A Collaborative DDoS Defence Framework Using Network Function Virtualization

Bahman Rashidi, *Member, IEEE*, Carol Fung, *Member, IEEE*, and Elisa Bertino, *Fellow, IEEE*

***Abstract*— High-profile and often destructive distributed denial of service (DDoS) attacks continue to be one of the top security concerns as the DDoS attacks volumes are increasing constantly. Among them, the SYN Flood attack is the most common type. Conventional DDoS defense solutions may not be preferable, since they demand highly capable hardware resources, which induce high cost and long deployment cycle. The emerging of network function virtualization (NFV) technology introduces new opportunities to decrease the amount of proprietary hardware that is needed to launch and operate network services. In this paper, we propose a DDoS defense mechanism named CoFence, which facilitates a "domain-helps-domain" collaboration network among NFV-based domain networks. CoFence allows domain networks to help each other in handling large volume of DDoS attacks through resource sharing. Specifically, we design a dynamic resource allocation mechanism for domains so that the resource allocation is fair, efficient, and incentive-compatible. The resource sharing mechanism is modeled as a multi-leader-follower *Stackelberg* game. In this game, all domains have a degree of control to maximize their own utility. The resource supplier domains determine the amount of resource to each requesting peer based on optimizing a reciprocal-based utility function. On the other hand, the resource requesting domains decide the level of demand to send to the resource supplier domains in order to reach sufficient support. Our simulation results demonstrate that the designed resource allocation game is effective, incentive compatible, fair, and reciprocal under its Nash equilibrium.**

***Index Terms*— Software defined networking, network function virtualization, DDoS, collaborative network.**

I. Introduction

IN RECENT years, Denial of Service (DDoS) attacks have evolved to be increasingly powerful and destructive, which can cause severe damage to internet service providers (ISPs) and online services for resource-limited small and medium-sized organizations. The growth of "DDoS as a service" [6], [26] has further made attacks much easier to be launched since all that attackers need to do is to visit the website, customize, 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 [20].

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 utilize compromised nodes in the Internet, called bots or zombies, to launch an attack. On the other hand, IP spoofing DDoS is an attack in which the source IP addresses are fabricated (not the real IP address of the attacker). An example of this type of attack is SYN Floods attacks [30]. A recent Atlas security report shows that the SYN Floods take the vast majority of the attack volume in major DDoS attacks [2]. Existing solutions to protect from SYN Floods, including dedicated DDoS mitigation devices (e.g., Intrusion Prevention System (IPS) or firewall) and third-party DDoS filtering cloud services [1], [4], either have cost because of the need of 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 commodity hardware [11]. The network functions are implemented as software and deployed as virtual machines. As the virtual machines run on general purpose commodity hardware, 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 is costly and has a long cycle time. The usage of NFV technology makes device upgrading and creation fast and low cost, which results in a major opportunity for effective and inexpensive DDoS defenses. In our previous work [18], 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 resources 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

Manuscript received December 9, 2016; revised April 7, 2017; accepted May 21, 2017. Date of publication May 26, 2017; date of current version July 20, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Liang Xiao. *(Corresponding author: Bahman Rashidi.)*

B. Rashidi and C. Fung are with the Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284 USA (e-mail: rashidib@vcu.edu; cfung@vcu.edu).

E. Bertino is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907 USA (e-mail: bertino@cs.purdue.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2017.2708693

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 ecosystem. The resource allocation scenario is further modeled into a multi-leader-follower *Stackelberg* game by formalizing a two-level utility functions for resource requesters and suppliers. More specifically, the resource provider domains determines how much resource to allocate to each requesters, and resource requester domains decide how much resource to request from each provider. We study the optimal strategies of all players and derive a Nash equilibrium for the Stackelberg game. Our experimental results demonstrate that our proposed solution can effectively reduce the DDoS attack flow to the targeted server, and the resource allocation is fair and provides incentive for domains to maximally help other domains in need. The contributions of this paper include: 1) A novel collaborative DDoS defense network based on network function virtualization technology. 2) A dynamic resource allocation mechanism for domains so that the system is fair, efficient, and incentive-compatible. 3) A multi-leader-follower Stackelberg game model to study the resource allocation results of network domains. 4) An evaluation of our proposed solutions using simulation to verify that the proposed solution is effective, fair, and incentive-compatible.

The rest of this paper is organized as follows: Section II discusses relevant background of the work and previously proposed approaches. We then discuss our proposed framework in Section III. Our resource allocation method and Stackelberg game model are presented in Section IV and the evaluation results are presented in Section V. Finally, we conclude the paper in Section VII.

II. Background and Related Work

In this section we will briefly summarize the existing DDoS attack techniques, traditional DDoS defense strategies, and DDoS defense using network softwarization technologies.

A. DDoS Attack Techniques

DDoS attacks can be roughly divided into two categories: attacks based on IP spoofing and attacks based on real IP addresses. When an attack is carried out using IP spoofing techniques, typically bogus source IPs are used to hide the true IP address of the attackers. An example of such type of attack is SYN Floods [30].

Most DDoS attacks based on real source IPs utilize botnets [28] as attacking source. A bot master orchestrates a large number of compromised devices in the Internet to flood the target. However, the attacking bot nodes can be detected and blacklisted so that the overall cost of using bot nodes is much higher than the spoofing-based attacks. Therefore the botnet-based attack typically generates only a small percentage of the entire attacking stream.

On the other hand, the IP spoofing-based DDoS attacks, such as SYN Floods, can effectively hide the true identities of attacking nodes and also require much less resources to launch the attacks. For SYN Floods, the attacker fabricates a large number of TCP SYN packets using spoofed source IPs to initiate hand-shakings with the victim, which overflows the backlog on the victim to prevent legitimate SYN requests from being processed.

B. DDoS Defense Strategies

Many DDoS defense solutions have been proposed. They can be divided into three categories: (1) solutions on the source only; (2) solutions on the target only; and (3) solutions involving intermediate routers.

Solutions on the source only block attack traffic from the source domain, so that this traffic cannot get into the Internet. For example, the BCP 38 standard [13] enforces ingress routers to verify that the packets from its administered network use legitimate source IP addresses from this network and filters traffic that uses IP addresses outside its subscribed range. This type of solution has proven to be effective and can mitigate IP-spoofing attacks completely when all ISP gateways adopt BCP38. However, it is not practical due to the lack of incentives and corresponding law enforcement.

The solutions involving immediate routers include IP traceback mechanisms [19], and packet marking and filtering mechanisms [29], [34]. This type of solutions is effective only if there is sufficient number of participating routers along the routing paths. It is also not practical due to the lack of incentives and corresponding law enforcement. Also many attacks traverse routers in different countries and it is very difficult to trace the entire attack paths.

The third type of defense involves only the destination infrastructure, for example a firewall or a proxy, that performs filtering of an attack flow. However, dedicated high capacity hardware for DDoS may not be practical for small/medium enterprises due to the high cost. On the other hand, another popular approach is to use commercial cloud-based third party proxies (e.g., Arbor [1] or Prolexic [4]), where the excessive traffic is sent to a DDoS filtering service. However, this approach raises privacy concerns and also gives incentive for the third party DDoS proxy service providers to perform attacks in order to promote their protection products. A solution that is non-profit-driven and does not raise privacy concern is thus needed.

