CoFence: A Collaborative DDoS Defence Using Network Function Virtualization

Bahman Rashidi Carol Fung
Department of Computer Science, Virginia Commonwealth University, Richmond, VA, USA
{rashidib, cfung}@vcu.edu

***Abstract*—With the exponential growth of the Internet use, the impact of cyber attacks are growing rapidly. Distributed Denial of Service (DDoS) attacks are the most common but damaging type of cyber attacks. Among them SYN Flood attack is the most common type. Existing DDoS defense strategies are encountering obstacles due to their high cost and low flexibility. The emerging of Network Function Virtualization (NFV) technology introduces new opportunities for low-cost and flexible DDoS defense solutions. In this work, we propose CoFence – a DDoS defense mechanism which facilitates a collaboration framework among NFV-based peer domain networks. CoFence allows domain networks help each others handle large volumes of DDoS attacks through resource sharing. Specifically, we focus on the resource allocation problem in the collaboration framework. Through CoFence a domain network decides the amount of resource to share with other peers based on a reciprocal-based utility function. Our simulation results demonstrate the designed resource allocation system is effective, incentive compatible, fair, and reciprocal.**

I. Introduction

Distributed Denial of Service (DDoS) attacks can cause severe damage to ISPs and online services, especially for small and medium-sized organizations who lack sufficient resources to withstand a high volume of DDoS traffic. In recent years, the proliferation of black market and the growth of "DDoS as a service" [4], [14] has greatly enhanced the capability of attacks since all attackers need to do is to visit the website, schedule, and pay for an attack. Some recent incidents show that DDoS attacks are becoming stronger and more frequent. For example, the Spamhaus attack in 2013 [3] has generated 300 Gbps attack traffic. This number has been increased to 600 Gbps in January 2016 [12].

Regarding the DDoS attack techniques, there are two major types of attack traffic: IP spoofing attacks and real source IP-based attacks. The real source IP-based DDoS attacks commonly utilizes compromised nodes in the Internet, commonly called bots or zombies, to launch the attack. On the other hand, IP spoofing DDoS is the type of attacks where the source addresses are not the real IP address of the attacker. An example of this type of attack is SYN Floods [15]. An Atlas security report shows that the SYN Floods take the vast majority of the attack volume in recent major DDoS attacks [2]. Existing solutions on SYN Floods, including dedicated DDoS mitigation devices (e.g., IPS or firewall) and third-party DDoS filtering cloud services [5], [1], either bring high cost through purchasing dedicated hardware or trigger privacy concerns by directing traffic to untrusted third parties. In this paper, we introduce a novel approach for DDoS mitigation using collaborative networks and Network Function Virtualization (NFV) technology.

NFV is an emerging technology where network functions are implemented and provided in software, which runs on the commodity hardware [7]. The network functions are implemented as software and deployed as virtual machines. The virtual machines run on general purpose commodity hardware systems so that NFV not only provides the benefit of elasticity, but also reduces the cost by running on commodity platforms like x86- or ARM-based servers instead of specialized hardware, resulting in a much easier deployment and lower cost. At the same time, NFV also introduces new opportunities for DDoS detection and mitigation.

Traditional device-based DDoS mitigation is limited by the computation capacity of the dedicated network functions, such as firewall or IPS. Upgrading or adding new hardware introduces high cost and long cycle time. The usage of NFV technology makes device upgrading and creation fast and low cost, which brings a great opportunity for DDoS defense. In our previous work [11], we introduced a dynamic local networking system based on NFV technology which utilizes virtualized network functions running on commodity servers to perform DDoS data filtering. However, this solution may not be sufficient when the attack strength exceeds the available hardware capacity. Seeking external helping resource may be a viable solution.

In this work, we propose *CoFence*, a collaborative DDoS mitigation network system which facilitates a domain-helps-domain collaboration network. In this network, a domain can direct excessive traffic to other trusted external domains for DDoS filtering. The filtered clean traffic will be forwarded back to the targeted domain. Specifically, we focus on the resource allocation problem when multiple requesters ask for help. We design a fair and incentive-compatible resource allocation method which provides an effective collaborative DDoS defense with inherent reciprocal eco-system. Our experimental results demonstrate that our proposed solution can effectively reduce the DDoS attack flow to the targeted server, and the resource allocation is fare and provides incentive for domains to maximally help other domains in need. The contributions of this paper include: 1) Our work proposes a novel collaborative DDoS defense network based on network function virtualization technology. 2) We propose a dynamic resource allocation mechanism for domains so that the system

©IFIP, (2016). This is the author's version of the work. It is posted here by permission of IFIP for your personal use. Not for redistribution. The definitive version was published in IFIP CNSM 2016

is fair, efficient, and incentive-compatible. 3) We evaluate our proposed solutions using simulation and verify that the proposed solution is effective to DDoS defense and meets our design goals.

