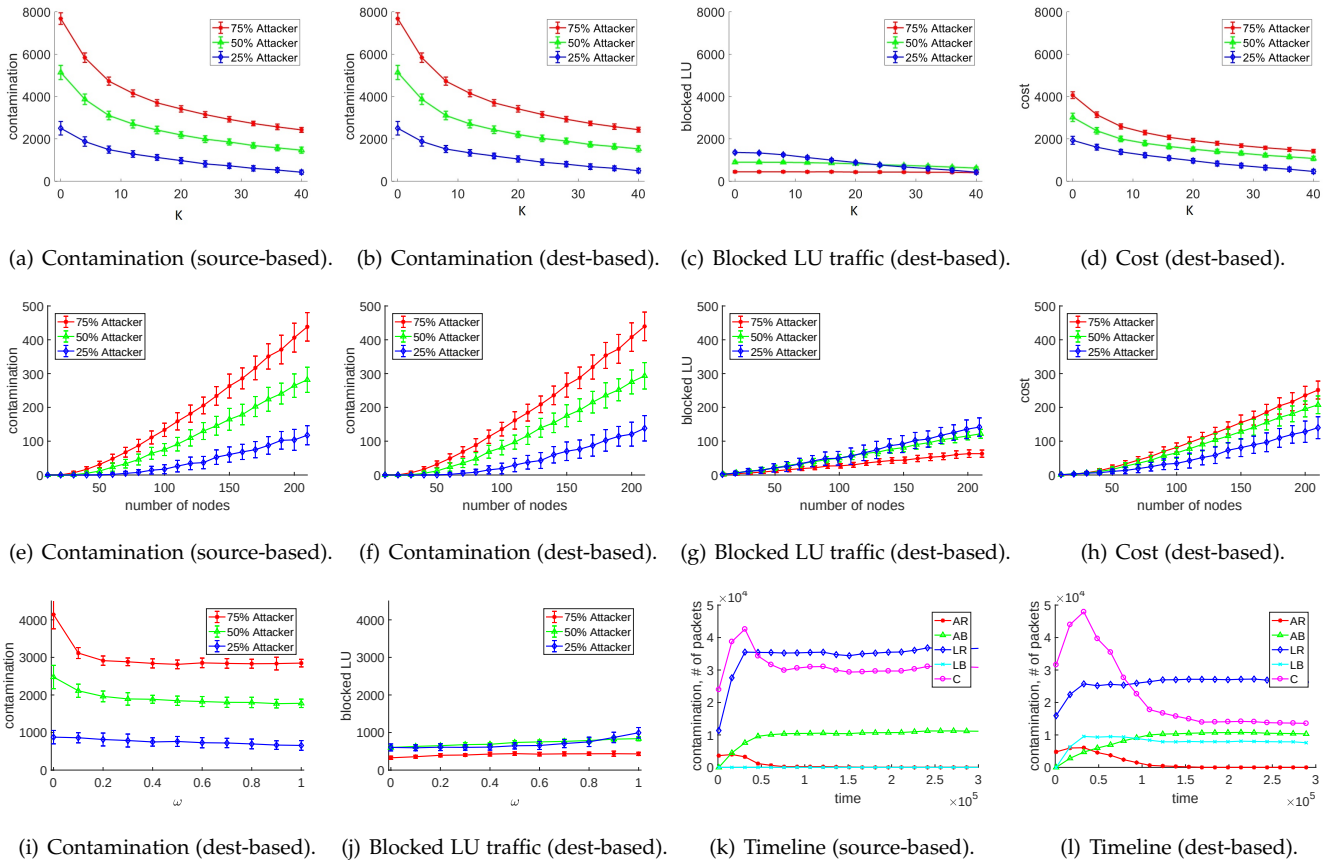


Fig. 11: Formation of topology.

1 to 40. The highest contamination is with 75% attackers, the lowest is 25%, and 50% is in between for all budgets. The contaminations of all attacker distributions decrease by the number of filters. The higher the number of filters, the closer the filters are deployed to the attackers. As a result, a higher number of filters produces a lower contamination. For 75% attackers, the contaminations with 1 and 40 filters are $7,681$ and $2,424$. Therefore, the contamination is 68% reduced. For 25% attackers, the contaminations with 1 and 40 filters are $2,503$ and 426. Therefore, the contamination is 82% reduced.

Figs. 12(b), 12(c), and 12(d) show the contamination, blocked LU traffic, and cost for destination-based filters. We keep the ω as 0.5. We observe that the contamination of source-based filter and destination-based filters are similar. This is because, we give equal priority to the contamination and blocked LU traffic. The amount of blocked LU traffic is also decreasing by the number of filters. As a result, the cost is decreased by the number of filters. For 75% attackers the blocked LU traffic with 1 and 40 filters are 446 and 402. Therefore, the blocked LU traffic is reduced by 10%. For 25% attackers, the blocked LU traffic with 1 and 40 filters are $1,360$ and 427. Therefore, the blocked LU traffic is reduced by 69%.

