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

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

***Abstract*—In a denial-of-service (DoS) attack, the attacker sends out a huge number of requests to exhaust the capacity of a server. The victim cannot serve incoming requests and then DoS occurs. When the attacker stops sending requests, the victim gets back to working state. The most devastating distributed DoS attack is performed by bots, malicious programs that reside on the affected user computers. By using a special type of router called filter router (FR), the victim can protect itself. A server needs to send filters to FR for blocking attack traffic. A filter blocks both attack and user traffic which are destined for the victim at the FR. The victim needs to select a subset of FRs wisely to minimize the blockage of users. The victim's operation is not hampered if the total incoming traffic does not exceed its capacity. In this paper, we formulate a problem for selecting FRs given a budget on the number of filters. The problem considers that the victim has limited incoming bandwidth and we provide an optimal dynamic programming solution. We conduct extensive simulation in different settings. Our simulation results strengthen support for our solutions.**

***Index Terms*—distributed denial-of-service, destination-based filter, filter assignment policy, network security**

I. Introduction

A denial-of-service attack (DoS attack) is a cyber-attack in which the attacker makes a server or resource temporarily unavailable to its users. According to the Computer Fraud and Abuse Act, DoS attacks are considered as a federal crime. This kind of crime penalties 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 an important issue in Network Forensics. Moreover, DoS attacks are increasing day by day in both number and size; CloudFlare [2] recently reported a 400 Gbps DoS attack that took place at their servers. Most powerful distributed DoS (DDoS) attacks are conducted by bots. Bots are malicious programs residing in users' computers. The bots are controlled by a coordinator called master. The master commands the bots to send a huge number of requests to the victim. As a result, the incoming bandwidth of the victim becomes exhausted. An attack scenario is depicted in Fig. 1.

An effective method of preventing DDoS attacks is deploying filter routers (FRs) in the network. FRs are a special types of routers that can do two tasks: packet marking and traffic filtering. Packet marking means probabilistically attaching the FR's IP address to the header of the packets it forwards. The task of traffic filtering happens in two steps: receiving filters from a web server and applying filters to drop some packets.

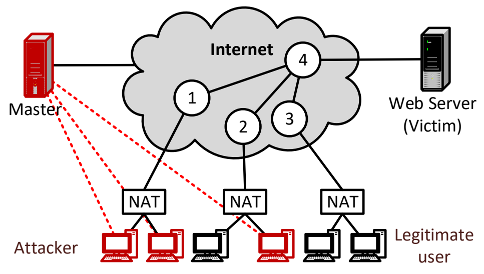


Fig. 1: DDoS attack by bots.

A server can deploy a filter to a FR. A filter is simply a blocking rule. A filter sent by any server is applied to packets which are destined for it. There are several types of filters including source-based and destination-based filters [3]. The source-based filters block packets based on the source address of the packet. The destination-based filters block packets based on the destination address of the packet. The source-based filters cannot protect when the bots spoof legitimate users' (LU) addresses. Therefore, in this work we are considering the destination based filters. The downside of destination-based filter is that it also blocks LU traffic. Therefore, the victim should deploy the destination-based filters wisely so that a minimum number of LUs are blocked.

The complete system consists of four phases. In the first phase, the FRs mark forwarded packets by appending their own IP addresses. In the second phase, from the marking of the packets, the victim constructs the traffic topology and filter. In the third phase, the victim finds and selects some FRs to assign the filters. This phase is important because a good filter assignment can reduce the cost of filter deployment and the number of blocked LUs. In the last phase, the FRs evict filters that are not used for a long time from their storage.

In this paper, we focus on finding an optimal destination-based filter assignment considering a limited budget on the number of filters and limited incoming bandwidth of the victim. We assume that the victim has already constructed the traffic topology. We formulate a problem and propose an optimal dynamic programming solution. We provide extensive simulation to support our model.

The remainder of this paper is arranged as follows: Section II presents some related works. Section III presents the system model and attack model. Section IV presents the formal definition of the problem and our proposed dynamic programming solution. In Section V, we present the simulation results that strengthen support for our proposed solutions.

II. Related Work

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 detection of DDoS attacks in [6]. A novel DDoS detection mechanism is proposed based on artificial neural networks in [7]. There are other methods of detecting DDoS attack such as [8, 9].

In [10], the authors propose a DDoS defense method, called RADAR. The RADAR uses adaptive correlation analysis on SDN switches. The system is capable of defending against flooding-based DDoS attacks including SYN flooding, link flooding, and amplification attacks. In [11], the authors propose a cooperative DDoS mitigation system. A domain directs traffic to other trusted external domains for filtering. The filtering system distinguishes the DDoS packets and removes them. The filtered clean traffic is then forwarded back to the destination domain.