II. Related Work

Network softwarization technology including SDN and NFV has been growing rapidly due to high interest from both academia and industry. In SDN the networking functions are separated into control plane and data plane. The main idea of NFV is to replace dedicated network appliances, such as hardware-based routers and firewalls, with software that runs on commercial off-the-shelf servers. Both SDN and NFV can lower the cost of network deployment and high flexibility for management compared to traditional computer networks.

There are many work proposed to utilize NFV for DDoS defense. Fayaz et al. propose Bohatei [8], a DDoS defense solution using software defined networking (SDN) and network functions virtualization (NFV) paradigms. Bohatei is designed to cope with the expensiveness and proprietary hardware appliances of the existing solutions. Bohatei uses its resource manager component to assign available network resources to the defense VMs once suspicious traffic is detected. Since Bohatei is using a limited amount of network resources (to launches VMs), it cannot be effective in case of attacks that incoming traffic's strength is heavier than what it can handle. In contrast, in CoFence, in addition to utilizing NFV technology, attack victims can receive external help from their collaborators in terms of network resources.

Lim et al. propose a SDN-based approach to overcome legitimate looking DDoS attacks [13]. In their work they investigate a DDoS blocking application that runs over the SDN controller while using the standard OpenFlow interface. The DDoS blocking application runs on a SDN controller. The scheme requires a large amount of communication between the DDoS blocking application running on the SDN controller and the server to be protected. The blocking application needs to cooperate with SDN controller which increases the dependency and a high latency consecutively.

Guenane et al. have designed a architecture of cloud-based firewalling service using NFV technology and other network Virtualization capabilities to overcome DDoS attacks [10]. The designed cloud architecture acts as an intermediary to filter the attack traffic for the customer and transmits only legitimate traffic.

Our previous work [9] proposes a traffic prioritization algorithm to reduce the impact from DDoS attacks based on real IPs. VFence [11] on the other side, is specially effective to reduce the impact from DDoS attacks on IP-spoofing by creating additional virtual IPSs in NFV-enabled networks. IPSs verify the source IP of the incoming traffic and verified traffic will be further forwarded to the server. However, it is not effective when the attack volume is high enough to overwhelm the available location resource. CoFence can solve this problem by utilizing external resources in the collaboration network. Hence, it further improves the DDoS resistance level of NFV-enabled domain networks.

III. Collaborative DDoS defense

In this section we present the network design of CoFence. The purpose of CoFence is to provide a platform for domain networks (e.g., an enterprise network or an ISP) to help each others to enhance resistance against large-scale DDoS attacks. With the help of network function virtualization (NFV) technology, each NFV-enabled domain network can contribute their spare network resource to help other domains in the network when needed.

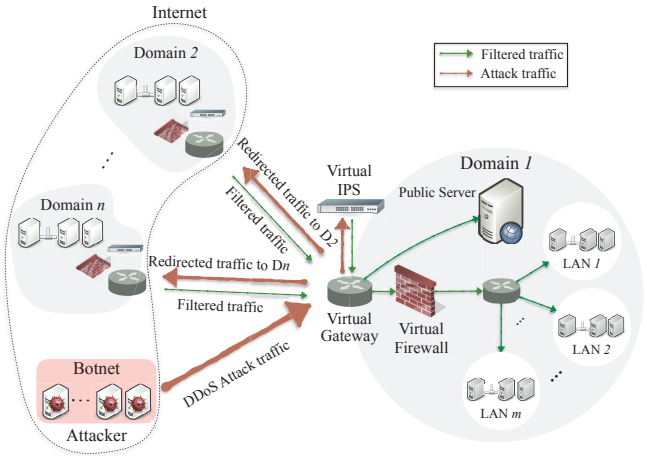

Fig. 1. A case study of collaborative DDoS defense

In a CoFence network, each NFV-enabled domain contains a virtual gateway and a virtual IPS. The purpose of the virtual IPS is to detect and filter DDoS attacks. Due to the flexibility of NFV, a virtual IPS can be created and its capacity can be configured dynamically based on need. When a domains joins CoFence, the domain can choose whether to share its IPS with other trusted domains or not, and configure the maximum external traffic it is willing to handle for other domains. In a CoFence network trust is already established. Trust can be addressed through a service agreement process in which domains agree with a number of predefined terms and conditions.