C. DDoS Defense Using Network Softwarization Technologies

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 increase network management flexibility compared to traditional computer networks.

Several approaches have been proposed that use NFV for DDoS defense. Fayaz *et al.* propose Bohatei [12], a DDoS defense solution using SDN and NFV. 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 uses a limited amount of network resources (to launches VMs), it can not be effective in case of attacks in which the 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.* [22] propose a SDN-based approach to overcome legitimate looking DDoS attacks. 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.* [17] designed an architecture of cloud-based firewall service using NFV technology and other network virtualization capabilities to defend DDoS attacks. The cloud architecture acts as an intermediary to filter the attack traffic for the customer and transmits only legitimate traffic.

Chen *et al.* [33] propose *SDNShield*, a defense mechanism against DDoS attacks on SDN control plane. They use specialized software boxes in order to improve the scalability of ingress SDN switches to accommodate control plane workload surges. The proposed approach has two main steps. They first statistically filter legitimate flows from attack ones, and then recover the false positives of the first filtration with in-depth TCP handshake verification.

Beigi-Mohammadi *et al.* [8] propose CAAMP, an automated DDoS attack mitigation system using hybrid-Clouds. CAAMP is designed specifically to mitigate DDoS attacks on public cloud applications using SDN and NFV techniques. The main function of CAAMP is that once a suspicious traffic is identified, a copy of the application's topology on-the-fly and transfer to an isolated environment in a private cloud. A SDN controller is designed to to create the virtual switches dynamically to redirect the suspicious traffic to a shark tank until final decision is made.

Xu and Liu [32] propose a solution to detect DDoS attacks leveraging on SDN's flow monitoring capability. The proposed approach utilizes measurement resources available in the SDN network to dynamically balance the coverage and granularity of attack detection. The proposed solution addresses two issues. First, capture traffic rate and its diversion from a regular traffic. Second, utilizing available resources in the whole SDN network to monitor suspicious traffics.

Our previous work VGuard [14] proposes a traffic prioritization algorithm to reduce the impact from DDoS attacks based on real source IPs. Our another work VFence [18] is specially effective in reducing 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 through delegated handshaking and verified traffic will be further forwarded to the server. Figure 1(a) shows the scenario that an IPS handles traffic with legitimate source IP addresses and Figure 1(b) shows the scenario that an IPS handles

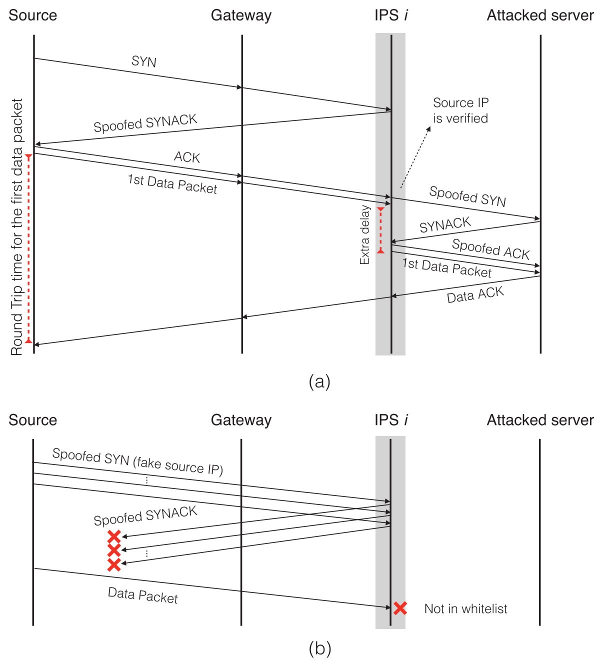

Fig. 1. SYN Flood Attack overview: (a) a scenario on how the IPS handles traffic with authentic source IP addresses; (b) a scenario how the IPS blocks DDoS attacks with spoofed source IP addresses.

SYN-flood traffic with spoofed source IP addresses. However, VFence is not effective when the attack volume is sufficiently high to overwhelm the available local resource. CoFence can overcome this obstacle by utilizing external resources in the collaboration network. Hence, it further improves the DDoS resistance level through collaboration 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 other to enhance resistance against large-scale DDoS attacks. With the help of NFV technology, each NFV-enabled domain network can contribute their spare network resources to help other domains in the network when needed.

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 domain 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.

Figure 2 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 exceeds 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 the filtered traffic is forwarded back to domain 1.

Figure 3 illustrates the collaboration network topology. The shared IPS resources from all domains are organized into a

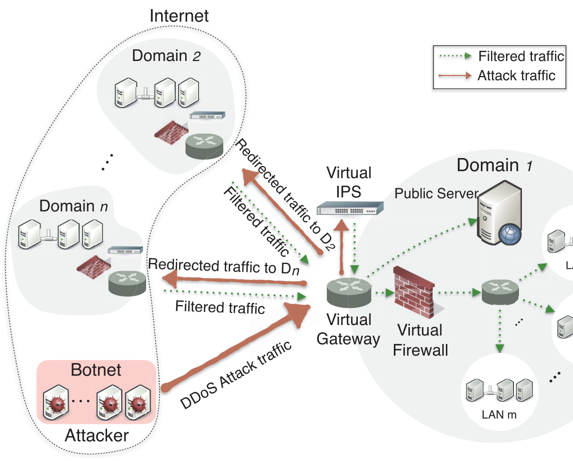

Fig. 2. A case study of collaborative DDoS defense.

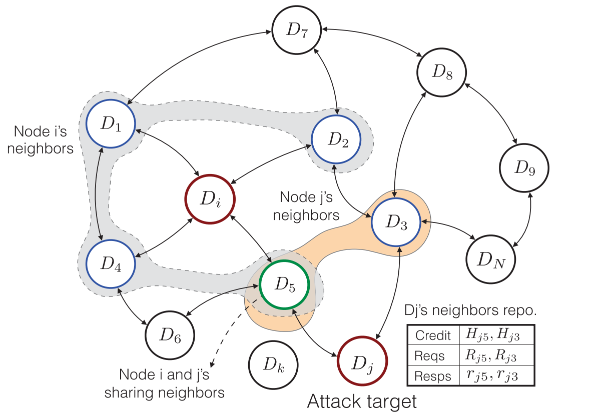

Fig. 3. An example of the virtual network of collaborative network graph.

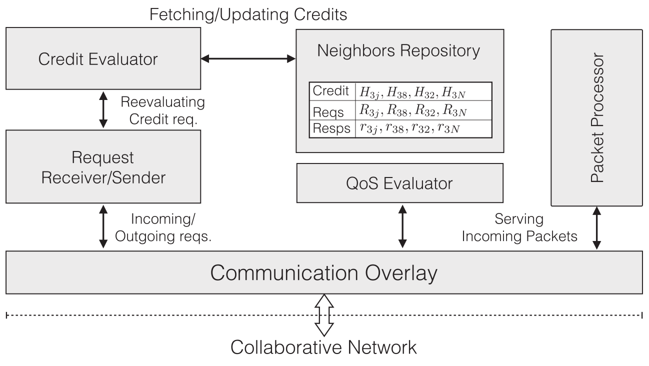

Fig. 4. Domain's public server architecture.