A four-phase DDoS mitigation system is proposed in [12]. The system is composed of FRs. Packet marking is used to identify FRs. The victim generates and sends filters to the upstream FRs. The FRs then send the filters to their upstream FRs. After that the attack traffic is blocked by the upstream FR and the FRs do not get any attack traffic. The filter remains unused at that FR and gets evicted. Thus the filters propagate to the effective FRs. An adaptive PFS is proposed in [13]. In the adaptive PFS system, first the victim sends filters to the highly capable FRs, then the filters propagate to the effective FRs. The capability of a FR is determined by its remaining storage, the number of connected links etc. Highly capable FRs use a high rate of marking. The victim identifies FR's capability by its marking rate. However, these two systems do not select the FRs optimally when there is a limitation on selecting FRs.

In our previous work [3], we propose a dynamic programming algorithm that can select a subset of the FRs aiming to keep the number of blocked LUs and attack traffic minimum. The system can select the optimal filter assignment in some special settings. In work [14], optimal filter assignment policy is proposed for transit-link DDoS attack. The system considers that no attack traffic reaches the victim. The limitation of this system is that the assigned filters may block a large amount of LU traffic in order to block a small amount of attack traffic. Besides, the system does not consider the limited incoming bandwidth of the victim.

III. System Model

A. Network Model

Our system is composed of legacy routers (LRs), filter routers (FRs), network address translators (NATs), attackers, legitimate users (LUs), and a victim (v). In reality, there are multiple victims in a network. We consider each victim defends separately and does not coordinate/share information.

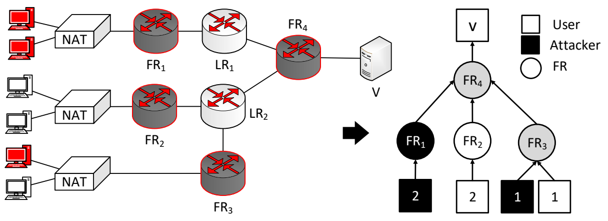


Fig. 2: System model.

There are two reasons why a victim does not coordinate with another. Firstly, victims can be owned by different entities and they might not be willing to share their users' information including IP address, location, and traffic rate with other entities' victims. Secondly, the attacker can share fake information with the victim to make wrong decisions. We consider that all the users and attackers are connected to the FR or LR through NATs. Therefore, the victim cannot get the IP address of users but instead the IP address of the NAT.

Fig. 2 shows the complete system model. In the figure, we can see that two attackers are connected to FR_1 through a NAT. The traffic travels $FR_1 \rightarrow LR_1 \rightarrow FR_4 \rightarrow v$. We ignore the LR and the path becomes $FR_1 \rightarrow FR_4 \rightarrow v$. Similarly, the traffic that comes to FR_2 and FR_3 travels LR_2 and FR_4. After ignoring all the NATs and LRs, we get the traffic topology (right figure). FRs are a special type of routers that is capable of doing two additional jobs: packet marking and traffic filtering. Packet marking means appending the router IP address to the header of a forwarded packet. Traffic filtering means blocking some traffic according to filters.

The complete system is composed of four phases. In **phase 1**, the FRs do packet marking. FRs append the IP addresses probabilistically. The probability of marking is adjusted to maintain a good balance of efficiency and router overhead. If the marking probability is very high, then the victim can construct the topology very quickly (efficient). At the same time, the router computation overhead becomes high. If the marking probability is very low, then the victim needs longer time to construct the topology but the router overhead is low.

In **phase 2**, the victim collects the marking from the packets and constructs the traffic topology. For example, if the victim gets a packet that is marked by FR_1 and FR_4, respectively, then it assumes that FR_1 remains before FR_4 on the path from the source of the packet. This way the victim can find out all the relative positions of FRs on the path from each source to it. As the LRs do not mark packets, they are not observed by the victim. By combining all the paths, it generates the traffic topology. The traffic topology can be a Directed Acyclic Graph (DAG). For simplicity, we are considering tree topology instead of a DAG. Using the methods described in [8, 9, 15–17] the victim distinguishes the attacker. We color the attackers as black and LUs as white. A black (or white) FR only forwards messages from attackers (or LUs). A gray FR forwards packets from both LUs and attackers. After identifying the attackers, the victim construct a filter. A filter is basically simple traffic blocking rule. For example, the victim constructs a filter that says "If the destination is v, then discard the packet".

In **phase 3**, the victim selects some of the FRs to send the filter. As the filter blocks the attack traffic based on destination, some of the LUs get blocked. Therefore, the victim needs to select FRs wisely so that the number of blocked LUs is minimum. The incoming bandwidth of a victim is limited. Therefore, the filter assignment should not yield a higher amount of traffic than the incoming bandwidth. There is also a cost for deploying a filter on a FR. If the owner of the FR is different than the victim, then it charges some money for deploying a filter. We consider the victim has a limited budget on the number of filters. In this paper, we focus on finding a filter assignment considering a limited budget (K) on the number of filters and limited incoming bandwidth of victim (B) that blocks the minimum amount of LU traffic.

In **phase 4**, the FRs remove unused filters from their storage. A FR can receive multiple filters from different victims. The storage and computation capacity is limited. Therefore, a FR removes the filter which is unused for a long period.

We assume that the data rates of LUs are identical. For simplicity, we consider the number of LUs is equal to the legitimate traffic.