Figure 1 illustrates a case study of CoFence. When the attacker launches a DDoS attack against the public server in domain 1 and the attack traffic volume exceed the maximum capacity of the local IPS, some incoming traffic can be redirected to its collaborator domains for filtering (this can be done by updating the forwarding table in the gateway). The SYN flood will be filtered remotely and only filtered traffic is forwarded back to domain 1.

To be able to collaborate with other domains, a virtual IPS should contain the following functions:

1) Communication Component: This is used to communicated with other domains in the network. The communication in the collaboration network can be divided into three types: (a) request for help and offer to help. (b) request to add as

neighbors and respond to the neighbor adding request. (c) request to remove as neighbors and respond to the neighbor removal request.

CoFence is set up as a "*separat*" defense network and usual traffic is transferred within another network. This way we can ensure that requesting help is possible in case the network link is saturated.

2) Resource Allocation: After receiving a helping request, a domain needs to decide how much of its spare resource it should offer to the requesting neighbor. This decision problem becomes non-trivial when there are multiple requesters at the same time. A resource allocation component should be in place to compute the optimal way for the resource allocation decision. Several design goals include how to make the resource usage efficient, fair to new neighbors, and incentive-compatible to encourage more generous sharing. The focus of this paper is to design a resource allocation mechanism to meet the above goals. It is worth talking into consideration that CoFence is a distributed model and it can be applied to networks with different scales.

IV. Resource Allocation Mechanism

In this section, we describe the resource allocation model for nodes in the CoFence network. We start from the statement of the resource allocation design goals and then a modeling of the resource allocation to fulfill the goals.

A. The Design Goals of Resource Allocation

Our design goals can be stated as follows. First of all we are interested to build a collaboration system which is fair, incentive-compatible, and reciprocal. The fairness property means the system can control the discrepancy of received help among different nodes so starving can be avoided for new participants. For example, a new participant with no credit shall receive help when resource allows. The incentive-compatibility provides incentive mechanism for participants to contribute more to help others. i.e., the more a node contributes resource to help others, the more help it receives in return when needed. The reciprocal property provides a pair-wise mutual beneficial relation. For example, the more node x helps node y, the more node y helps x in return. In terms of performance we aim at a system to be efficient, with low communication overhead, and effective to defend against DDoS attacks. In the next subsection, we discuss the mechanism design of the resource allocation mechanism to fulfill the above design goals.

B. Resource Allocation Modeling

The CoFence network consists of a set of n domains, and the domains are connected into a collaboration network denoted by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ denotes the set of domains (nodes) and $\mathcal{E}$ represents the connection between pair of nodes if they have established a trusted collaboration relationship. In the network, domains (nodes) share attack traffic with another domain (nodes) under DDoS attack if they have an edge in between them. We use $\mathcal{N}_i$ to denote the set of *neighbors* *node* of i. i.e., they are connected to node i directly by an edge in $\mathcal{G}$. When a node i receives a helping requests from its neighbor j, helping resource will be allocated using a fair resource allocation algorithm described in the next subsection.

TABLE I
Summary of Notations.

Notation	Meaning
$\mathcal{N}$	Set of all participating domain nodes in CoFence
$\mathcal{N}_i$	Set of all neighbors for domain i
$\mathcal{S}_i$	Set of neighbor nodes requesting help from domain i
$\vec{r}_i$	The helping data rate from domain i to neighbors
r_{ij}	The helping data rate from domain i to node j
R_{ij}	The requested helping rate from i to j (set by node j)
$S_{ij}(r_{ij})$	The satisfaction level of j given helping rate r_{ij}
$\hat{r}_i$	Total reserved helping resource amount of node i
H_{ij}	Level of helpfulness from i to j in the past
λ	Forgetting factor
Δt	Helping credit updating interval

Let set $\mathcal{S}_i$ ($\mathcal{S}_i \in \mathcal{N}_i$) denote the set of domains which request for help from domain i. We use a vector $\vec{r}_i$ to represent the traffic handling volume that domain i offers to help the requesters and use R_{ij} to denote the traffic handling volume that node j requests from node i when it is under attack, where $i \in \mathcal{N}$ and $j \in \mathcal{S}_i$. Note that R_{ij} is controlled by node j and informed to node i. We use $\vec{R}_j$ to denote the requested handling rates node j imposes to all its neighbors. Our system requires that each node controls its helping rate under the requested helping rate, i.e., $r_{ij} \leq R_{ij}$. Also the total helping rate should not exceed the spare resource that the offering node is willing to share, i.e., $\sum_{j \in \mathcal{S}_i} r_{ij} \leq \hat{r}_i$, where $\hat{r}_i$ is the maximum resource amount that node i is willing to share with other. We list all notations in Table. I.

