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

Rajorshi Biswas[1], Jie Wu[1], Wei Chang[2], and Pouya Ostovari [3]
[1]Department of Computer and Information Sciences, Temple University
[2]Department of Computer Science, Saint Joseph's University
[3] Charles W. Davidson College of Engineering, San Jose State University

***Abstract*—A transit-link distributed denial-of-service (DDoS) attack is a special attack in which the attacker sends out a huge number of requests to exhaust the capacity of a link on the path the traffic comes to a server. As a result, denial-of-service and degradation of Quality-of-Service (QoS) occurs. Because the attack traffic does not go to the victim, protecting the legitimate traffic alone is hard for the victim. With the help of a special type of router called filter router (FR), the victim can protect the legitimate traffic. A FR can receive filter from servers and apply the filter to block a link incident to it. By analyzing traffic rates and paths, the victim can identify some links that may be congested. The victim needs to select some of these possible congested links and send a filter to the corresponding FR so that the legitimate traffic follows non-congested paths. In this paper, we formulate an optimization problem for selecting the minimum number of possible congested links so that the legitimate traffic goes through a non-congested path. We consider the scenario where every user has at least one non-congested shortest path. We transform the problem to the vertex separation problem to find the links to block. We build our own Java multi-threaded simulator and conduct extensive simulations.**

***Index Terms*—Botnet, DDoS defence, Quality-of-Service, filter router, link flooding attack, network security, transit-link DDoS.**

I. Introduction

The growing amount of distributed denial-of-service (DDoS) attacks is a threat to critical services on the Internet. The primary objective of a DDoS attack is to generate a lot of packets from different workers to exhaust incoming/outgoing bandwidth or resources used by the victim. 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 the botnet. Bots are usually computers that are infected with malicious programs. Because of the huge amount of traffic or service requests, a machine or network resource becomes temporarily unavailable to its users.

DDoS attacks have evolved due to a decrease in the cost of conducting an attack. The transit-link DDoS attack using botnet is one of the most challenging DDoS attacks. The strategy and goal are slightly different from the traditional DDoS attack. In the traditional DDoS attack, the bots generate packets destined to the victim. In a transit-link DDoS attack, the bots generate traffic that is not destined to the victim but the packets congest at least one of the links on the paths from the legitimate users to the victim. Because of the congested links, the legitimate traffic suffers from packet drop and delay. The transit-link DDoS can be so devastating that it can disconnect a server from the Internet.

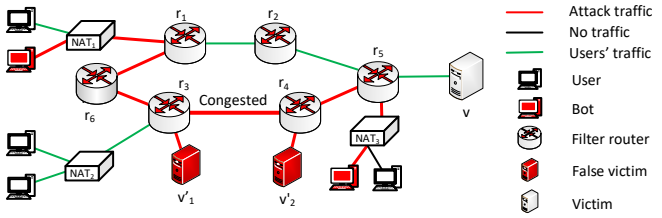


Fig. 1: Transit-link DDoS attack by bots.

Fig. 1 shows the scenario of a transit-link DDoS attack. The bots attached to NAT_1 and NAT_3 generate huge traffic towards the false victim server v_2' and v_1', respectively. False victim servers are used to receive the attack packets from bots. The attack packets congest link $r_3 \leftrightarrow r_4$. The legitimate packets from the user attached to NAT_2 travel the shortest path $r_3 \leftrightarrow r_4 \leftrightarrow r_5$ and face delay, packet drop, and DoS. The legitimate traffic from users attached to NAT_1 and NAT_3 do not encounter any congested links and the victim observes good traffic rate from them. Because of low data rates from users attached to NAT_3, the victim assumes that either any or both of the $r_3 \leftrightarrow r_4$ and $r_4 \leftrightarrow r_5$ links are congested.

The victim can disable a link for the packets destined to him by sending a filter to the corresponding filter router (FR). The FR and filter-based system are not new and well studied in [1, 2]. Assigning a filter to a FR has a cost. For the victim, it costs a lot to disable all the possible congested links. For example, if the victim and FR belong to different ISP, then ISP of FR can charge money for accepting filters from the victim. Besides, there is no need to disable all the possible congested links to redirect the legitimate traffic through non congested paths. In this paper, we focus on directing all legitimate traffic through non-congested paths with a minimum number of filters.