B. Attack Model

The attackers are usually user devices with malicious programs that generate traffic destined for a target. The programs are controlled by a remote coordinator called master. These programs are called bots. The bots ask for commands from the master and according to the command it continues to send requests to the victim. The network of bots, called Botnet, is an overlay network. We assume the data rates of bots are identical. Therefore, the amount of traffic is proportional to the number of bots. For simplicity, we consider the number of bots is equal to the attack traffic.

IV. Filter Assignment Policy

In this section, we formulate a problem of assigning filters to the FRs so that the blocked user traffic is minimum.

A. Find a filter assignment so that the blocked user traffic is minimal by ensuring that the total incoming traffic at the victim is less than its capacity.

The problem can be expressed as the following:

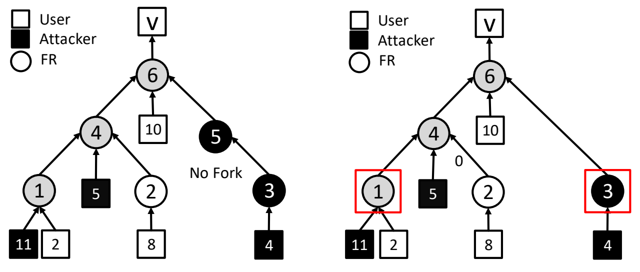
$$\begin{aligned} &\text{minimize} \quad U_b(g) \\ &\text{subject to} \quad |g| \leq K, \forall g \subset G, I(v) \leq B \end{aligned} \tag{1}$$

Here, $I(v)$ denotes the incoming traffic at victim v. The problem can be solved using dynamic programming. Next, we discuss the proposed solution of this problem.

B. A Dynamic Programming Solution

Let us assume that n is the FR ID, k is the number of filters, and b is the amount of yielded traffic. $T(n)$ is the subtree rooted by node n. We define the problem $P(n, k, b)$ as following:

- $P(n, k, b)$: Find and return the minimum number of possible blocked LUs using k filters in $T(n)$ by yielding b amount of traffic to the downstream node.



(a) Traffic topology. (b) After simplification.

Fig. 3: Solution example.

$P(n, k, b)$ is recursive. $P(n, k, b)$ depends on $P(c_i(n), k_i, b_i)$ where $c_i(n)$ is the i th child of n, k_i is the filter assigned to $T(c_i(n))$, and b_i is the traffic yielded by $T(c_i(n))$. There are two options to assign values of k_i. Firstly, k filters are divided among its children without assigning a filter to n itself. Secondly, $k-1$ filters are divided among its children and a filter is assigned to n itself.

According to option I, $\sum_{i=1}^{\Delta} k_i = k$ and $TL[n] + \sum_{i=1}^{\Delta} b_i = b$. $TL[n]$ is the traffic load of the users and attackers attached to node n. The minimum blocked LU $P(n, k, b)$ can be represented as following:

$$P(n, k, b) = \min_{\forall k_i, b_i} \left\{ \sum_{i=1}^{\Delta} P(c_i, k_i, b_i) \right\} \tag{2}$$

According to the option II, $\sum_{i=1}^{\Delta} k_i = k - 1$. The filter assigned to n blocks all traffic. Therefore, the yielded traffic amount is 0 for any value of b_i. This option is applicable only when $b = 0$. For $b > 0$, we choose k_i and b_i according to the first option. The minimum blocked LU $P(n, k, b)$ can be represented as following:

$$P(n, k, b) = \sum_{n' \in T(n)} L[n'] \tag{3}$$

We take the minimum quantity from the two options.

Let us consider a N node tree having maximum node degree Δ. The nodes are labeled in a bottom-up and then left-right order. We define A as a $N \times K \times B$ array which contains the minimum number of blocked LUs in the subtree rooted by every node, budget, and yielded traffic. For example, $A[n, k, b]$ contains the minimum number of blocked LUs in $T(n)$ of budget k by yielding b traffic.

We define TL as a $1 \times N$ array. TL contains the total traffic loads of LUs and attackers attached to every node.

We define R_1 as an $N \times K \times B \times (\Delta + 1)$ array that contains the number of filters assigned to every node and its subtrees, budget, and yielded traffic. For example, $R_1[n, k, b, \Delta + 1]$, $R_1[n, k, b, 0]$, and $R_1[n, k, b, 1]$ are the number of filters to the first subtree, the second subtree, and node n of subtree $T(n)$ for budget k by yielding b traffic.