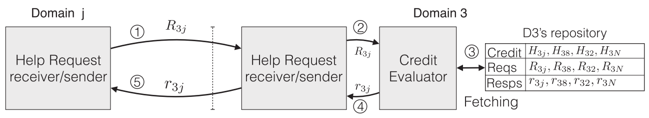
Domain j
Help Request receiver/sender
① R_{3j}
⑤ r_{3j}
Help Request receiver/sender
Domain 3
② R_{3j}
④ r_{3j}
Credit Evaluator
③
D3's repository
Credit $H_{3j}, H_{38}, H_{32}, H_{3N}$
Reqs $R_{3j}, R_{38}, R_{32}, R_{3N}$
Resps $r_{3j}, r_{38}, r_{32}, r_{3N}$
Fetching

Fig. 5. Request/Response flow of n_j, n_3.

virtual network. Domains are connected to their trusted neighbor domains. For example, a university may have multiple campuses and they can trust each other. In this example, domain i has four trusted neighbors (D_1, D_2, D_4, D_5) and domain j has two trusted neighbors (D_3, D_5). A domain can be trusted by multiple other domains (e.g., D_5). Each domain maintains a repository to store information about its neighbors, including the neighbor's credit (how much that neighbor has helped in the past), the amount of resource the neighbor is requesting, and the amount of resource offered to the neighbor.

Figure 4 illustrates a domain's public server internal architecture equipped with CoFence. We explain every component of this architecture in the next sections.

To be able to collaborate with other domains, a virtual IPS should include 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 new neighbors and respond to the neighbor adding request; (c) request to remove neighbors and respond to the neighbor removal request. For example, as illustrated in Figure 5, domain j under attack sends a request to its neighbor (domain 3) for helping with DDoS traffic handling, and domain 3 responds with the decision regarding the amount it offers to help. After finishing the negotiation, domain j directs part of the agreed-upon traffic flow to domain 3. Note that all administration communication should be encrypted to ensure confidentiality and integrity.

2) Resource Allocation: After receiving a help 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.

In the next section, we describe the resource allocation mechanism that allows each domain to determine how much it should help its neighbors.

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 introduce the resource allocation model to fulfill the design goals.

A. The Design Goals of Resource Allocation

Our design goals can be stated as follows. First of all we are interested in designing a collaboration system which is fair, incentive-compatible, and reciprocal. The fairness property means the system can control the discrepancy of the received help among different nodes so that 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 rewards to participants who 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 relationship. For example, the more node x helps node y,

TABLE I
SUMMARY OF NOTATIONS

Notation	Meaning
$\mathcal{N}$	Set of resource providers
$\mathcal{M}$	Set of resource requesters
$\mathcal{N}_i$	Set of all neighbors for node i
$\mathcal{S}_i$	Set of neighbor nodes requesting help from node i
$\mathbf{r}$	The matrix recording the helping data rate among nodes
$\vec{r}_i$	The helping data rate from domain i to all neighbors
r_{ij}	The helping data rate from domain i to node j
R_{ij}	The requested helping rate from i to j (requested by node j)
$S_{ij}(r_{ij})$	The satisfaction level of node j towards node i given helping rate r_{ij}
$\hat{r}_i$	Total reserved helping resource amount on node i
$\hat{R}_j$	The actual required resource to handle DDoS attack at node j
H_{ij}	Helping credit - the level of helpfulness from i to j in the past
λ	Forgetting factor
k	Fairness factor

the more node y helps x in return. In terms of performance we aim at a system that is 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 Optimization

In this section we model the optimal strategy for domains to allocate resource when requested by other domains under DDoS attacks. We model the CoFence network consisting of n domains (nodes) using a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ denotes the set of domains (nodes) and $\mathcal{E}$ represents the connections in between pair of nodes if they have established a *trusted* collaboration relationship. Domains (nodes) in the network share excessive traffic with other trusted domains (nodes) under DDoS attack. We use $\mathcal{N}_i$ to denote the set of *neighbors domain (node)* of domain (node) i. i.e., they are connected to domain (node) i directly by edges in $\mathcal{G}$. When a domain (node) i receives a request for help from its neighbor j, helping resource will be allocated using a resource allocation algorithm described in the next subsection. For convenience of presentation, we use the term node as a synonym for the concept of domain in the rest of this paper.

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 r_{ij} to denote the the rraffic handing volume that node i offers to node j, and a vector $\vec{r}_i$ is used to represent the traffic handling volume that domain i offers to help all the requesters. We 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., $r_{ij} \le R_{ij}$. Also the total helping rate should not exceed the spare resource that the offering node is willing to share, i.e., $\sum_{j\in\mathcal{S}_i} r_{ij} \le \hat{r}_i$, 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 $H_{ij} \ge 0$ represents the level of helpfulness from node i perceived by node j. Note that the matrix $\mathbf{H}$ can be asymmetric, i.e., $H_{ij} \ne H_{ji}$. 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 using the following formula:

$$\arg\max_{\vec{r}_i} U_i^h(\vec{r}_i) := \sum_{j\in\mathcal{S}_i} H_{ji} S_{ij}(r_{ij}) \tag{1}$$

$$\sum_{j\in\mathcal{S}_i} r_{ij} \le \hat{r}_i, \tag{2}$$

$$0 \le r_{ij} \le R_{ij}, \tag{3}$$

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

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

where $k \ge 0$ is the fairness factor, which controls the sensitivity of resource allocation thus impact the fairness of the system. 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. Maximizing this utility allows a node to provide more help to those from whom there was more help in the past.

The constraint (2) shows that the total helping rate shall not exceed the resource that the offering node is willing to share. The constraint (3) means that the resource a node offers shall not exceed the requested amount and shall not be a negative value. We can see that when $\hat{r}_i$ is sufficiently large, (2) is an *inactive constraint* (which means the solutions are not at the boundary set by the constraint), and thus 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 $\mathcal{L}^i : \mathbb{R}^{n_i} \times \mathbb{R} \times \mathbb{R}^{n_i} \to \mathbb{R}$

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

where $\mu_i, \delta_{ij}, \phi_{ij} \in \mathbb{R}_+$ satisfy the complementarity conditions $\mu_i\left(\sum_{j\in\mathcal{S}_i} r_{ij} - \hat{r}_i\right) = 0$, $\delta_{ij}(r_{ij} - R_{ij}) = 0$, and $\phi_{ij} r_{ij} = 0, \forall j \in \mathcal{S}_i$.