In this paper, we formulate a problem for directing legitimate traffic through the shortest non-congested paths with a minimum number of filters/blockage. We transform the problem into a vertex separator problem and propose an optimal solution. We provide extensive simulation with synthetic and real topology and compare it with the existing approaches to validate our model.

The remainder of this paper is as follows: Section II presents some related works. In Section III, we present the system model for preventing transit-link DDoS attacks. In Section IV and V, we formulate a problem in a special scenario where there is a non-congested shortest path from every user, and we provide a solution. In Section VI, we present the simulation results to support our model.

II. Related Work

There are many multi-path routing mechanisms that helps mitigating congestion, including [3, 4]. These schemes do not give priority to legitimate traffic and the destination machine does not have any control over it. There are several methods that employ routers to identify attack traffic and block them. The SPIFFY [5] logically increases link capacity when it detects congestion. After the capacity increases, the attacker may increase the data rate of each bot to keep the cost constant and be identified. The ColDef [6] enables routers to distinguish low-rate attack flows from legitimate flows. In this mechanism, the domains which are uncontaminated by botnet help route the legitimate traffic. In [7], the authors propose PUSHBACK, a router functionality based mechanism which enables each router to detect and preferentially drop packets that likely belong to an attacker. The upstream routers are also notified of the drop event so that they can use the resources for legitimate traffic. In [8], the authors propose a router subsystem called FLoc (Flow Localization) that provides differential bandwidth guarantee to legitimate traffic at congested links. The FLoc can distinguish between the traffic from a bot-contaminated domain and that from an uncontaminated one.

In [9], the authors propose a scheme which mitigates transit-link DDoS by using some BGP rules in BGP routers. When a link congestion is detected, the BGP router advertises its neighbors in such a way that the congested link is avoided. When a BGP router detects congestion, it creates a false path by appending its own address and advertises the false path. Other BGP routers from different autonomous systems find that the path has a loop and avoid it.

We discussed two types of existing defense systems: (1) attack packet/flow detection at a router followed by packet drop, rate control, and redirection, and (2) identification of congested links at a router or controller and avoiding the link. The first type increases router computation overhead and the false positive creates great loss to the victims. Besides, bots are smart enough to create traffic similar to users and are unidentifiable using statistical techniques. For the second type, the identification of the congested link for a victim far from the congested link is tough without additional access. The victim can distinguish between the attacker and legitimate traffic. Therefore, a defense system in which routing is controlled by the victim is necessary.

The most related work preventing DDoS attacks is the probabilistic filter scheduling (PFS) system [10]. The victim generates filters and sends them to the upstream FRs. The FRs send the filters to its upstream FRs and thus the filters propagate to the effective FRs. An adaptive version of PFS is proposed in [11]. The system directly sends filters to the high 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. In [12], multiple filter assignment policies are presented considering a limited budget on filters. These works are not applicable for transit-link DDoS protection.

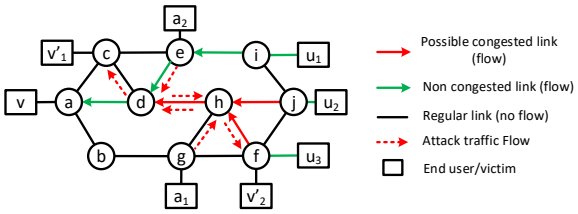


Fig. 2: System model.

III. System Model

A. Network Model

Our network is composed of network address translators (NATs), filter routers (FRs), bots, users, and a victim. We assume that all the users are connected to FRs through NAT. We also assume that there is at least one non-congested path from every user to the victim.

The FRs are a special kind of router capable of two functions. Firstly, they can mark packets that are used to construct traffic topology for the victim. Secondly, they can receive filters from the victim and apply the filters to block a link. A filter is a special instruction to a FR to block an incident link. When the victim v sends a filter to a FR, any packet destined to the v will be affected by the filter. For example, in Fig. 2, assume a filter "block link $h \leftrightarrow d$" is sent to a FR h by v. The FR h may forward an incoming packet through link $h \leftrightarrow d$ if the destination is not v. Therefore, the attack traffic from a_1 heading to the false victim v_1' is not blocked. If the packet is destined to v, then the FR will never forward the packet through the link $h \leftrightarrow d$. FR h will not forward user traffic from u_2 through $h \leftrightarrow d$.