We also define R_2 as an $N \times K \times B \times \Delta$ array which contains the yielded traffic at the minimum number of blocked LUs of its subtrees for every node, budget, and yielded traffic. The complete algorithm is shown in Alg. 1.

k \ b	0	13
0	∞	0
1	2	∞
2	2	∞
3	2	∞

A[1]

k \ b	0	13
0	-	0, 0, 0
1	0, 0, 1	-
2	0, 0, 2	-
3	0, 0, 3	-

R₁[1]

k \ b	0	13
0	-	0, 0
1	0, 0	-
2	0, 0	-
3	0, 0	-

R₂[1]

k \ b	0	8
0	∞	0
1	8	∞
2	8	∞
3	8	∞

A[2]

k \ b	0	8
0	-	0, 0, 0
1	0, 0, 1	-
2	0, 0, 2	-
3	0, 0, 3	-

R₁[2]

k \ b	0	8
0	-	0, 0
1	0, 0	-
2	0, 0	-
3	0, 0	-

R₂[2]

k \ b	0	4
0	∞	0
1	0	∞
2	0	∞
3	0	∞

A[3]

j k \ b	0	4
0	-	0, 0, 0
1	0, 0, 1	-
2	0, 0, 2	-
3	0, 0, 3	-

R₁[3]

k \ b	0	4
0	-	0, 0
1	0, 0	-
2	0, 0	-
3	0, 0	-

R₂[3]

Fig. 4: A, R_1, and R_2 for leaf nodes.

C. Example

Let us consider the tree in Fig. 3(a). We first simplify the tree. There is a node (node 5) without a fork. A node without a fork refers to the node having only one child. We need to remove node 5 first. Then we make node 3 the child of node 6. Finding out all non-forked nodes and deleting them takes $O(N)$ time. Therefore, the simplification takes $O(N)$ time.

Next, we calculate the A, R_1, and R_2 for the leaf nodes. We use $K = 3$ and $B = 26$ for the calculation. There are three leaf nodes 1, 2, and 3. We need to calculate the values of $A[n,k,b]$, $R_1[n,k,b]$, and $R_2[n,k,b]$ for all possible values. For node 1, there are two possible values of b (0 and 13). If we want to yield 0 traffic from $T(1)$, we need at least a filter at node 1. The filter will block all traffic from $T(1)$. The filter blocks 2 LUs. Therefore, for $k > 0$ and $b = 0$ the number of blocked LUs $A[1,k,0] = 2$. As we are assigning all the filters to node 1 and blocking all traffic, $R_1[1,k,0] = [0,0,k]$, and $R_2[1,k,0] = [0,0]$. Without any filter ($k = 0$) we cannot block any traffic in $T(1)$. Therefore, $A[1,0,0] = \infty$, $R_1[1,0,0] = [-]$, and $R_2[1,0,0] = [-]$ ($[-]$ means no valid assignment).

To yield 13 traffic from $T(1)$, no filters is needed at node 1. Therefore, for $k = 0$ and $b = 13$ the number of blocked LUs $A[1,0,13] = 0$ because no LU traffic is blocked. No filter is assigned to node 1, left, and right subtrees. Therefore, $R_1[1,0,13] = [0,0,0]$, and $R_2[1,0,13] = [0,0]$. For $k > 0$, we are assigning all the k filters to node 1 and cannot yield 13 traffic. Therefore, $A[1,k,13] = \infty$, $R_1[1,k,13] = [-]$, and $R_2[1,k,13] = [-]$. Similarly, we calculate the rest of the values for leaf nodes (see Fig. 4).

Next, we calculate the A, R_1, and R_2 for node 4. First, we find out the possible values of b. Values of b in left and right subtrees are $\{0,13\}$ and $\{0,8\}$. Combining them, we get $\{0,8,13,21\}$. There are some attackers with 5 traffic load attached to node 4. This traffic will be added to the traffic yielded from subtree if no filter is assigned to node 4. In this way, possible values of b are $\{5,13,18,26\}$. If a filter is assigned to node 4 then b is 0. Then, all possible values of b for node 4 are $\{0,5,13,18,26\}$.

For $k = 0$ and $b < 26$, without any filter, any portion of the traffic is not possible to block. Therefore, $A[4,0,b] = \infty$, $R_1[4,0,b] = [-]$, and $R_2[4,0,b] = [-]$. If $b = 26$, then we are not blocking any traffic. Without any filter that is possible and there is no blocked LU. Therefore, $A[4,0,26] = 0$, $R_1[4,0,26] = [0,0,0]$, and $R_2[4,0,26] = [0,0]$.

For $k = 1$ and $b = 0$, we have only option II. Therefore, a filter is assigned to node 4 and we block all traffic including 10 LUs. After assigning a filter to 4, there are no remaining filter. So, $T(1)$ and $T(2)$ are assigned 0 filters ($k_1 = 0, k_2 = 0$). Next, we need to find values of b_1 and b_2. Any value of b_1 and b_2 will work because all traffic will be blocked at node 4. We choose $b_1 = 13$ and $b_2 = 8$. Therefore, $A[4,1,0] = 10$, $R_1[4,1,0] = [0,0,1]$, and $R_2[4,1,0] = [13,8]$.

For $k = 1$ and $b = 5$, we have only option I. Therefore, no filter is assigned to node 4. We have two choices for k_1 and k_2. **Choice (1):** $k_1 = 0$ and $k_2 = 1$. Node 4 is attached to attackers with 5 traffic loads. So, the subtrees need to yield $5 - 5 = 0$ traffic. We have only one choice for b_1 and b_2 ($b_1 = 0$ and $b_2 = 0$). The minimum blocked LU in $T(1)$ using $k_1 = 0$ and yielding $b_1 = 0$ traffic is $A[1,0,0] = \infty$. The minimum blocked LU in $T(2)$ using $k_2 = 1$ and yielding $b_2 = 0$ traffic is $A[2,1,0] = 0$. According to this choice, the number of total blocked LU is ∞.