The optimization problem of (1) with constraints (2) and (3) is equivalent to finding solutions for the following set of equations:

$$\arg\max_{\vec{r}_i} \mathcal{L}^i(\vec{r}_i, \mu_i, \delta_{ij}, \phi_{ij}) \tag{6}$$

$$\mu_i \left(\sum_{j \in \mathcal{S}_i} r_{ij} - \hat{r}_i\right) = 0 \tag{7}$$

$$\delta_{ij}(r_{ij} - R_{ij}) = 0, \quad \forall j \in \mathcal{S}_i \tag{8}$$

$$\phi_{ij} r_{ij} = 0, \quad \forall j \in \mathcal{S}_i \tag{9}$$

We maximize the Lagrangian with respect to $\vec{r}_i \in \mathbb{R}_+^{|\mathcal{S}_i|}$ and obtain the first-order *Kuhn-Tucker condition*:

$$\frac{kH_{ji}}{kr_{ij} + R_{ij}} = \mu_i + \delta_{ij} - \phi_{ij}, \quad \forall j \in \mathcal{S}_i.$$

When (2) is active ($\mu_i \neq 0$) but (3) is inactive ($\delta_{ij} = 0$ and $\phi_{ij} = 0$), we can find an closed-form solution supplied with the equality condition

$$\sum_{j \in \mathcal{S}_i} r_{ij} - \hat{r}_i = 0, \tag{10}$$

and consequently, we obtain the optimal solution

$$r_{ij}^\star := \frac{H_{ji}}{\sum_{u \in \mathcal{S}_i} H_{ui}} \left(\hat{r}_i + \frac{1}{k} \sum_{v \in \mathcal{S}_i} R_{iv}\right) - \frac{1}{k} R_{ij}. \tag{11}$$

When the constraint (3) is active, the optimal solution is attained at the boundaries. 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 (11) 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 $|\mathcal{S}_i|$ 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 $\frac{\partial r_{ij}^\star}{\partial R_{ij}} = \frac{1}{k}\left(\frac{H_{ji}}{\sum_{u \in \mathcal{S}_i} H_{ui}} - 1\right) < 0$, 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}.*

To compute the numerical solution to optimization problem (1) under all conditions, each domain can use an iterative computation method described in Algorithm 1 to find a solution to optimize their utility.

The overall idea of Algorithm 1 is to start from a feasible solution (line 8). It then continuously move the resource from the node with the lowest marginal gain to the node with the highest marginal gain till no more move is possible to increase the overall utility or the changing step is sufficiently small.

Algorithm 1 Optimal Resource Allocation for Domain u

```
1: // This algorithm describes the algorithm for each domain $u$ to find the optimal resource allocation given requested amount and past credit.
2: **Inputs** :
3: $\mathcal{N}_u$: the set of neighbors of domain $u$ that are trusted by domain $u$
4: $\vec{H}_u$: helping credits for all neighbors of domain $u$
5: $\vec{R}_u$: requested helping resource needed by the neighbors of domain $u$
6: $\vec{r}_u$: allocated helping resource for the neighbors of domain $u$
7: $\hat{r}_u$: the total available resource for domain $u$.
8: $U$: the utility function.
9: // Iteratively compute the amount of resource allocated to requested peers given their requested amount till the aggregated satisfaction is optimal.
10: $\vec{r}_u \Leftarrow \{r_{uj} = \frac{\hat{r}_u R_{uj}}{\sum_v R_{uv}}\}$ // initially all resource is distributed proportionally regarding to request.
11: $\vec{r'}_u(\vec{r}_u) \Leftarrow \{r'_{uj} | r'_{uj} = \frac{H_{ju}}{R_{uj}+r_{uj}}, \forall j \in \mathcal{N}_u\}$ // the marginal growth.
12: $\Delta r \Leftarrow r^0$ // initialize the changing step
13: **while** $\Delta r > \epsilon$ **do**
14:   $a \Leftarrow$ **Highest**$(\vec{r'}_u)$ // index of the domain with the highest marginal gain except at the upper bound
15:   $b \Leftarrow$ **Lowest**$(\vec{r'}_u)$ // index of the domain with the lowest marginal gain except at the lower bound
16:   **if** $a \neq b$ **then**
17:     $\Delta r = min(\Delta r, R_{ua} - r_{ua}, r_{ub})$ // adjustment step to keep result in boundary
18:     // if moving $\Delta r$ resource from b to a improves the overall utility then move it; otherwise reduce the step size by half.
19:     **if** $U(r_{ua} + \Delta r, r_{ub} - \Delta r) - U(r_{ua}, r_{ub}) > 0$ **then**
20:       $r_{ua} = r_{ua} + \Delta r$
21:       $r_{ub} = r_{ub} - \Delta r$
22:     **else**
23:       $\Delta r = \Delta r/2$
24:     **end if**
25:   **end if**
26: **end while**
27: return $\vec{r}_u$ // helping resource domain $u$ allocates to neighbors
```

C. A Stackelberg Game Model for CoFence

The previous optimization problem gives solution on how much resource a provider domain u shall allocate to maximally satisfy its requesting neighbors. A requesting domain j has another degree of freedom to choose its level of requested resource amount (which may be different from its actual desired resource amount). Let R_{uj} denote the amount of help that domain j requests from domain u. The next question is how much resource a domain j shall request its neighbors in order to obtain sufficient help? In addition to the aggregated satisfaction utility optimization problem controlled by the resource offering domains described in Equation (1), the optimization at this level is a second tier optimization problems controlled by the resource requesting domains. The objective of an under-attack node j is to choose $\vec{R}_j$ so that its utility $U_j^r : \mathbb{R}_+^{n_j} \to \mathbb{R}$ is maximized, i.e.,

$$\arg\max_{\vec{R}_j} U_j^r(\vec{R}_j), \tag{12}$$

subject to the constraint that the requested resource is non-negative, i.e.,

$$R_{uj} \geq 0, \quad \forall u \in \mathcal{N}_j$$

Let $U_j^r(\vec{R}_j)$ take the form of

$$U_j^r(\vec{R}_j) := \sum_{u \in \mathcal{N}_j} \log_2(1 + r_{uj}^\star / \hat{R}_j), \tag{13}$$

where $r_{uj}^\star$ is the optimal solution by optimizing (1); $\hat{R}_j$ is the amount of resource domain j actually needs. The log function indicates that the requesting domain prefers a more distributed helping resource from multiple domains with less load on each helper, than fewer helping resources with more load on each helper. This is a reasonable assumption since as more flows are distributed to multiple destinations, the lower the chance of traffic jamming on some routers. The utility optimization can be further expanded as follows.

$$\arg\max_{\vec{R}_j} \sum_{u \in \mathcal{N}_j} \log_2\left(1 + \frac{r_{uj}^\star}{\hat{R}_j}\right). \tag{14}$$

subject to,

$$\sum_{u \in \mathcal{N}_j} R_{uj} \leq \hat{R}_j \tag{15}$$

$$R_{uj} \geq 0, \quad \forall u \in \mathcal{N}_j \tag{16}$$

where (15) indicates the received helping offer should be no more than the actual need. In this game the change of the requested resource $\vec{R}_j$ of domain j will have impact to the choice of the optimal solution from its neighbors ($r_{uj}^\star$), which in turn influence the utility of j. All domains will solve their coupled optimization problems by tuning their $\vec{R}_j$ till all domains reach their local optimal. This game can be seen as a *Multi-leader-follower Stackelberg Game* [9]. Generally, in a Stackelberg game, there are one or more distinct players called the leaders, who optimizes the upper-level problem, and a number of remaining players called the followers, who optimize the lower-level problems jointly. In particular, the leaders anticipates the responses of the followers, and then uses this ability to select their optimal strategy. At the same time, all followers select their own optimal responses parameterized by the leader's decision. In our case, the resource requesting domains are the leaders and the resource suppliers are the followers. The leaders will find their optimal requesting amount and the followers will find their optimal offering amount to each requester based on their credits and the requested amount.

Definition 1: The multi-player non-cooperative DDoS defense resource allocation game is a multi-leader-follower Stackelberg game with the followers' utility functions defined by Formula (1) and the leaders' utility function defined by (12).