Fig. 12(e) shows the contamination by the number of nodes. We vary the number of nodes from 10 to 210. We keep the number of filters as 20. Similar to the above experiment, the highest contamination is with 75% attackers, the lowest is 25%, and 50% is in between for all numbers of nodes. The contaminations of all attacker distributions increase by the number of nodes. The higher the number of nodes, the higher the number of attackers and the height of the tree. As a result, a higher number of nodes produces higher contamination. For 75% attackers, the contamination with 10 and 210 nodded trees are 0 and 438. For 25% attackers, the contamination with 10 and 210 nodded trees are 0 and 117. In 10 nodded trees, we observe a contamination of 0, because 20 filters are more than enough to block every attacker at the closest router. Therefore, no attack traffic enters into the network and contamination is 0. Figs. 12(f), 12(g), and 12(h) show the contamination, blocked LU traffic, and cost for destination-based filters. We keep the ω as 0.5. We observe that the contamination of source-based and destination-based filters are also similar. The amount of blocked LU traffic is also increasing by the number of nodes. As a result, the cost is increasing by the number of nodes. For 75% attackers the blocked LU traffic in 10 and 210 nodded trees are 1 and 63. For 25% attackers the blocked LU traffic in 10 and 210 nodded trees are 2 and 147. The contamination, blocked LU traffic, and cost increase almost



(a) Contamination (source-based). (b) Contamination (dest-based). (c) Blocked LU traffic (dest-based). (d) Cost (dest-based).

(e) Contamination (source-based). (f) Contamination (dest-based). (g) Blocked LU traffic (dest-based). (h) Cost (dest-based).

(i) Contamination (dest-based). (j) Blocked LU traffic (dest-based). (k) Timeline (source-based). (l) Timeline (dest-based).

Fig. 12: Simulation results.

linearly by the number of nodes.

Figs. 12(i) and 12(j) show the contamination, blocked LU traffic, and cost for destination-based filters. We vary the value of ω from 0 to 1. We keep the number of filters as 30. The contamination and amount of blocked LU traffic decrease and increase by ω, respectively. For 75% attackers, when $\omega = 0$, the contamination and blocked LU traffic are $4,141$ and 327. When $\omega = 1$, the contamination and blocked LU traffic are $2,848$ and 431. When ω is lower, the blocked LU traffics are prioritized over contamination. Therefore, when $\omega = 0$ the contamination is higher than when $\omega = 1$. Similarly, when $\omega = 0$, the blocked LU traffic is lower than when $\omega = 1$.

Figs. 12(k) and 12(l) show contamination (C), the number of blocked attack packets (AB), the number of received attack packets (AR), the number of blocked legitimate packets (LB), and the number of received legitimate packets (LR) using source-based and destination-based filters over time. The time for this simulation is the system time of the machine we used for simulation. This time does not reflect the actual time but it shows the changes of the C, AB, AR, LB, and LR over time. Topology I is used in this experiment because it is smaller than Topology II. For this reason, we can observe the effect of topology construction better in Topology I than in Topology II. Here, the contamination is the total number of attack-packet forwarding events. We can see that, at the beginning (ignoring the warm-up period from time 0 to 0.025), the C in every approach is higher. The C reduces over time and becomes gradually more stable. This is because at the beginning, the victim knows a small subset of the topology. Over time, the victim gets more and more information from the marked packet and constructs the traffic topology. Finally, the victim's knowledge about the topology becomes stable. That is why the AR is high at the beginning, decreases over time, and finally converges to 0. The AB shows the opposite behavior for the same reason. We also observe that the LB is 0 in source-based filter. The AB is initially 0 when no filter is deployed. Gradually, the AB increases and becomes stable after some time.

7 Conclusion

The DDoS attack is the most powerful attack that makes a service unavailable to users. It is not possible to protect any server from DDoS attacks without the help of the network equipment. As the most important component in a network, routers can be upgraded to filter routers easily. Besides, the filter router can work in a network with legacy routers. In the four-phase DDoS protection system, the filter routers block the attack traffic according to the victim's instruction. Although the blocking control of an Internet service provider (ISP) is at the victim's hand, who may not belong to the ISP but it will help the ISP minimize traffic congestion. Therefore, both parties are benefited. In this work, we present three filter assignment policies for two different settings. We observe the performances of the proposed policies in synthetic and real topologies. Both the source-based and destination-based filters have some advantages and limitations. In the future, we may formulate another problem for finding an optimal assignment using the filter type most fitted to a filter router.

ACKNOWLEDGMENTS

This research was supported in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS-1651947, CNS 1564128.