Choice (2): $k_1 = 1$ and $k_2 = 0$. The subtrees need to yield $5 - 5 = 0$ traffic and we have only one choice for b_1 and b_2 ($b_1 = 0$ and $b_2 = 0$). The minimum blocked LU in $T(1)$ using $k_1 = 1$ and yielding $b_1 = 0$ traffic is $A[1,1,0] = 2$. The minimum blocked LU in $T(2)$ using $k_2 = 0$ and yielding $b_2 = 0$ traffic is $A[2,0,0] = \infty$. According to this choice, the number of total blocked LU is ∞.

We take the minimum of the two choices which is ∞. Therefore, the number of total blocked LU is $A[4,1,5] = \infty$. So, $R_1[4,1,5]$ and $R_2[4,1,5]$ are invalid.

For $k = 1$ and $b = 13$, we also have only option I. Similarly, we have two choices for k_1 and k_2.

Choice (1): $k_1 = 0$ and $k_2 = 1$. Node 4 is attached to attackers with 5 traffic loads. So, the subtrees need to yield $13 - 5 = 8$ traffic. We have only one choice for b_1 and b_2 ($b_1 = 0$ and $b_2 = 8$). The minimum blocked LU in $T(1)$ using $k_1 = 0$ and yielding $b_1 = 0$ traffic is $A[1,0,0] = \infty$. The minimum blocked LU in $T(2)$ using $k_2 = 1$ and yielding $b_2 = 8$ traffic is $A[2,1,8] = \infty$. According to this choice, the number of total blocked LU is ∞.

Choice (2): $k_1 = 1$ and $k_2 = 0$. The subtrees need to yield $13 - 5 = 8$ traffic and we have only one choice for b_1 and b_2 ($b_1 = 0$ and $b_2 = 8$). The minimum blocked LU in $T(1)$ using $k_1 = 1$ and yielding $b_1 = 0$ traffic is $A[1,1,0] = 2$. The minimum blocked LU in $T(2)$ using $k_2 = 0$ and yielding $b_2 = 8$ traffic is $A[2,0,0] = 0$. According to this choice, the number of total blocked LU is 2.

We take the minimum of the two choices which is 2. Therefore, the number of total blocked LU is $A[4,1,13] = 2$. So, $R_1[4,1,5]$ is $[1,0,0]$ and $R_2[4,1,5]$ is $[0,8]$.

Similarly, we calculate the remaining entries of A, R_1, and R_2. Fig. 5 shows the complete values of A, R_1, and

k \ b	0	5	13	18	26
0	∞	∞	∞	∞	0
1	10	∞	2	8	∞
2	10	10	2	8	∞
3	10	10	2	8	∞

A[4]

k \ b	0	5	13	18	26
0	-	-	-	-	0, 0, 0
1	0, 0, 1	-	1, 0, 0	0, 1, 0	-
2	1, 0, 1	1, 1, 0	2, 0, 0	0, 2, 0	-
3	2, 0, 1	2, 1, 0	3, 0, 0	0, 3, 0	-

R₁[4]

k \ b	0	5	13	18	26
0	-	-	-	-	13, 8
1	13, 8	-	0, 8	13, 0	-
2	0, 8	0, 0	0, 8	13, 0	-
3	0, 8	0, 0	0, 8	13, 0	-

R₂[4]

k \ b	0	10	14	15	19	23
0	∞	∞	∞	∞	∞	∞
1	20	∞	10	∞	∞	∞
2	20	10	10	∞	10	2
3	20	10	10	10	10	2

A[6]

k \ b	0	10	14	15	19	23
0	-	-	-	-	-	-
1	0, 0, 1	-	1, 0, 0	-	-	-
2	1, 0, 1	1, 1, 0	2, 0, 0	-	2, 0, 0	1, 1, 0
3	2, 0, 1	2, 1, 0	3, 0, 0	2, 1, 0	3, 0, 0	2, 1, 0

R₁[6]

k \ b	0	10	14	15	19	23
0	-	-	-	-	-	-
1	26, 4	-	0, 4	-	-	-
2	13, 4	0, 0	0, 4	-	5, 4	13, 0
3	13, 4	0, 0	0, 4	5, 0	5, 4	13, 0

R₂[6]

Fig. 5: A, R_1, and R_2 for nodes 4 and 6.