Given there are M resource requesters and N resource providers, the *Stackelberg-Nash equilibrium* is a situation where none of the resource requester can improve its utility by changing its requested amount alone. For any given requests set, the followers will reach a Nash equilibrium in every scenario. We now give a rigorous definition of the equilibrium.

Definition 2: The Nash equilibrium of the M leader N follower resource allocation Stackelberg game is an M + N tuple $(\vec{R}_1^*, \ldots, \vec{R}_M^*; \vec{r^*}_1, \vec{r^*}_N)$ *such that,*

$$\vec{R}_j^* \in \arg\max_{\vec{R}_j} \sum_{u \in \mathcal{N}_j} \log_2(1 + r_{uj}^* / \hat{R}_j), \quad \forall j$$

$$\vec{r^*}_i \in \arg\max_{\vec{r}_i} \sum_{v \in \mathcal{S}_i} H_{vi} \log_2\left(1 + k\frac{r_{iv}}{R_{iv}^*}\right), \quad \forall i$$

Theorem 1: When (2) is active but (3), (15) and (16) are inactive, a Nash equilibrium for the Stackelberg game can be computed as follows,

$$R_{ij}^* = r_{ij}^* = \frac{H_{ji}}{\sum_{v \in \mathcal{S}_i} H_{vi}} \hat{r}_i. \tag{17}$$

Proof: When (15) and (16) are inactive (solutions are not at the boundaries), we first take the derivative of the utility function of the leader domain j over the requesting amount R_{ij}. We have,

$$\frac{\partial U_j^r(\vec{R}_j)}{\partial R_{ij}} = \frac{1}{\hat{R}_j + r_{ij}^\star} \frac{\partial r_{ij}^\star}{\partial R_{ij}} \tag{18}$$

$$= \frac{1}{\hat{R}_j + r_{ij}^\star}\left(\frac{H_{ji}}{\sum_{u \in \mathcal{S}_i} H_{ui}} - 1\right) < 0 \tag{19}$$

We can see that over stating R_{ij} decreases the utility of the leader domains. To obtain the maximized utility, the leader domain j shall understate the requesting amount R_{ij} till the best response from domain i meets the requested amount, i.e., $r_{ij}^\star = R_{ij}$. By definition, the Nash equilibrium state means all domains are playing their best strategies, i.e.,

$$\vec{R}_j^* \in \arg\max_{\vec{R}_j} \sum_{u \in \mathcal{N}_j} \log_2(1 + r_{uj}^* / \hat{R}_j), \quad \forall j$$

$$R_{ij}^* = r_{ij}^*, \quad \forall i, j$$

From (11) we get,

$$R_{ij}^* = r_{ij}^* = \frac{k}{1+k} \frac{H_{ji}}{\sum_{v \in \mathcal{S}_i} H_{vi}} \left(\hat{r}_i + \frac{1}{k} \sum_{v \in \mathcal{S}_i} R_{iv}^*\right), \quad \forall i, j. \tag{20}$$

Give that (2) is active, we have $\sum_{v \in \mathcal{S}_i} R_{iv}^* = \sum_{v \in \mathcal{S}_i} r_{iv}^* = \hat{r}_i$. Replace $\sum_{v \in \mathcal{S}_i} R_{iv}^*$ with $\hat{r}_i$ in (20) and we get (17). □

D. The Performance of the Stackelberg Game

An alternative problem formulation is to assume that there exists a coordinator that can realize an optimal resource allocation by directing resource flows from one node to another.

The optimization problem can be formulated as following:

$$\arg\max_{\mathbf{e}} U^o(\mathbf{e}) := \sum_{j\in\mathcal{M}} H_j \log_2\left(1 + k\frac{e_j}{\hat{R}_j}\right) \tag{21}$$

$$\sum_{j\in\mathcal{M}} e_j \le \sum_{i\in\mathcal{N}} \hat{r}_i, \tag{22}$$

$$0 \le e_j \le \hat{R}_j, \tag{23}$$

where e_j is the total received resource by node j; H_j is the accumulated helping credit that node j has helped all other nodes in the past. The Oracle collects shared resource from all resource providers and optimally allocated to the requesters based on their credit and needs.

The above optimization problem has a trivial solution $e_j = \hat{R}_j, \forall j \in \mathcal{M}$ when the resource is abundant, i.e., $\sum_{i\in\mathcal{M}} \hat{R}_i \le \sum_{i\in\mathcal{N}} \hat{r}_i$. Otherwise a closed-form optimal solution can be obtained as follows under the condition that constraint (22) is active and (23) is inactive (solutions are not at the borders):

$$e_j^\star = \frac{H_j}{\sum_{u\in\mathcal{M}} H_u}\left(\sum_{i\in\mathcal{N}} \hat{r}_i + \frac{1}{k}\sum_{j\in\mathcal{M}} \hat{R}_j\right) - \frac{1}{k}\hat{R}_j \tag{24}$$

From (17) we can get the received helping resource by node j under the Stackelberg-Nash Equilibrium can be computed as:

$$e_j^* = \sum_{i\in\mathcal{N}} r_{ij}^* = \sum_{i\in\mathcal{N}} \frac{H_{ji}}{\sum_{v\in\mathcal{M}} H_{vi}} \hat{r}_i. \tag{25}$$

The reasons for not adopting the optimal mode include: 1) scalability issue with the centralized model; 2) single point of failure; 3) difficulty to deal with dishonest nodes providing false information to the coordinator. We will compare the performance of the Stackelberg game with the optimal model in our evaluation section.

E. 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 has 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 tracking the helping resource 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:

$$H_{ij}(t_0) = \int_{-\infty}^{t_0} r_{ij}(t)\lambda^{(t_0-t)}dt \tag{26}$$

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 forgetting speed. If $\lambda = 1$ then all past credits will be remembered and carry the same weight as new credits. 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:

$$\begin{aligned} H_{ij}(t_0) &= \int_{-\infty}^{t'} r_{ij}(t)\lambda^{(t_0-t)}dt + \int_{t'}^{t_0} r_{ij}(t)\lambda^{(t_0-t)}dt \\ &= \int_{-\infty}^{t'} r_{ij}(t)\lambda^{(t'-t)}\lambda^{(t_0-t')}dt + \int_{t'}^{t_0} r_{ij}(t)\lambda^{(t_0-t)}dt \\ &= \lambda^{\Delta t} H_{ij}(t-\Delta t) + \int_{t_0-\Delta t}^{t_0} r_{ij}(t)\lambda^{(t_0-t)}dt \end{aligned} \tag{27}$$

Equation (28) 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 at which a new node i joins the network to be 0 with initial credit c_i, and the current time be a relative time T after the initial joining time, then the helping credit perceived by j can be computed as follows:

$$H_{ij}(T) = \lambda^T c_i + \int_0^T r_{ij}(t)\lambda^{(T-t)}dt \tag{28}$$

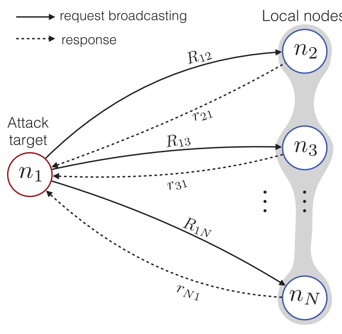

Fig. 6. An example of the protocol for requesting help.