For the design of a reciprocal system, we use matrix $\mathbf{H} = [H_{ij}]_{i,j \in \mathcal{N}}$ to denote the *helping credit* of nodes, where $H_{ij} \geq 0$ represents the level of helpfulness from node i perceived by node j. Note that the matrix $\mathbf{H}$ can be asymmetric, i.e., $H_{ij} \neq H_{ji}$. Our goal is to devise a resource allocation protocol such that the helping resource is fairly distributed to others based on their helpfulness in the past. To achieve this goal, each node i solves an optimization problem that maximizes the aggregated satisfaction level of its requesting neighbors as follows:

$$\begin{aligned} \underset{\vec{r}_i}{\operatorname{argmax}} U_i^h(\vec{r}_i) \quad &:= \quad \sum_{j \in \mathcal{S}_i} H_{ji} S_{ij}(r_{ij}) && (1) \\ \sum_{j \in \mathcal{S}_i} r_{ij} \quad &\leq \quad \hat{r}_i, && (2) \\ 0 \leq r_{ij} \quad &\leq \quad R_{ij}, && (3) \end{aligned}$$

where $S_{ij}(r_{ij}) \in [0, 1]$ is the satisfaction level of the requesting node j in response to the helping rate r_{ij} from the node i. We let S_{ij} take the following form

$$S_{ij}(r_{ij}) := \log_2 \left(1 + \frac{r_{ij}}{R_{ij}}\right). \quad (4)$$

The concavity and monotonicity of the satisfaction level indicate that a requesting node becomes increasingly pleased when more help is received but the marginal satisfaction decreases as the amount of help increases. The parameter H_{ji} in (1) suggests that the satisfaction level of a node j carries more weight when it is a more helpful node to i in the past.

The utility U_i^h measures the aggregated satisfaction level experienced by node i's collaborators weighted by their helpfulness in the past. It allows a node to provide more help to those with whom there was more helpful in the past.

It can be seen that when $\hat{r}_i$ is sufficiently large then (2) is an inactive constraint, the solution to (1) becomes trivial and $r_{ij} = R_{ij}$ for all $j \in \mathcal{S}_i$. The situation becomes more interesting when (2) is an active constraint. Assuming that R_{ij} has been appropriately set by node j, we form a Lagrangian function $\mathcal{L}^i : \mathbb{R}^{n_i} \times \mathbb{R} \times \mathbb{R}^{n_i} \to \mathbb{R}$

$$\mathcal{L}^i(\vec{r}_i, \mu_i, \delta_{ij}) := \sum_{j \in \mathcal{S}_i} H_{ji} \log_2 \left(1 + \frac{r_{ij}}{R_{ij}}\right) - \mu_i \left(\sum_{j \in \mathcal{S}_i} r_{ij} - \hat{r}_i\right) - \sum_{j \in \mathcal{S}_i} \delta_{ij}(r_{ij} - R_{ij}), \quad (5)$$

where $\mu_i, \delta_{ij} \in \mathbb{R}_+$ satisfy the complementarity conditions $\mu_i \left(\sum_{j \in \mathcal{S}_i} r_{ij} - \hat{r}_i\right) = 0$, and $\delta_{ij}(r_{ij} - R_{ij}) = 0, \forall j \in \mathcal{S}_i$. We minimize the Lagrangian with respect to $\vec{r}_i \in \mathbb{R}_+^{|\mathcal{S}_i|}$ and obtain the first-order *Kuhn-Tucker condition*:

$$\frac{H_{ji}}{r_{ij} + R_{ij}} = \mu_i + \delta_{ij}, \quad \forall j \in \mathcal{S}_i.$$

When (2) is active but (3) is inactive, we can find an open form solution supplied with the equality condition

$$\sum_{j \in \mathcal{S}_i} r_{ij} = \hat{r}_i, \quad (6)$$

and consequently, we obtain the optimal solution

$$r_{ij}^\star := \frac{H_{ji}}{\sum_{u \in \mathcal{S}_i} H_{ui}} \left(\hat{r}_i + \sum_{v \in \mathcal{S}_i} R_{iv}\right) - R_{ij}. \quad (7)$$

When the constraint (3) is active, the optimal solution is attained at the boundary. Since the $\log$ function has the fairness property, the optimal solution $r_{ij}^\star$ has non-zero entries when the resource budget $\hat{r}_i > 0$. In addition, due to the monotonicity of the objective function, the optimal solution $r_{ij}^\star$ is attained when all resource budgets are allocated, i.e., constraint (2) is active.