Algorithm 1 DP Blocking Strategy

```
Input: The number of filters $K$, bandwidth limit $B$, and topology tree $T$.
Output: A set of nodes in $T$.
1: Procedure: BLOCK-DP($K, B, T$)
2:     $N \leftarrow$ number of nodes in $T$
3:     for every leaf node $n$ do
4:         Initialize $TL[n]$.
5:         for $k = 0$ to $K$ do
6:             for $b = 0$ to $B$ do
7:                 Initialize $A[n, k, b]$, $R_1[n, k, b]$, and $R_2[n, k, b]$
8:     CALC($N, K, B$)
9:     return ASSIGNMENT($R_1, R_2, N, K, B$)
```

Algorithm 2 Calculate A, R_1, and R_2

```
1: Procedure: CALC($N, K, B$)
2:     for $n = 1$ to $N$ do
3:         for $k = 0$ to $K$ do
4:             for $b = 0$ to $B$ do
5:                 $min \leftarrow= \infty$, $map \leftarrow \emptyset$
6:                 for $\forall k_i, b_i : \sum_{i=1}^{\Delta} k_i = k \sum_{i=1}^{\Delta} b_i = b - TL[n]$ do
7:                     $p \leftarrow \sum_{i=1}^{\Delta} A[c_i(n), k_i, b_i]$
8:                     $key \leftarrow [k_1, k_2, ..., k_\Delta, 0, b_1, b_2, ..., b_\Delta]$
9:                     PUT($map, key, p$)
10:                for $\forall k_i, b_i : \sum_{i=1}^{\Delta} k_i = k - 1$ do
11:                    $p \leftarrow \sum_{n' \in T(n)} L[n']$
12:                    $key \leftarrow [k_1, k_2, ..., k_\Delta, 1, b_1, b_2, ..., b_\Delta]$
13:                    PUT($map, key, p$)
14:                $A_1[i, j, b] \leftarrow$ MIN($map$)
15:                $R_1[i, j, b]$, $R_2[i, j, b] \leftarrow$ ARGMIN($map$)
```

R_2 for node 4 and 6. As B is 26, we do not need to consider $b > 26$. All possible values of b for node 6 are $\{0, 10, 14, 15, 19, 23, 27, 28, 32, 36, 40\}$. The highest possible value of $b <= 26$ is 23. That is why we calculate up to 23 yielded traffic for node 6.

D. Assignment Set Formulation

We find the filter assignment set for $K = 3$ and $B = 26$. We cannot get any assignment that yields 26 traffic. The highest amount of traffic that does not exhaust the capacity of the victim is 23. $R_1[6, 3, 23] = [2, 1, 0]$, which means $T(4)$ and $T(3)$ is assigned 2 and 1 filters, respectively. $R_2[6, 3, 23] = [13, 0]$ means $T(4)$ and $T(3)$ are yielding 13 and 0 traffics, respectively. Now, we look up $R_1[4, 2, 13] = [2, 0, 0]$, which means $T(1)$ is assigned to 2 filters. $R_2[2, 2, 13] = [0, 8]$ means $T(1)$ and $T(2)$ are yielding 0 and 13 traffic, respectively. Then, we look up $R_1[1, 2, 0] = [0, 0, 2]$, which means node 1 is assigned to 2 filters. We got one of our desired nodes and the assignment set is $\{1\}$. Now, we look up the other branch of node 6. $R_1[3, 1, 0] = [0, 0, 1]$, which means node 3 is assigned to 1 filter. Therefore, our assignment set is $\{1, 3\}$. The algorithm is shown in Alg. 3.

Algorithm 3 Find Filter Assignment

```
1: Procedure: ASSIGNMENT($R_1, R_2, N, K, B$)
2:     $e.n \leftarrow N, e.k \leftarrow K, e.b \leftarrow B$, $g \leftarrow \emptyset$, and $Q \leftarrow \emptyset$.
3:     ENQUEUE($Q, e$).
4:     while $Q \neq \emptyset$ do
5:         $e \leftarrow$ DEQUEUE(Q)
6:         if $R_1[e.n, e.k, e.b, \Delta + 1] \neq 0$ then
7:             $g \leftarrow g \cup \{e.n\}$
8:         else
9:             for $i = 1$ to $\Delta$ do
10:                $e'.n \leftarrow c_i(e.n)$
11:                $e'.k \leftarrow R_1[e.n, e.k, e.b, i]$
12:                $e'.b \leftarrow R_2[e.i, e.j, e.b, i]$
13:                ENQUEUE(Q, $e'$)
14:    return $g$
```

Theorem 1. *Alg. 1 provides an optimal filter assignment.*

Proof. Alg. 1 first calculates the assignment and the number of blocked LUs for base cases. Then it uses a dynamic programming bottom-up strategy to search the optimal assignment. We need to show that for base cases the assignment is optimal. For a one-node tree, $K = 0$, and $B = 0$, there is no way to block all traffic and this option is invalid. For $K = 0$ and B is the maximum, there is no way to block any traffic and this option is valid as the yielded traffic will be maximum. No LU traffic is blocked and the optimal number of blocked LUs is 0. For $K \geq 1$ and $B = 0$, 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. For $K \geq 1$ and $B > 0$, the filter blocks all traffic and yielded traffic is 0. These options are also invalid. Therefore, we can conclude that all base cases are optimal so that Alg. 1 provides an optimal filter assignment. ☐

Theorem 2. *Complexity and space needed for Alg. 1 are $O(N(KB)^{(\Delta-1)})$ and $O(NKB\Delta)$.*

Proof. Let us assume a topology that is a N node tree with maximum node degree Δ. The victim has budget of K and

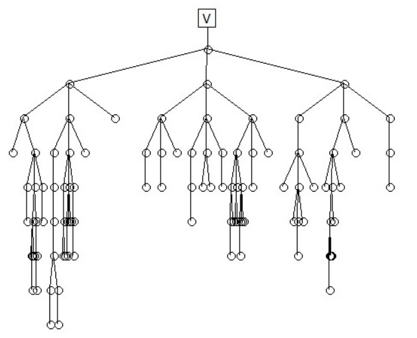


Fig. 6: Topology I (randomly generated).