F. 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 its collaborators about the desired help, a node under attack can broadcast the helping requests use the following algorithm.

As described in Algorithm 2, 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, the requesting process terminates. 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. Figure 6 illustrates a case study of the process for help request. Node n_1 sends requests to its neighbor nodes $n_2, n_3, \cdots, n_n$ sequentially.

V. Evaluation

In this section we present our experiments evaluating the proposed collaborative model. We first explain our experimental setup and then the results on the performance of the model. We conducted a series of experiments on different case studies to evaluate the performance of the model. We also did an

Algorithm 2 Seek Help by Node u

```
1:  // This algorithm describes the algorithm for a node to broadcast its requested help to its neighbors. It is triggered when DDoS attacks are detected.
2:  **Inputs** :
3:  $\mathcal{N}_u$: the set of neighbor domains that are trusted by domain $u$
4:  $\vec{H}_u$: helping credits for all neighbors of domain $u$ in descending order
5:  $A_u$: required helping resource needed for domain $u$ during DDoS attack
6:  $\Delta t$: the time interval to recompute the credit of all neighbors
7:  // Send request for help to each trusted neighbor ordered by their level of helpfulness in the past.
8:  **for each** node $v$ in $\vec{H}_u$ **do**
9:      $h$ $\Leftarrow$sendRequest($v$,$A_u$) //$v$ computes resource offer using Eq. (1)
10:     $A_u = A_u - h$ //Reduce the required amounts after receiving help
11:     **if** $A_u = 0$ **then**
12:         **break** the for loop
13:     **end if**
14: **end for**
15: **RedirectTraffic** to helpers
16: **set timer** ($\Delta t$, "**UpdateCredit**($\mathcal{N}_u$)") //Update neighbors' helping credits periodically (every $\Delta t$)
```

experiment to evaluate the impact of the attack on nodes with different level of generosity and amount of reciprocal help that they receive from each other.

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* [5] 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 other domains.

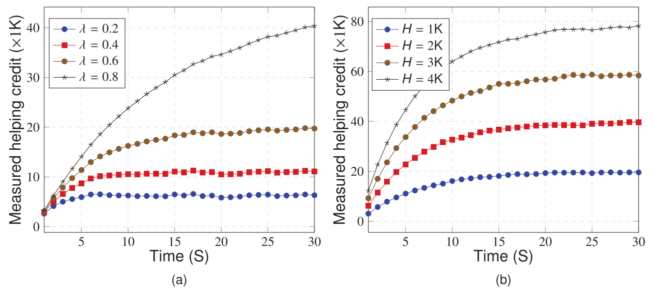

Fig. 7. Helping credit measurement: (a) the helping credit of node j perceived by node i for a fix helping rate $H = r_{ji} = 1K$/second at different λ settings; (b) the helping credit of node j perceived by node i for a fixed value of $\lambda = 0.6$ and different settings of help rate (H) from node j.

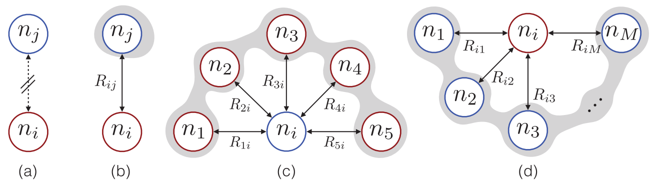

Fig. 8. Case studies: (a) attack target n_i is not equipped with CoFence; (b) n_i is using CoFence and has one neighbor; (c) attack targets $n_1 - n_5$ share one neighbor node (n_i); (d) attack target n_i has a list of neighbor nodes $n_1 - n_M$ in the network.

Equation (26) 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 7(a) shows that under all λ settings the the helping credits increase with time at the beginning and converges to stable. A higher λ leads to a higher helping credit value.

In the next experiment, we fixed the value of λ to 0.6 and let $H = r_{ji}$ increases from 1K to 4K with step 1K. Figure 7(b) shows that the more generous that node j helps node i, the higher helping credit it gains.

C. Case Studies

In the next set of experiments we use four case studies, as shown in Figure 8, to evaluate the efficiency of our proposed CoFence model against DDoS attacks. Figure 9 (a) shows the traffic trace we use in these case studies. We can see that the normal traffic to both nodes is 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 time 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 in Figure 8 (a). In this scenario node i is under attack. Since node i can only handle a data rate of $2K$/second, much traffic to node i has to dropped. Figure 9 (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 with minimum packets drops.

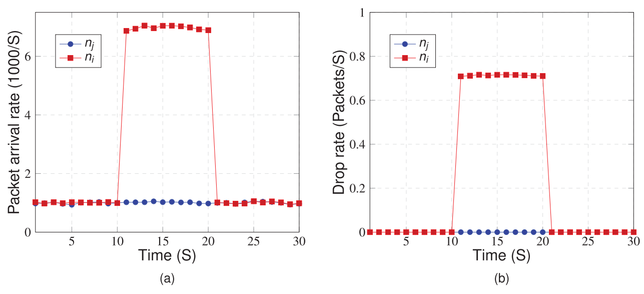

Fig. 9. Incoming packet and drop rate for case study (a).

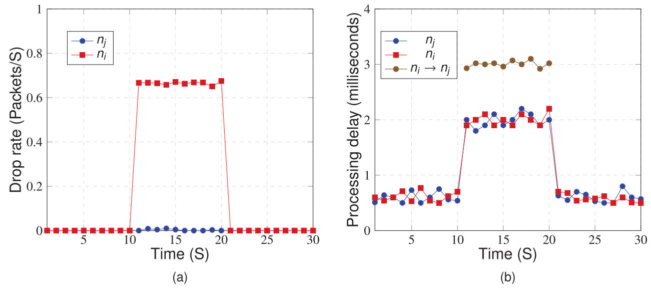

Fig. 10. Average dropping rate and processing delay for case study (b).

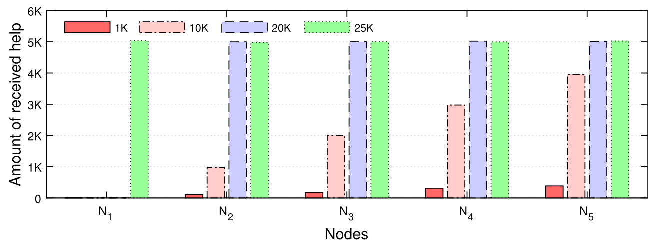

Fig. 11. Amount of help that nodes $1-5$ receive for different available capacities of node i (case study (c)).

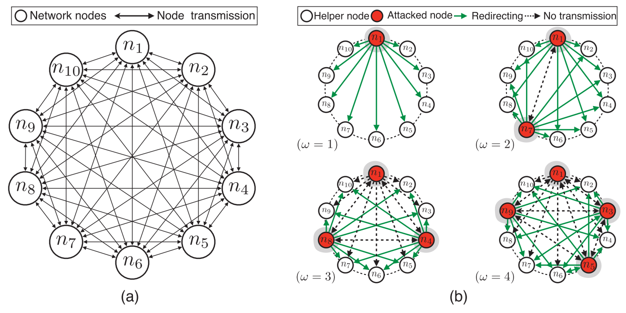

Fig. 12. A fully-connected graph to evaluate the nodes' reciprocal help.