Remark 1: We can interpret (7) as follows. The solution $r_{ij}^\star$ is composed of two components. The first part is a proportional division of the resource capacity $\hat{r}_i$ among $|\mathcal{N}_i|$ collaborators according to their compatibilities. The second part is a linear correction on the proportional division by balancing the requested sending rate R_{ij}. It is also important to notice that by differentiating $r_{ij}^\star$ with respect to R_{ij}, we obtain $\frac{\partial r_{ij}^\star}{\partial R_{ij}} = (\frac{H_{ji}}{\sum_{u \in \mathcal{S}_i} H_{ui}} - 1) < 0$, suggesting that at the optimal solution, the helping rate decreases as the recipient sets a higher requesting rate. If a node wishes to receive higher helping rate from its neighbors, *it has no incentive to overstate its level of request*. Rather, a node j has the incentive to understate its request level to increase $r_{ij}^\star$. However, the optimal solution is upper bounded by R_{ij}. Hence, by understating its request R_{ij}, the optimal helping rate is achieved at R_{ij}.

C. Helping Credit Computation

To build an incentive-compatible, and reciprocal resource allocation system, it is important for a node to track how much a neighbor node have helped in the past. We call it *helping credit*. In our model, we use the cumulative helping resource a node have offered in the past to represent the helping credit. Each node tracks the helping credit from its neighbors so that the helping resource tracking is fully distributed and the measured helping credit is private to each node.

Let $r_{ij}(t)$ denote the helping data rate that node i offers to node j at time t, then node j computes the helping credit of node i at time t_0 using the following equation:

$$H_{ij}(t_0) = \int_{-\infty}^{t_0} r_{ij}(t) \lambda^{(t_0 - t)} dt \quad (8)$$

A node gains credit by providing help to other nodes in the network. The credit in the past is being "forgotten" with an exponential speed λ, where $\lambda \in (0, 1]$ is called a *forgetting factor*. A smaller λ represents faster forgotten speed. If $\lambda = 1$ then all past credit will be remembered and carry the same weight as new credit. Note that when a new node joins the network, its credit can start from a small value to all its neighbors. Given the credit of node i perceived by j at a historical time $t' = t_0 - \Delta t$ is known to be $H_{ij}(t')$, then we can compute the credit at time t_0 using the following equation:

$$\begin{aligned} H_{ij}(t_0) &= \int_{-\infty}^{t'} r_{ij}(t) \lambda^{(t_0 - t)} dt + \int_{t'}^{t_0} r_{ij}(t) \lambda^{(t_0 - t)} dt \quad (9) \\ &= \int_{-\infty}^{t'} r_{ij}(t) \lambda^{(t' - t)} \lambda^{(t_0 - t')} dt + \int_{t'}^{t_0} r_{ij}(t) \lambda^{(t_0 - t)} dt \\ &= \lambda^{\Delta t} H_{ij}(t - \Delta t) + \int_{t_0 - \Delta t}^{t_0} r_{ij}(t) \lambda^{(t_0 - t)} dt \end{aligned}$$

Equation (10) indicates that the helping credit of a node at any time (e.g., t_0) can be computed incrementally based on the credit at an earlier time (e.g., t'). Therefore the credit computation requires fixed memory and storage.

Alternatively, let the time that a new node i joins the network to be 0 with initial credit c_i, and the current time is a relative time T after the initial joining time, then the helping credit perceived by j can be computed as follows:

$$H_{ij}(T) = \lambda^T c_i + \int_0^T r_{ij}(t) \lambda^{(T - t)} dt \quad (10)$$

D. Requesting for Help

As stated in Remark 1, a node under attack does not have incentive to overstate its level of request. To effectively inform the desired help to its collaborators, a node under attack can broadcast the helping requests use the following algorithm.

As described in Algorithm 1, when a domain u detects DDoS attacks, it sends requests for help to its trusted neighbors sequentially, starting from the neighbor with the highest helping credit. When sufficient helping resource is achieved, it exits the rest of the requesting process. After negotiation, domain u directs the traffic flows to the helpers. At last, each domain updates the credit of its neighbors to reflect the up-to-date status after interval Δt.