REFERENCES

[1] United States Code: Title 18,1030, "Fraud and related activity in connection with computers — Government Printing Office," http://www.gpo.gov, 2014.
[2] "CloudFlare," https://blog.cloudflare.com/.
[3] B. B. Gupta and O. P. Badve, "Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a Cloud computing environment," *Neural Computing and Applications*, vol. 28, no. 12, Dec 2017.
[4] J. Wang and I. C. Paschalidis, "Statistical Traffic Anomaly Detection in Time-Varying Communication Networks," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, Jun 2015.
[5] W. Wei, F. Chen, Y. Xia, and G. Jin, "A Rank Correlation Based Detection against Distributed Reflection DoS Attacks," *IEEE Communications Letters*, vol. 17, no. 1, Jan 2013.
[6] A. Kulkarni and S. Bush, "Detecting Distributed Denial-of-Service Attacks Using Kolmogorov Complexity Metrics," *J. Netw. Syst. Manage.*, vol. 14, no. 1, Mar. 2006.
[7] T. A. Ahanger, "An effective approach of detecting DDoS using Artificial Neural Networks," in *2017 International Conference on Wireless Communications, Signal Processing and Networking*, Mar 2017.
[8] D. Almomani, M. Alauthman, F. Albalas, O. Dorgham, and A. Obeidat, "An Online Intrusion Detection System to Cloud Computing Based on Neucube Algorithms," *International Journal of Cloud Applications and Computing*, vol. 8, 04 2018.
[9] B. B. Gupta, *Computer and cyber security: principles, algorithm, applications, and perspectives*. CRC Press, 2018.
[10] K. Bhushan and B. B. Gupta, "Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, Apr 2018.
[11] X. Ma and Y. Chen, "DDoS Detection Method Based on Chaos Analysis of Network Traffic Entropy," *IEEE Communications Letters*, vol. 18, no. 1, Jan 2014.
[12] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "A System for Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, Feb 2014.
[13] P. Ning and S. Jajodia, "Intrusion Detection Techniques," 2004.
[14] N. Ye, S. Emran, Q. Chen, and S. Vilbert, "Multivariate statistical analysis of audit trails for host-based intrusion detection," *IEEE Transactions on Computers*, vol. 51, no. 7, Jul 2002.
[15] M. Zareapoor, P. Shamsolmoali, and M. A. Alam, "Advance DDOS detection and mitigation technique for securing cloud," *International Journal of Computational Science and Engineering*, vol. 16, no. 3, 2018.
[16] B. K. Joshi, N. Joshi, and M. C. Joshi, "Early Detection of Distributed Denial of Service Attack in Era of Software-Defined Network," in *2018 Eleventh International Conference on Contemporary Computing*, Aug 2018.
[17] A. Procopiou, N. Komninos, and C. Douligeris, "ForChaos: Real Time Application DDoS Detection Using Forecasting and Chaos Theory in Smart Home IoT Network," *Wireless Communications and Mobile Computing*, vol. 2019, 2019.
[18] V. Matta, M. D. Mauro, and M. Longo, "DDoS Attacks With Randomized Traffic Innovation: Botnet Identification Challenges and Strategies," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, Aug 2017.
[19] H. Luo, Z. Chen, J. Li, and A. V. Vasilakos, "Preventing Distributed Denial-of-Service Flooding Attacks With Dynamic Path Identifiers," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, Aug 2017.
[20] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Y. Yau, and J. Wu, "Realtime DDoS Defense Using COTS SDN Switches via Adaptive Correlation Analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, Jul 2018.
[21] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and M. Rajarajan, "Scale Inside-out: Rapid Mitigation of Cloud DDoS Attacks," *IEEE Transactions on Dependable and Secure Computing*, 2017.
[22] C. Wang, T. T. N. Miu, X. Luo, and J. Wang, "SkyShield: A Sketch-Based Defense System Against Application Layer DDoS Attacks," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 559–573, Mar 2018.
[23] B. Rashidi, C. Fung, and E. Bertino, "A Collaborative DDoS Defence Framework Using Network Function Virtualization," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, Oct 2017.
[24] A. Bhardwaj and S. Goundar, "Comparing Single Tier and Three Tier Infrastructure Designs against DDoS Attacks," *Int. J. Cloud Appl. Comput.*, vol. 7, no. 3, Jul 2017.
[25] D. Seo, H. Lee, and A. Perrig, "PFS: Probabilistic filter scheduling against distributed denial-of-service attacks," in *2011 IEEE 36th Conference on Local Computer Networks*, Oct 2011.
[26] ——, "APFS: Adaptive Probabilistic Filter Scheduling against distributed denial-of-service attacks," *Computers Security*, vol. 39, Nov 2013.
[27] "Autonomous systems AS-733," https://snap.stanford.edu/data/as-733.html.

Rajorshi Biswas is a PhD student of Computer and Information Sciences at Temple University, Philadelphia. He achieved his bachelor's degree from Bangladesh University of Engineering and Technology, Bangladesh. He is currently doing research at the Center for Networked Computing (CNC) which is focused on network technology and its applications. His research areas include Wireless Networks, Wireless Sensor Networks, Wireless Security, Cryptography, Cognitive Radio Networks etc. He published his research in many conferences and journals including IEEE ICPADS 2018, IEEE GLOBECOM 2019, IEEE ICC 2019, IEEE Sarnoff 2019, and Resilience Week 2019.

Jie Wu is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Mobile Computing, IEEE Transactions on Service Computing, and Journal of Computer Science and Technology. Dr. Wu was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program cochair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a Fellow of the AAAS and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.