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 8 (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 10 (a) illustrates the packet dropping rate 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.

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 10 (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 ($n_i \rightarrow n_j$).

3) Case Study 3: In the third experiment, we evaluate the case in which 5 nodes ($n_1 \sim n_5$) are under attack and all of them share one helper node (n_i) (Figure 8(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 11 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.

4) Case Study 4: In the fourth experiment we evaluate the impact of the number of neighbors (helper nodes) for the attacked node. Figure 8 (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 13 (a) 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.

D. Fairness and Incentive Compatibility

Finally we evaluate the fairness and incentive compatibility of the proposed resource allocation mechanism. The case study is a fully connected network consisting of 10 nodes as shown in Figure 12 (a). We let each node in this network provide different level of help, we name it *generosity level*, to others ranging from 0 to $4.5K$ packets per second. For example, a node with generosity level of 0 provides no help to others, while node with generosity level 4.5K help others to handle at most $4.5K$ packets per second. Table II lists the generosity

TABLE II
NODES' GENEROSITIES (HELPING AMOUNT)

Node	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
Traffic (Pkt/S)	0	500	1K	1.5K	2K	2.5K	3K	3.5K	4K	4.5K

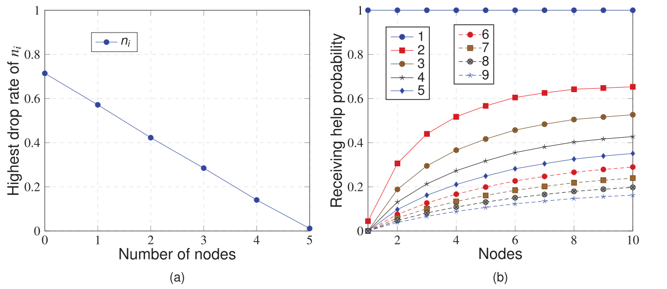

Fig. 13. (a) the impact of different number of nodes $(1-6)$ in a attack target node's locality on packet drop rate (case study (d)); (b) reciprocal help when different number of nodes are under attack and nodes have different levels of generosity (the amount of traffic that nodes can help others to handle when other nodes are under attack).

level for each node. We can see that the maximum external helping resource a node can receive is 22K/s.

Our attack scenario in this experiment is to randomly attack a number of nodes and evaluate the amount of help those nodes can receive from other nodes. In each round we randomly select ω number nodes and attack each selected nodes with a DDoS packets rate of $18K$/s. We set ω with different values from 1 to 9. We repeat each round 1000 times to compute the average percentage of received help from others. Figure 12(b) illustrate the helping map when ω is 1, 2, 3, and 4.

Figure 13 (b) shows the received helping rate for each node under all ω settings. We can see that when there is only one node under attack, the only node will receive sufficient helping resource from other nodes, even the attacked node has no credit. This reflect the fairness of this resource mechanism. a.k.a., no node should starve if there is sufficient resource. However, when multiple nodes are under attack, the nodes with higher generosity also receive higher amount of helping resource in return. This reflects the incentive-compatibility of the design, where participants are encouraged to be more generous in terms of helping others.

E. Stackelberg Game Evaluation

In this section, we evaluate the performance of the Stackelberg game model and compare it with the optimal model.

1) Experiment Setup: We use a case study with 5 domains $(n_1, \cdots, n_5)$ as shown in Figure 17(b) as our experimental scenario. The domains are set with different levels of generosity (the amount of resource that they share with others). Table III shows the amount of resource each domains shares ranging from $1K$ to $3K$. The experiment was carried out in many rounds. In each round we let 4 nodes under DDoS attack and one node (helper) provide resource to help others. Each node has equal chance to be the resource provider. The desired helping resource for each attack target was $1.5K$.

TABLE III
THE GENEROSITIES OF NODES (SHARED RESOURCE AMOUNT)

Node	n_1	n_2	n_3	n_4	n_5
Shared Resource (Pkt/S)	1K	1.5K	2K	2.5K	3K

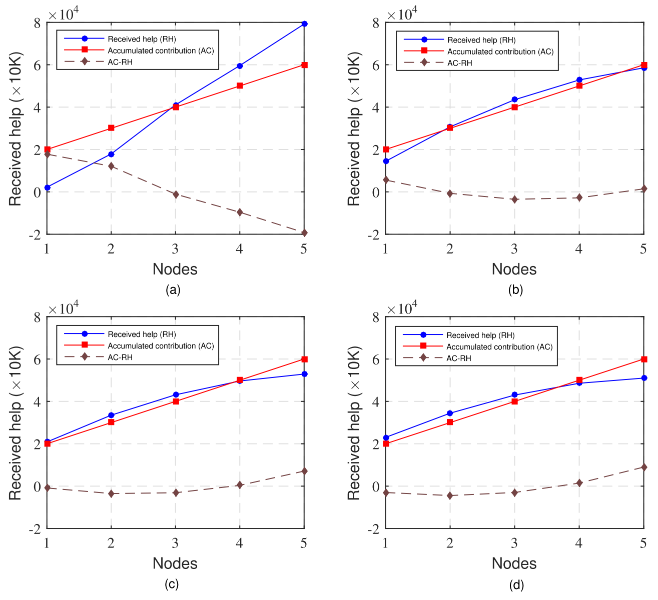

Fig. 14. The resource allocation results with different impact factor settings of $k = 1, 4, 8$ and 12 .

2) The Impact From the Fairness Factor: In this experiment, we study the impact from the fairness factor k that has been used in equations (4) and (21). Note that the coordinator in the optimal model and the resource providers in the game model use the same resource allocation utility function except that they have different helping credit evaluation methodologies. In the optimal model the coordinator keeps track of the helping credits of all participants, while in the game model resource providers track the helping credits of their neighbors only. In our experiment, all helping credits start from small values and are updated through iterative interactions among participants.

Figure 14 (a) to (d) show the received helping resource of all participating nodes under different fairness factor settings $k = \{1, 4, 8, 12\}$, respectively. We can see that for all nodes the received help increases with their accumulated contributions. The higher k value is, the more equality is shown among all participating nodes. When $k = 1$ we can see the sign of starving for nodes with low shared resource and when $k = 12$ the difference of received help among all nodes are the smallest. From the above results we can see that the fairness factor k can be used to tune the equality of nodes in the networks.

3) The Nash Equilibrium for the Stackelberg Game: In this experiment we run the Stackelberg game among the 5 participants and observe their best responses in each iteration and the formation of the Stackelberg-Nash Equilibrium. Each resource requester will adjust their *requested resource* R_{ij} based on the *allocated resource* r_{ij} to achieve improved utility in the

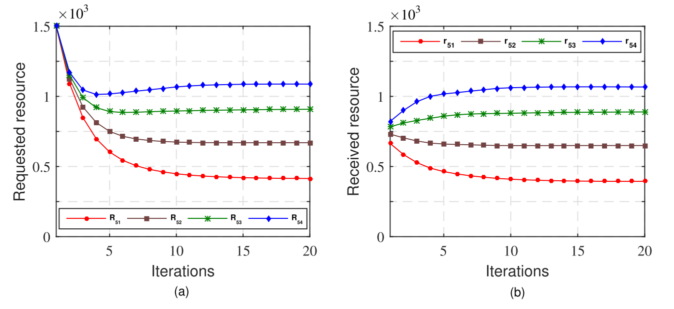
Fig. 15. (a) Requested resource by nodes 1, 2, 3, 4 from node 5 in 20 negotiation rounds; (b) Allocated resource from node 5 to nodes 1, 2, 3, 4 in 20 negotiation rounds.