Sometimes an attacker can impersonate v and send filter to FRs but it is identifiable using a simple handshaking protocol. For example, an attacker may send a wrong filter by sending a filter request message as the victim. When a filter request comes to a victim, it will ask the sender whether it has sent the filter or not. The attacker will not get the verification message and will not succeed in the filter spoofing attack.

We assume that the victim knows the network topology. The topology and traffic topology (the way the traffic comes to the victim) can be obtained from packet marking. From the packet arrival rate, the victim can identify the users that are suffering from congestion. It is hard for the victim to know which links are congested using the data rate of users. For example, in Fig. 2, the attackers congest the link $d \leftrightarrow h$. As a result, traffic from u_2 and u_3 suffer from congestion. The victim v observes that data rates from u_2 and u_3 are below the regular data rate. Therefore, one or multiple links on the path $a \leftrightarrow d \leftrightarrow h \leftrightarrow j$ and $a \leftrightarrow d \leftrightarrow h \leftrightarrow f$ may be congested. The victim also observes that the data rate from u_1 is good. Therefore, none of the links on $a \leftrightarrow d \leftrightarrow e \leftrightarrow i$ are congested. With this observation, the victim concludes that $d \leftrightarrow h$, $h \leftrightarrow j$, or $h \leftrightarrow f$ links have possibility to be congested.

After finding the possible congested links, the victim wants his incoming traffic to avoid the possible congested links. A trivial solution is to block all the possible congested links. For

example, if v wants to disable link $d \leftrightarrow h$, it would send a filter "block $d \leftrightarrow h$" to FR h. Sending a filter to FR has a cost. If the victim and FR belong to different ISP, then ISP of FR charges money for accepting filters from the victim. There are too many FR routers in the network, and the victim wants to reduce the filtering cost. Moreover, there is no need to block all of the possible congested links. For example, if v blocks $h \leftrightarrow j$ and $f \leftrightarrow h$, then all legitimate traffic will follow non-congested paths. Therefore, blocking $d \leftrightarrow h$ is unnecessary. The victim wants the legitimate traffic to follow the non-congested path by blocking a minimum number of links by spending the minimum amount of resources. The victim can run a centralized algorithm to find the minimum amount of required blocking.

The attack traffic continues to follow the same path because the filters do not affect any packet that is not destined to the victim. Therefore, the system does not reduce network congestion by blocking attack traffic. It only forces the legitimate traffic to follow a non-congested path. Whenever we mention blocking a link, we refer to blocking only the victim's legitimate traffic through that link.

B. Attack Model

The attacker knows the topology of the network but does not know the traffic topology. The attacker knows the location of the victim and users. By analyzing the location of the victim and users, the attacker selects one or multiple links as a target. Target links are selected based on two principles. Firstly, the target links should be used by a reasonable number of users. Secondly, the target links should be reachable by the flow from bots to the false victims. Then, based on the location of the target link, the attacker selects false victims and bots to generate traffic. The selection of a pair $< bot, false\,victim >$ is done in such a way that the traffic from the bot to the false victim passes through one or multiple targeted links.

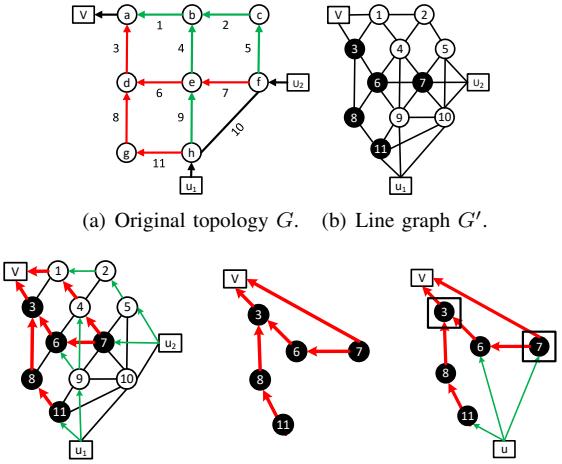
In Fig. 2, after analyzing the topology, location of the victim, and the users, the attacker finds that link $a \leftrightarrow d$ carries the most legitimate traffic but cannot be reached by the false victim and bots. The second most important link is $d \leftrightarrow h$ which carries two legitimate flows and is reachable by bots and the false victims. Therefore, the attacker starts $< a_1, v_1' >$ and $< a_2, v_2' >$ flows by sending commands to a_1 and a_2. $< a_1, v_1' >$ and $< a_2, v_2' >$ pass through paths $g \leftrightarrow h \leftrightarrow d \leftrightarrow c$ and $e \leftrightarrow d \leftrightarrow h \leftrightarrow f$. The traffic created by a bot cannot congest a link so that $c \leftrightarrow d$, $e \leftrightarrow d$, $g \leftrightarrow h$, and $f \leftrightarrow h$ are not congested. Link $d \leftrightarrow h$ becomes congested because both flows pass through it. As a result, legitimate traffic from u_2 and u_3 suffer from low data rates.