Algorithm 1 Seek Help by Node u

//This algorithm describes the algorithm for a node to broadcast its requested help to its neighbors. It is triggered when DDoS attacks are detected.
Inputs :
$\mathcal{N}_u$: the set of neighbor domains that are trusted by domain u
$\vec{H}_u$: helping credits for all neighbors of domain u in descending order
A_u: required helping resource needed for domain u during DDoS attack
Δt: the time interval to recompute the credit of all neighbors
//Send request for help to each trusted neighbor ordered by their level of helpfulness in the past.
for each node v in $\vec{H}_u$ **do**
    $h \Leftarrow$sendRequest(v,A_u) //v computes resource offer using Eq. (1)
    $A_u = A_u - h$ //Reduce the required amounts after receiving help
    if $A_u = 0$ **then**
        break the for loop
    end if
end for
RedirectTraffic to helpers
set timer (Δt, "**UpdateCredit**($\mathcal{N}_u$)") //Update neighbors' helping credits periodically (every Δt)

V. Evaluation

In this section we present our experiments evaluating the proposed collaborative model. We conducted a series of experiments on different case studies to evaluate the performance of the model.

A. Simulation Setup

Since the vast majority of DDoS attacks is *SYN Flood* attack, we simulate *SYN Flood* attack in our experiment. We use *Discrete Event Simulation* (DTS) to build the environment. A DTS framework models the operation of a network system through processing a sequence of discrete events ordered by time. More specifically we used *SimPy* [6] framework as our simulator. SimPy is a process-based discrete-event simulation framework based on standard Python. The packets arrival are simulated using *Poisson process*.

We simulate a collaboration network with domains (nodes) sharing their virtual IPS DDoS data filtering capability. We defined two types of network traffic in the simulation: *legit traffic*, which is the normal traffic that a node receives during the normal situation; and *attack traffic*, which is the packets flow that a domain receives when it is under attack. We set the packet arrival rate parameter λ to 1K packets/second for legit traffic and 6K packets/second for the attack traffic. In addition, We set the maximum packet processing rate for each virtual IPS to be 2K packets/second by default, and the buffer size to be 10.

B. The Computation of Helping Credits

First of all we evaluate the helping credit computation that one domain gains based on its helping effort to another domain. Equation 8 is used to compute the credits. We set up two nodes i and j. Node i is under DDoS attack and node j provides help to node i. Node j provides help to handle traffic with data rate r_{ji}. At first we fix r_{ji} to 1K/second and observe the change of credit of node j perceived by node i under different setting of parameter λ. Figure 2(a) shows the the helping credit of node j computed by node i. As we can see, under all λ settings the the helping credits increase with time. A higher λ leads to a higher credit value.

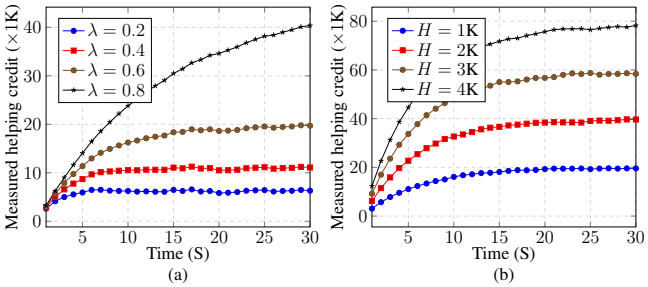

Fig. 2. Helping credit measurement: (a) the helping credit of node j perceived by node i for a fix helping rate $r_{ji} = 1K$/second at different λ settings; (b) the helping credit of node j perceived by node i for a fixed value of $\lambda = 0.6$ and different settings of help rate from node j.

In the second part of the experiment, we fixed the value of λ to 0.6 and let r_{ji} increases from 1K to 4K with step 1K. Figure 2(b) shows that the more generous that node j helps node i, the higher credit it gains.

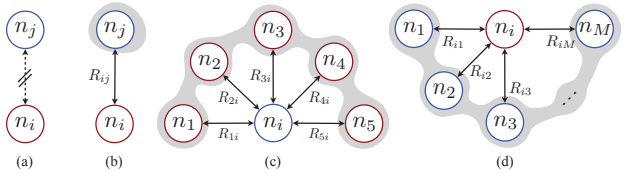

Fig. 3. Case studies: (a) attack target n_i is not equipped with CoFence; (b) n_i is using CoFence and has one neighbor; (c) attack targets $n_1 - n_5$ share one neighbor node (n_i); (d) node n_i has different number of neighbor nodes in the network

C. Case Studies

In the next set of experiments we use four case study, as shown in Figure 3, to evaluate the efficiency of our proposed CoFence model against DDoS attacks. Figure 4 (a) shows the traffic trace we use in these case studies. We can see that the normal traffic to both nodes are set to be $1K$/second while the attack traffic to node n_i is $6K$/second. For the proof of the concept, we only simulate a short period of DDoS attack flow from time 10s to 20s.