TABLE I: Topology Parameters

	Topology I	Topology II
Number of nodes	100	400
Internal user probability	0.1	0.1
Attacker ratio	0.25-0.75	0.25-0.75
Max Node Degree	3	20

the maximum bandwidth capacity of the victim is B. To find the partitions $k_1, k_2, ..., k_\Delta$, we need $O(K^{(\Delta-1)})$ time if we use the naive nested iteration approach. Similarly, to find the optimal yielded traffic $b_1, b_2, ..., b_\Delta$, we also need $O(B^{(\Delta-1)})$ time. Therefore, the complexity of Alg. 1 is $O(N(KB)^{(\Delta-1)})$.

For A, R_1, and R_2 we need NKB, $NKB(\Delta+1)$, and $NKB\Delta$ space. For TL we need N space. Therefore, total space needed is $O(NKB\Delta)$. $\square$

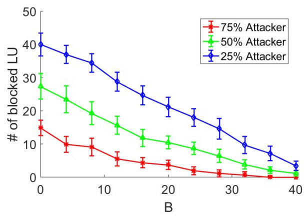
V. Simulation

A. Experimental Settings

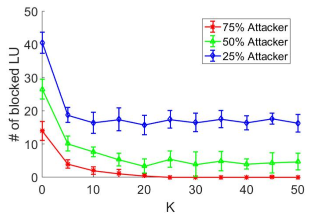
We use our custom built Java simulator for all the experiments. The reason for using a custom built simulator is its scalability. Besides, we do not need to consider packet loss or real transmission issues. We only need to count the number of LUs, attackers, and filters. The network topologies we considered contain $10-500$ routers. NS3 or other similar simulators would take a very long time.

We use the randomly generated the topologies. We control the number of nodes, depth, and maximum node degree of randomly generated trees. At first we generate the desired number of nodes and keep them in a node pool. We pick a node from the pool and add to a randomly selected eligible node in the tree. The first node becomes the root. Initially, all nodes are eligible for adding a child node. The nodes which have a node degree equal to the desired node degree are marked as ineligible. The nodes which are at the desired depth are also marked ineligible. We keep node degree within $[0-3]$, internal node user probability 0.1 and maximum depth of 8. Topology I is a randomly generated tree having 100 nodes and maximum node degree of 3. Topology II is taken from Stanford University AS-733 dataset [18]. We run a breadth-first search to generate the topology. The dataset contains $6,474$ nodes and we consider 400 nodes for topology II. The details are shown in Table I. Topology I is shown in Fig 6. Due to the higher number of nodes, Topology II is not shown.

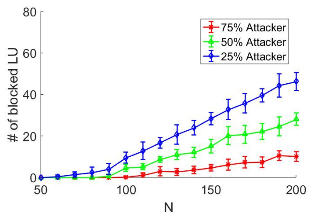
We control the number of LUs and attackers. We add the leaf nodes and 10% of internal nodes (randomly chosen) to the



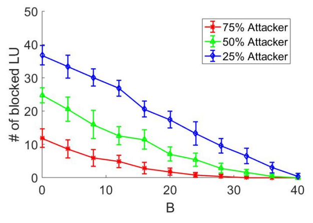
(a) Effect of B (Topology I).



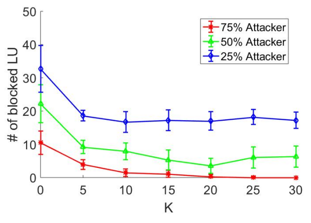
(b) Effect of K (Topology I).



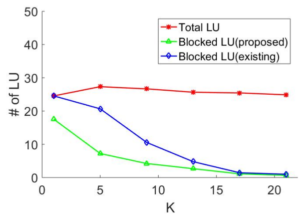
(c) Effect of topology size.



(d) Effect of B (Topology II).



(e) Effect of K (Topology II).



(f) Comparison with an existing work.

Fig. 7: Simulation results.

pool. We pick randomly a node from the pool and deploy an LU and/or an attacker. The process continues until the desired number of LUs and attackers is reached.

We measure the performance of our proposed solution in terms of blocked LU traffic. We vary the number of filters and bandwidth limit of the victim for the two topologies and observe performance changes. We also observe the performance by changing the size of the topology. All the measurements are average of 100 rounds of simulation.

B. Simulation Results

In our proposed model, we have two parameters for the filter assignment: the number of filters (K) and victim's bandwidth (B). First, we change these parameters and observe the effect on the number of blocked LUs. We use both topologies for these experiments. We considered that each leaf node has one or more LUs. Every LU and attacker has an identical data rate.