IV. Problem Definition

In this section, we formulate the problem for finding a minimum number of links to block.

Problem: *Find the minimum number of filter assignments so that all legitimate traffics follow non-congested shortest paths.*

In this problem, we assume that, from every user, there is at least one non-congested shortest path. Let, P_u^s be the set of the shortest paths from a user u to victim v. L_c is the set of possible congested links. Some of the paths in P_u^s contain possible congested links and are defined as possible congested paths. After deploying B number of filters to FRs in G, the set of the shortest paths from u is $P_u^s(B)$.



(a) Original topology G. (b) Line graph G'.

(c) G' with flow. (d) G'(black nodes). (e) Final graph G''

Fig. 3: An example.

In this case, the problem can be expressed as the following optimization problem:

$$\begin{aligned} &\text{minimize} && B \\ &\text{subject to} && \forall_{1 \le u \le U,\ p \in P_u^s(B)} \quad p \cap L_c = \emptyset, \\ & && P_u^s(B) \subseteq P_u^s \end{aligned} \tag{1}$$

V. Optimal Solution for the Problem

Let us consider the topology in Fig. 3(a). The victim is connected to the FR a. A couple of users are connected to the FR h and FR f. The victim v observes that when traffic from u_1 follows the paths $h \leftrightarrow g \leftrightarrow d \leftrightarrow a$ or $h \leftrightarrow e \leftrightarrow d \leftrightarrow a$, the data rate falls below a threshold. Similarly, if the traffic from u_2 follows $f \leftrightarrow e \leftrightarrow d \leftrightarrow a$ the data rate falls below the threshold. On the other hand, the rate is good if the traffic from u_1 and u_2 follow $h \leftrightarrow e \leftrightarrow b \leftrightarrow a$ and $f \leftrightarrow c \leftrightarrow b \leftrightarrow a$, respectively. From this observation, v finds out that links 3, 6, 7, 8, and 11 are possible congested links. Because of limited knowledge or access, the v cannot detect the actual congested links. A victim in an autonomous system might not know the status of a link in other autonomous systems. For example, this situation could happen if only links 3, 6, and 8, or 7 and 11 are congested. The v does not want to compromise data rates from its users and wants all the legitimate traffic coming to it to follow non-congested paths. We color the congested links red and non-congested links green. The black links do not carry any legitimate traffic and are assumed to be non-congested.

We divide the process of determining the minimum number of filter assignments into four steps.

In Step 1, we transform the original graph $G = (V, E)$ to line graph $G' = (V', E')$. V is the set of FRs and E is the set of links in the topology. The edges in G become the nodes in G'. Two nodes are neighbors in G' if the corresponding links in G are adjacent. We color the corresponding node of a possible congested link as black and other nodes as white. In

Algorithm 1 Find a minimum number of blockage

Input: Topology G, possible congested links L_c.
Output: A set of links to block from L_c.
1: **Procedure:** FIND-ASSIGNMENT(G)
2: $G' \leftarrow$ the line graph of G.
3: CREATE-FLOW(G')
4: $G'' \leftarrow$ ELIMINATE(G')
5: $g \leftarrow$ minimum vertex separation set in G''
6: **return** g