1) Case Study 1: In the first experiment we measure the packets dropping rate in the network when CoFence is not in place. The corresponding case study is Figure 3 (a). In this scenario node i is under attack. Since node i can only handle data rate $2K$/second, much traffic to node i has to drop. Figure 4 (b) shows the packet dropping rate at both node i and node j. We can see that the drop rate on node i increases significantly under attack. In contrast, node j handles its incoming network traffic which is half of its capacity without with minimum packets drops.

2) Case Study 2: In the second experiment we evaluate the efficiency of CoFence when the attacked target (node i) has one neighbor (node j) in CoFence. The case study is illustrated in Figure 3 (b). In this case, when node i is under attack, node j can offer its spare resource to process part of the excessive data flow from node i so that it can reduce the dropping rate on node i. Figure 5 (a) illustrates the packet dropping rate

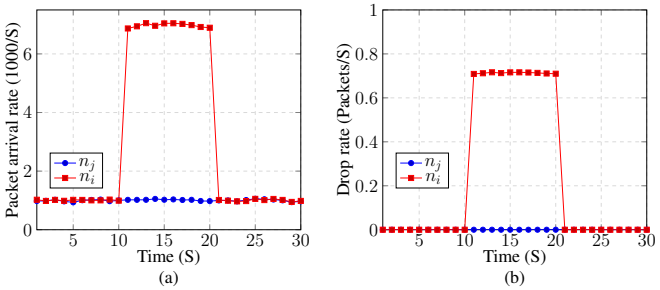

Fig. 4. Incoming packet and drop rate for case study (a).

on both nodes. We can see that the drop rate on node i is reduced with the help of node j. In this case, node j offered $1K$/second processing power to node i. Note that in this case node j's packet incoming rate reaches its processing capacity, a small portion of the packets are dropped.

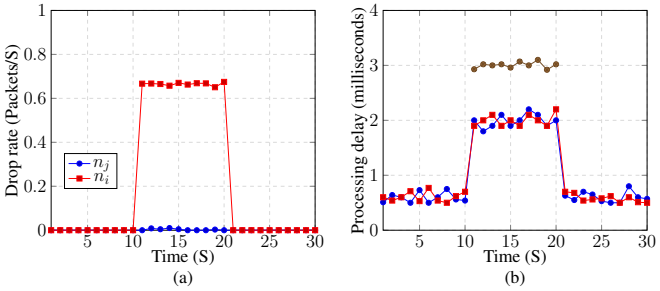

Fig. 5. Average dropping rate and processing delay for case study (b).

We also study the average *processing delay* for legit packets arriving at both node i and j. We define the *processing delay* to be the time elapsed from a legit packet's arrival at a gateway to its arrival at the online server. Figure 5 (b) shows the average processing delay for packets arriving at both nodes i and j. Since every node processing rate is 2K packets per second, the average process time (in a second) per packet is $0.5ms$ or slightly higher than the nodes' processing time when no node is under attack. When attack happens, a packets' processing delay at node i increases to $1.8ms - 2.3ms$. At node j we have two types of packets to be processed: node's j's regular incoming packets and packets coming from node i. The delay for incoming regular packets follows the same delay as node i, but the computed delay for received packets from node i includes the redirection time, which includes the transmission time and propagation time between two domains. For the simplicity we assume that the average redirecting packets from one domain to another takes $1ms$. Therefore, the delay for this packets is $1ms$ higher.

3) Case Study 3: In the third experiment, we evaluate the case that 5 nodes ($n_1 \sim n_5$) are under attack and all of them share one helper node (n_i) (Figure 3(c)). We stress all nodes with DDoS traffic $6K$/s on each node. Under this case all attacked nodes turned to node i for help. We let the credits for nodes $1--5$ range from 0 to 80 with step 20. Each round we let the node's i maximal shared capacity be 1K, 10K, 20K, and 25K. Figure 6 shows the amount of received help from node i. We observe that when node i has more free capacity, attacked nodes receive more help. The higher a node's credit is, the more resource it received from node i. When the available free capacity is set to 25K, all the nodes including the node with 0 credit will receive all they request. This demonstrate the efficiency and fairness of the resource allocation. i.e., all available resource will be utilized and no node should starve if resource is available.