Fig. 7(a) shows the number of blocked LUs by the victim's bandwidth. We keep the number of filters at 10. We vary the victim's bandwidth from 0 to 40. The total number of attackers and LUs remains the same. Therefore, if the number of attackers is large, then the number of LUs is small. The highest number of blocked LUs is with 25% attackers, the lowest is 75%, and 50% is in between for all bandwidth. The higher the victim's bandwidth, the more the users can get to it. As a result, a higher number of bandwidth produces a lower number of blocked LUs. For 25% attackers, the average total number of LUs is 40 with a variance of 3.4. If the victim's bandwidth is 40, the number of blocked LUs is 3.4. Therefore, the blocked LUs are 8%. For 50% attackers, the total number of LUs is 27.36 with a variance of 3.86. If the victim's bandwidth is greater than 15, the number of blocked LUs is 3.8. Therefore, the blocked LUs are 13%. For 75% attackers, the total number of LUs is 14.89 with a variance of 2.33. If the victim's bandwidth is greater than 15 the number of blocked LUs is 4.3. Therefore, the blocked LUs are 34%. We observe similar behavior for Topology II (see Fig. 7(d)). We limit the number of users in this experiment. We limit the number of attackers and LUs to 100. The amount of blocked LUs is lower in topology II.

Fig. 7(b) shows the number of blocked LUs by the number of filters. We keep the victim's bandwidth at 40. We vary the number of filters from 1 to 50. Similarly, the highest number of blocked LUs is with 25% attackers, the lowest is 75%, and 50% is in between for all numbers of filters. The higher the number of filters, the more options of placing filters. As a result, a higher number of filters produces a lower number of blocked LUs. After a certain number of filters, the number of blocked LUs does not reduce. For 25% attackers, the total number of blocked LUs reduces from 40.5 to 16.42 during $K = [1, 10]$. After that the number of blocked LUs remains almost constant. For 50% attackers, the total number of blocked LUs reduces from 26.4 to 3.36 during $K = [1, 20]$. After that the number of blocked LUs remains almost constant. For 25% attackers, the total number of blocked LUs reduces from 13.94 to 0.0 during $K = [1, 25]$. After that the number of blocked LUs remains almost constant. We also observe similar behavior for Topology II (see Fig. 7(e)). We also limit the number of attackers and LUs to 100. The amount of blocked LUs is also lower in topology II. The number of filters after which the number of blocked LUs becomes constant is little different than topology I.

Fig. 7(c) shows the number of blocked LUs by the number of nodes. We keep the victim's bandwidth at 40 and the number of filters at 20. We vary the number of nodes from 50 to 200. Similarly, the highest number of blocked LUs is with 25% attackers, the lowest is 75%, and 50% is in between for all numbers of nodes. The higher the number of nodes, the higher the number of LU and attackers. As a result, a larger topology produces a higher number of blocked LUs.

Fig. 7(f) shows the number of blocked LUs by the number of filters of our proposed model and an existing model. The existing model is one of our previous work [3]. In the existing model, we consider that no attack traffic can reach the victim. In this paper, we relax that constraint and observe a good improvement. In this experiment, we keep the bandwidth of the victim equal to the total LU traffic. We can observe that the blocked LUs in the existing model is higher than the proposed mode. Therefore, we can conclude that by allowing some attack traffic the victim can serve more LUs.

VI. Conclusion

The DDoS attack is a powerful and inexpensive attack that makes a service unavailable to users. The victim alone cannot defend against DDoS attacks. With the help of a special type of routers, the victim can defend against DDoS attacks. We propose an optimal filter assignment policy that focuses on minimizing blockage of legitimate users with a limited cost of filter deployment and incoming bandwidth of the victim. The filter routers can work in a network with legacy routers and legacy router can be upgraded to FRs. One of the important issues is that the blocking of traffic happens according to the victim's command. An internet service provider (ISP) provides some control of the packets to the server. By deploying FRs, an ISP can earn money by accepting filters. Therefore, both parties gets benefit from the model.

Acknowledgments

This research was supported in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1618398, CNS 1651947, and CNS 1564128.

References

[1] US Code: Title 18,1030, "Fraud and related activity in connection with computers — government printing office," www.gpo.gov, 2014.
[2] "Cloudflare," blog.cloudflare.com.
[3] R. Biswas and J. Wu, "Filter assignment policy against distributed denial-of-service attack," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems*, Dec 2018.
[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] P. Ning and S. Jajodia, "Intrusion detection techniques," 2004.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] ——, "APFS: Adaptive Probabilistic Filter Scheduling against distributed denial-of-service attacks," *Computers Security*, vol. 39, Nov 2013.
[14] R. Biswas, J. Wu, W. Chang, and P. Ostovari, "Optimal filter assignment policy against transit-link distributed denial-of-service attack," in *IEEE Global Communications Conference*, Dec 2019.
[15] 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.
[16] 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.
[17] 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.
[18] "Autonomous systems AS-733," https://snap.stanford.edu/data/as-733.html.