Algorithm 2 Create flow in the line graph

Input: Topology G'.
1: **Procedure:** CREATE-FLOW(G')
2: $u \leftarrow$ all users in G'
3: **while** $u \neq \emptyset$ **do**
4: **for** all $n \in u$ **do**
5: $nh \leftarrow \{n' : n'$ offers shortest paths to $n\}$
6: **if** $C_n(n) = black$ or $C_f(n) = red$ **then**
7: $\forall_{n' \in nh} C_e(e_{nn'}) \leftarrow red$
8: $\forall_{n' \in nh} C_f(n') \leftarrow red$
9: **else if** $C_e(e_{nn'}) \neq red$ **then**
10: $\forall_{n' \in nh} C_e(e_{nn'}) \leftarrow green$
11: $\forall_{n' \in nh} C_f(n') \leftarrow green$
12: $u' \leftarrow u' \cup nh$
13: $u \leftarrow u'$

Fig. 3(a), link 6 is connected to d and e so that all the links which are connected to d $(3, 8)$ and e $(4, 7, 9)$ will be neighbor of 6 in G'. Therefore, node 6 is connected to $3, 8, 4, 7,$ and 9 in G'. Fig. 3(b) shows the line graph G'.

In Step 2, we create traffic flows from every user in G'. The flows travel through all the next hops remaining on the shortest paths to the victim. The flow continues and the traveled links are colored green. When a flow encounters a black node, the flow becomes red and continues coloring the traveled links red. If a link is already colored red, the color never changes, but if it is marked as green, then a red flow changes the link's color to red. The flow terminates at the victim. The process is shown in Alg. 2. In Fig. 3(c) the green flows travels nodes 11 and 9 which remain on the shortest paths from u_1 to v. The edges $u_1 \leftrightarrow 11$ and $u_1 \leftrightarrow 9$ become green. As 11 is a black node, the flow becomes red and the edge $11 \leftrightarrow 8$ becomes red. The process continues until every flow reaches v.

In Step 3, we create another graph $G'' = (V'', E'')$ with the victim node, red links, and nodes which are endpoints of the red links. Then, we eliminate the white nodes from G''. When we eliminate a white node, we connect its incoming neighbors (who are sending flows to the white node) with the outgoing neighbors (who are receiving flows from the white node). Alg. 3 shows this step in details. For example, when we eliminate node 1, we connect v and 4 by $4 \leftrightarrow v$. Similarly, we connect v and 7 by $7 \leftrightarrow v$ when we remove node 4. We create a virtual user u and edges from u to any black node that is connected to at least one green edge. Any black node that appears first on the shortest path from any user to the victim has a green incident edge. According to Fig. 3(c), nodes 11, 6, and 7 will be connected to the virtual source. Now G'' contains only the source, the victim, and black nodes. Graph G'' represents all the congested shortest path flow from every user to the victim. Fig. 3(e) shows G''.

In Step 4, a minimum number of nodes are selected to separate the victim and virtual user in G'' using Acid and Campos's minimum d-separating set [13]. When we are separating the victim and the virtual source, we are actually cutting all the shortest possible congested paths with a minimum number of link blockage. Applying Acid and Campos's method, we find the minimum separation set $\{3, 7\}$. Therefore, if we block the links 3 and 7 in G by sending filters to FR d and f, no legitimate traffic will follow any congested path. The complete procedure is shown in Alg. 1.

Algorithm 3 Elimination of white nodes from the flow graph

Input: Topology G'.
Output: The final graph G''.
1: **Procedure:** ELIMINATE(G')
2: $G'' \leftarrow G'$
3: **for** all $n \in V'$ **do**
4: **if** no incident edge to n is red **then**
5: Remove n from V''
6: $\forall_{j \in Neighbor(n)}$ remove e_{nj} from E''
7: **else if** $C_n(n) = white$ **then**
8: Remove n from V''.
9: Add edges from incoming to outgoing neighbors of n.
10: Create a virtual node u
11: **for** all nodes n in V'' **do**
12: **if** # of green incident edge to $n \geq 1$ **then**
13: Add edge e_{un} to E''

Theorem 1. *Alg. 1 finds an optimal number of links to block.*

Proof. Alg. 1 transforms the original topology to a flow graph where all the shortest congested flows are considered. According to the assumption, there is at least a non-congested shortest path. The flows travel one of the non-congested shortest paths after disabling all the possible congested paths. To prove the optimality of Alg. 1, we need to prove that G'' contains all the possible congested flows. Assume there is another red path from a black node n to v which does not use any link in the E''. Then, there might be another node n' which was not selected for forwarding flow during flow creation process in Step 5 of Alg. 2. Which means n' offers a path longer than the shortest path. FR n will never select the next hop offering a longer path when it already has a neighbor offering shortest path. Therefore, there is no other congested flows from a user which is not in G''. Cutting all the flows by removing some nodes means blocking the congested shortest paths in G. As the Acid and Campos's method provides the minimum size vertex cut set, the algorithm will return the optimal solution. □