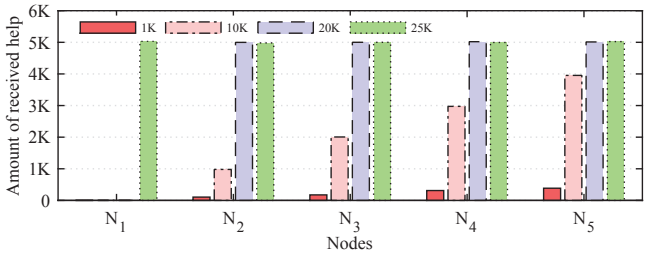

Fig. 6. Amount of help that nodes $1-5$ receive for different available capacities of node i (case study (c)).

4) Case Study 4: In the fourth experiment we evaluate the impact of the number of neighbors (helper nodes) for the attacked node. Figure 3 (d) illustrates the case study for this experiment. In the network node i has m neighbor nodes in the collaboration network. In our experiment we start from $m = 1$ and increase m by one in each round. We measure the packet dropping rate of node n_i is under DDoS attack. Figure 7 shows that after adding the five helper nodes, the drop rate of node i reduces to 0. This implies that more neighbors a domain has, the more DDoS attack resistant it is.

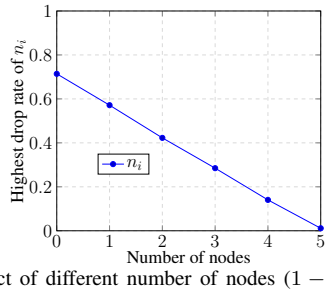

Fig. 7. The impact of different number of nodes ($1-6$) in a attack target node's locality on packet drop rate (case study (d)).

VI. Conclusion

In this paper we propose CoFence, a collaborative network to defend against DDoS attacks based on network virtualization technology, where domain networks under DDoS attack can redirect excessive traffic to other collaborating domains for filtering. Specifically we focus on the resource allocation mechanism that determines how much resource one domain should offer to the requestors so that the resource is distributed efficiently, fairly, and with incentives. Our evaluation results demonstrate that the collaborative DDoS defense can effectively reduce the impact from the attack and the proposed resource allocation mechanism can meet the design goal. In order to make our credit evaluation process more fair and effective we will include the impact of link bandwidth into our credit evaluation process.

REFERENCES

[1] Arbor ddos detection and protection. http://security.arbornetworks.com/protection/?gclid=CNrToYHYqM0CFRY7gQodjvcN_w.

[2] Atlas q2 2015 update. http://www.slideshare.net/Arbor_Networks/atlas-q2-2015final.

[3] Biggest internet attack in history threatens critical systems. http://www.ibtimes.co.uk/biggest-internet-attack-history-threatens-critical-infrastructure-450969.

[4] The booter website. https://booter.xyz/.

[5] Prolexic routed. https://www.akamai.com/us/en/solutions/products/cloud-security/prolexic-routed.jsp.

[6] The simpy simulator. https://simpy.readthedocs.io.

[7] ETSI NFV ISG. Network Functions Virtualization White Paper 3: Network Operator Perspectives on Industry Progress. In *SDN and OpenFlow World Congress*, Oct. 2014.

[8] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey. Bohatei: Flexible and elastic ddos defense. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 817–832, Washington, D.C., 2015. USENIX Association.

[9] C. J. Fung and B. McCormick. Vguard: A distributed denial of service attack mitigation method using network function virtualization. In *Network and Service Management (CNSM), 2015 11th International Conference on*, pages 64–70. IEEE, 2015.

[10] F. Guenane, M. Nogueira, and A. Serhrouchni. Ddos mitigation cloud-based service. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 1, pages 1363–1368, Aug 2015.

[11] A. H. M. Jakaria, W. Yang, B. Rashidi, C. Fung, and M. A. Rahman. Vfence: A defense against distributed denial of service attacks using network function virtualization. In *11th IEEE International Workshop on Security, Trust, and Privacy for Software Applications(STPSA 2016)*. IEEE, 2016.

[12] S. Khandelwal. 602 gbps! this may have been the largest ddos attack in history. http://thehackernews.com/2016/01/biggest-ddos-attack.html.

[13] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang. A sdn-oriented ddos blocking scheme for botnet-based attacks. In *2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 63–68, July 2014.

[14] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Zambenedetti Granville, and A. Pras. Booters - an analysis of ddos-as-a-service attacks. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 243–251. IEEE, 2015.

[15] H. Wang, D. Zhang, and K. G. Shin. Detecting syn flooding attacks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1530–1539. IEEE, 2002.