Theorem 2. *The complexity of Alg. 1 is $O(m^3 d)$.*

Proof. Let us assume that G has n nodes, m edges, and d maximum node degree. Step 1 takes $O(md)$ time. If d is the maximum node degree in G, then G' has $2d - 1$ max node degree which is $O(d)$. Step 2 takes $O(md)$ to create flow for a user. In the worst case, the number of users can be m and Step 2 takes $O(m^2 d)$ time. To create G'' from G', $O(md)$ time is needed in the worst case. Therefore, the conversion from G to G'' takes $O(m^2 d)$. Step 4 takes $O(m^3 d)$ to find a minimum vertex separator. Therefore, the algorithm takes $O(m^3 d)$. □

TABLE I: Topology Parameters

	Topology I	Topology II	Topology III
Number of nodes	65	131	5200
Number of edges	123	310	10800
Internal user probability	0.2	0.5	0.5
Max node degree	10	13	1459

VI. Experimental Results

A. Experimental Settings

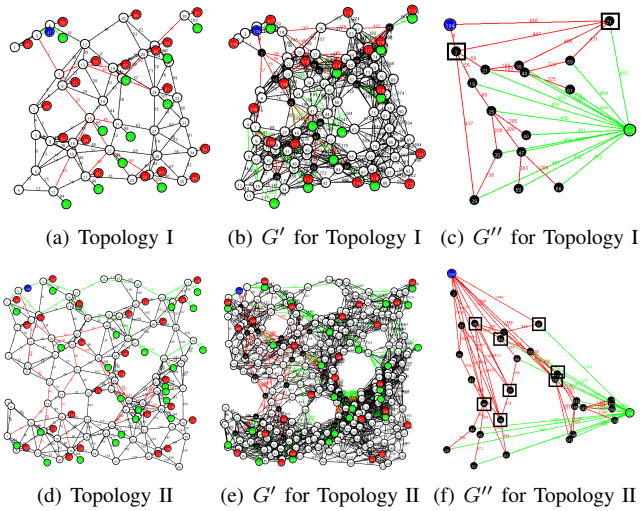
We conducted the experiments with a custom built Java simulator. We do not need to analyze transmission time or natural packet drop issues. We only need to count the number of legitimate packets, possible congested links, links to block, and hops to the victim. Using NS3 or other similar simulators for this kind of simulation would take several days to simulate $65-5200$ nodded topology. That is why we built our own Java multi-threaded simulator to get the results quickly.

We conducted simulations for randomly generated graph topologies and a subset of real network topology.The number of users, and the number of bots were selected randomly from a uniform distribution. The target links are selected based on the number of users using a link. If a topology has an internal user probability of 0.5, it means that 50% of the nodes are attached to attackers or to users. The Topology III was taken from the Autonomous systems AS-733 dataset [14]. The details are shown in Table I. Topologies I and II are depicted in Figs. 4(a) and 4(d). The green, red, blue, and white nodes represent the users, bots, victim, and FRs, respectively. The red links represent the possible congested links. The line graph G' with flows for Topologies I and II are shown in Figs. 4(b) and 4(e). The black nodes represent the congested links. The final flow graphs G'' are shown in Figs. 4(c) and 4(f). The black rectangles around a black node represent that the corresponding links of that node are selected for blocking.

B. Simulation Results

As the bots are distributed uniformly over the network, the attacker can congest any of the links in the network. We assume that the attacker is capable of congesting only one link. It selects the link which carries traffic from the highest number of users. Simulations shown in Figs. 5(a)-5(c) are conducted with this setting. Fig. 5(a) shows the regular number of received packets by the victim from the users in Topology II. There are 25 users and the number of received packets from all of them is above the threshold (150,000,000) for a particular period. When the attack is started, the numbers of received packets from some of the users are reduced. From Fig. 5(b), we can see that the number of received packets of 16 users fall bellow the threshold. When the filters are deployed, we can see that the number of received packets by all of the users is above the threshold (see Fig. 5(c)).

Figs. 5(d)-5(f) show the number of possible congested links (PCL), blocked links (BL), average hop counts before (HCB), and average hop counts after (HCA) by the number of attacked links. Here the attacker chooses the links that carry the top number of users' traffic. We observe that all the measurements



(a) Topology I (b) G' for Topology I (c) G'' for Topology I

(d) Topology II (e) G' for Topology II (f) G'' for Topology II

Fig. 4: Topologies I and II. (Green node=user, red node=bot, blue node=victim, red link=possible congested link)

increase with a higher number of attacked links. The HCA does not increase in Topology III, but it increases by 1 or 2 in Topology I. This is because Topology III has more redundant connections than Topology I.

Fig. 5(g) shows the number of affected users (AF) and total users (U). In Topology I, we see that after congestion, 10 of the 11 users become affected. For Topology I, 16 out of 25 users are affected. The ratio of the affected users in Topology III is less than in the other topologies. This is because Topology III has a higher node degree and many shortest paths. Therefore, the traffic from different users goes through many distinct paths. Fig. 5(h) shows the number of PCLs, BL, HCB, and HCA. We can observe that the BL is much smaller than the PCL. Especially, in Topology II, the number of PCLs is 40, but they can all be avoided by blocking only 2 links.

Fig. 5(i) shows the comparison between our proposed FR-based model and the BGP advertisement-based model. It is hard to compare these two models because the BGP advertisement-based model assumes that the deployer autonomous system knows exactly which links are congested. We assume that the victim (deployer) does not know exactly which links are congested, which is more practical. We assume that the victim in the FR-based model knows the congested links to give both models equal information. The BGP advertisement-based model will block all of the congested links (it avoids the congested links by false path advertisement) but our FR-based model blocks much fewer links than the BGP advertisement-based model. Fig. 5(i) shows the number of blocked links by the number of attacked links for both approaches. The attacked links are chosen randomly and an average of several rounds of simulation are shown in the figures.

We compare the FR-based model with some other approaches including FLoc [8], SPIFFY [5], ColDef [6], which use attack traffic detection at the router. We show the detrimental effects when the routers cannot detect the attack traffic. Fig. 5(j) shows the average number of received packets by the ratio of packet drop at congested links. The average number

of received packets is reduced by the drop rate at congested links. The number of received packets for 10 congested links is slightly less than that of one congested link. The numbers of received packets are similar regardless of the number of congested links or drop rate for the FR-based model. This is because when filters are deployed, none of the user traffic goes through any congested links.

From the above experimental results, we can conclude that the FR-based model can efficiently protect the users' traffic. The model works with limited knowledge about the network status and can protect its users with the minimum cost.

VII. Conclusion

The transit-link DDoS attack is the most powerful attack that makes a service unavailable to legitimate users. It is not possible to protect any server from DDoS attacks without the help of the network equipments. Current DDoS defense system cannot counter the transit-link DDoS attack because the attack packets do not go to the victim. Therefore, the victim remains unaware of the location of the attackers. Routers, the most important component in the network, can be upgraded to FRs easily. Besides, the FR can work in a network with legacy routers. An ISP can gain benefit economically by deploying FRs. Therefore, both the deployer and the victim benefit from this model. Simulation results show that our proposed approaches work much better than the existing approaches.

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] 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.

[2] D. Seo, H. Lee, and A. Perrig, "APFS: Adaptive Probabilistic Filter Scheduling Against Distributed Denial-of-service Attacks," *Comput. Secur.*, vol. 39, Nov 2013.

[3] K. Argyraki and D. R. Cheriton, "Loose Source Routing As a Mechanism for Traffic Policies," in *Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Aug 2004.

[4] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala, "Path Splicing," in *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4. ACM, Aug 2008.

[5] M. Suk Kang, V. D. Gligor, and V. Sekar, "SPIFFY: Inducing Cost-Detectability Tradeoffs for Persistent Link-Flooding Attacks," in *Network and Distributed System Security Symposium*, Jan 2016.

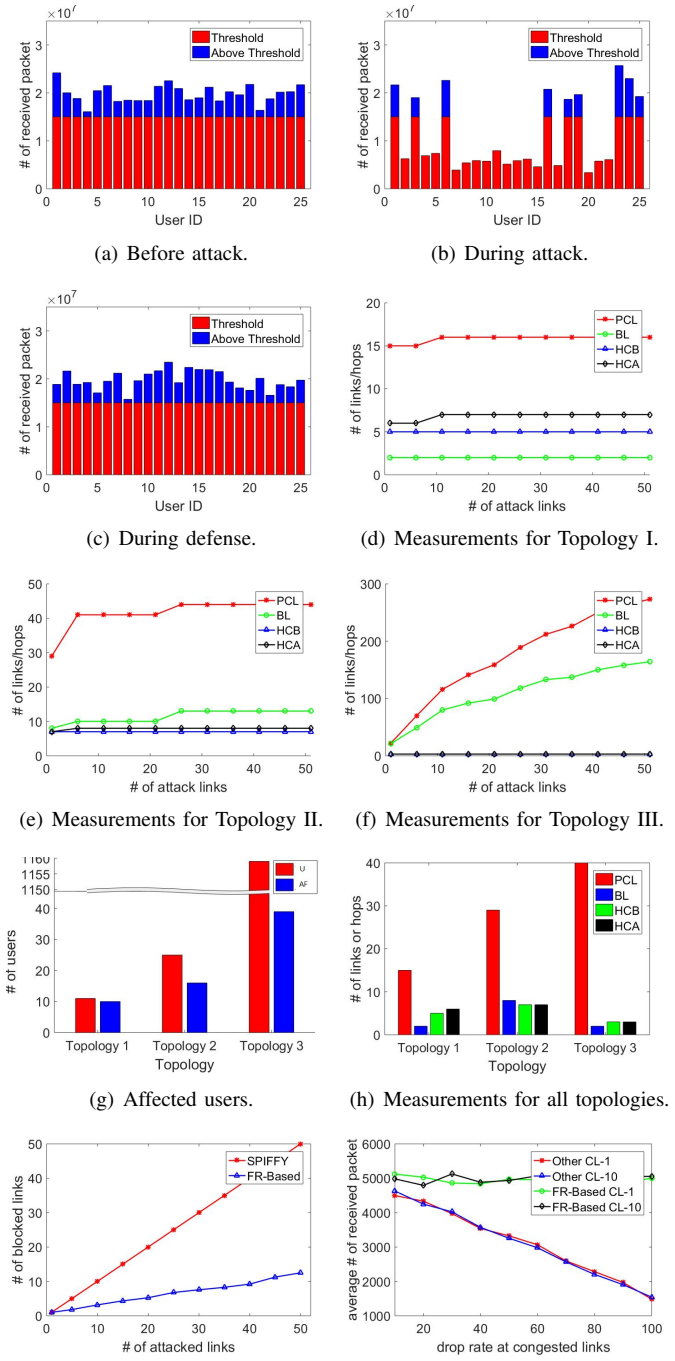
[6] S. B. Lee, M. S. Kang, and V. D. Gligor, "CoDef: Collaborative Defense Against Large-scale Link-flooding Attacks," in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, Dec 2013.

[7] J. Ioannidis and S. M. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," in *Network and Distributed System Security Symposium*, Mar 2002.

[8] S. B. Lee and V. D. Gligor, "FLoc : Dependable Link Access for Legitimate Traffic in Flooding Attacks," in *2010 IEEE 30th International Conference on Distributed Computing Systems*, Jun 2010.

[9] J. M. Smith and M. Schuchard, "Routing Around Congestion: Defeating DDoS Attacks and Adverse Network Conditions via Reactive BGP Routing," in *IEEE Symposium on Security and Privacy*, May 2018.

[10] 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.



(a) Before attack. (b) During attack. (c) During defense. (d) Measurements for Topology I. (e) Measurements for Topology II. (f) Measurements for Topology III. (g) Affected users. (h) Measurements for all topologies. (i) Comparison with BGP advertisement-based method. (j) Comparison with router level detection-based methods.

Fig. 5: Simulation results.

[11] ——, "APFS: Adaptive Probabilistic Filter Scheduling against distributed denial-of-service attacks," *Computers Security*, vol. 39, Nov 2013.

[12] 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 (ICPADS)*. IEEE, Dec 2018.

[13] S. Acid and L. M. De Campos, "An Algorithm for Finding Minimum D-separating Sets in Belief Networks," in *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., Aug 1996.

[14] "Autonomous systems as-733," https://snap.stanford.edu/data/as-733.html.