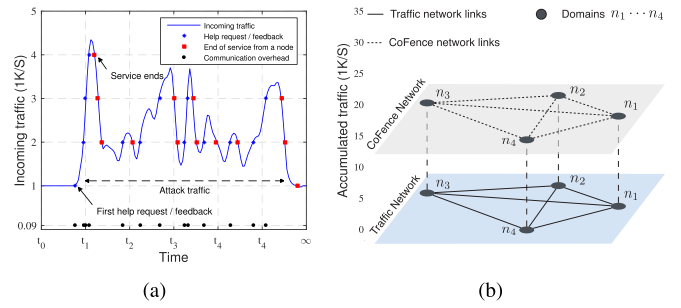
Fig. 17. Network communication overhead: (a) shows that incoming attack traffic to domain n_1 and outgoing help requests from this domain; (b) The CoFence's network is on the different network slice as the traffic network.

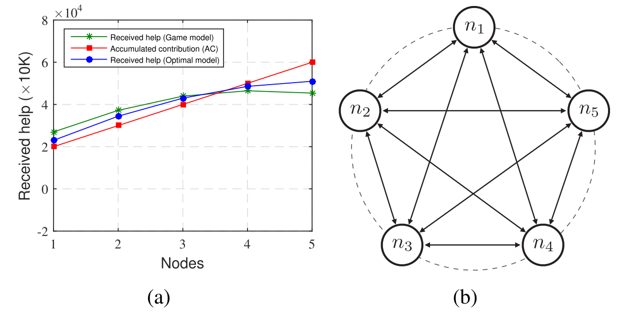
Fig. 16. (a) Comparison of amount of received help by domains $n_1, \cdots, n_5$ using game-model and optimal allocation; (b) An example network structure composed of 5 domains $n_1, \cdots, n_5$.

next round. Figure 15 shows the changes of the allocated resources and the requested resources after each negotiation iterations in 20 rounds. We can see that the *requested resources* set by the game leaders converge to Nash Equilibria and so do the allocated resources set by the game followers.

4) The Performance of the Stackelberg Game: In this experiment we compare the performance of the Stackelberg game model with the optimal model. We ran both models under the same setting with $k = 12$ and as of Table III. Figure 16(a) shows that the amount allocated resources to each node under the Stackelberg game assembles the optimal allocation. The game model results in a further equalized resource allocation. From this result, we can see that the distributed Stackelberg game model can achieve near optimal resource allocation while avoiding the shortcomings from the centralized model.

VI. Discussion and Security Threats

Although the purpose of CoFence is to facilitate a collaboration framework among NFV-based peer domain networks and protect networks against DDoS attacks, CoFence itself may be the target of attacks. In this section, we discuss a few potential threats and issues to CoFence and our future plans for addressing the issues.

A. The Cost of CoFence

In this section, we discuss the cost of the CoFence. Our discussion covers network communication overhead, financial (economic) cost, and the cost of computational resources.

1) Communication Overhead: Regarding the network bandwidth usage and domains communication, we have embedded a component which handles all the communication with other domains in the collaborative network. Messages passing through the communication overlay include: help-requests to/from neighbors, and responses from/to neighbors. It enables domains from different vendors to communicate through a common protocol. We discuss the *interoperability* challenge of CoFence in more details next.

We like to emphasize that the CoFence's communication overhead (negotiating help requests and responses) among domains do not have a significant impact on the communication network. To demonstrate the overhead of communication, we conducted an experiment on a network with 10 domains as shown in Figure 12(a). Each domain has the capacity of 1K to process incoming packets. A domains sends requests for help when the incoming packets exceed its capacity and sends requests for the termination of help it can handle all traffic. We set domain n_1 to be under attack and it communicates with other domains to seek help. We set the number of packets needed for every negotiation to 10 packets per pair. Figure 17(a) shows the traffic volume and the CoFence negotiation traffic on domain n_1 under attack. We recorded that the amount of communication overhead between domain n_1 and other nodes is about 180 packets in total, which is a trivial number compared to the normal traffic.

We should also mention that the CoFence communication can be done through a "separate" reserved network channel. This can be done through the NFV network slicing. This way we can ensure that the CoFence collaboration is possible under DDoS attacks. Figure 17(b) show an example of the CoFence's network slicing.

2) Computational Resources and Financial Cost: One main benefit of CoFence is that it allows domain to utilize trustable external resources to help combat DDoS attacks. This way domains can handle much larger volume of DDoS attacks. Compared to the cloud-based DDoS protection proxy approaches [1], [4], CoFence is a network resource exchange network that can be free of financial cost, while the cloud-based DDoS mitigation services typically induces high cost. Regarding the computational resources, each domain has full control on the amount of resource it is willing to share. The shared resource shall not exceed the spare resource it has so that it does not negatively impact its native operation.

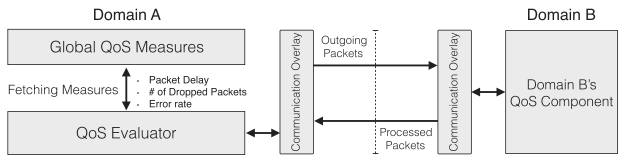

Fig. 18. Quality of Service (QoS) component architecture.

B. Service Agreement and Quality of Service

Since domains provide services to other domains, they need to ensure that other domains have agreeed to the same terms. Domains should contract with others to receive service. Service contracts can be customized for two domains or it may be a same agreement among all domains [7], [15]. Therefore, before domains start interacting with each other, they reach agreement on how they are going to provide service.

Quality of Service (QoS) is another important metric for the transport of traffic with special requirements and can be a part of the service agreement [21]. Network QoS can be influenced by different factors, both architectural and technical. Architectural factors include reliability and availability of service, latency, etc. Technical factors include scalability, effectiveness, and maintainability. Many things can happen to traffic packets as they travel from a domain to other domains and can result in a set of problems. For example, a domain should serve a specific amount of packets or packet processing delay should be less than a specific time otherwise packets drop. The agreement can also guarantee the packet dropping probability and/or error rate [25], [31]. Domains that provide service to others should guarantee a level of processing quality for an incoming traffic flow from other domains. In order to address this challenge we can apply networking Quality of Service standards [27]. Figure 18 shows our designed QoS component embedded into CoFence's architecture.

Our designed QoS architecture has two main components: Global QoS Measures and QoS Evaluator. Technically, perceived quality of service measures depend on multiple factors which include the designing of collaborative networks and networking such as packet delay, number of dropped packets, and error rate in terms of packet processing. Considering these measures, our QoS component evaluates the quality of received service from other domains and update the credits. The measured QoS has a direct impact on domains' credits. If a domain provides high quality service, it receives a higher credit from other domains.

C. Data Privacy and Protection

The CoFence model is mainly based on interactions and collaborations of different domains. One of the main concerns of collaborating with other domains is how to make sure that their network traffic is protected. More specifically, processing a domain's packets by other domains can raise data privacy concerns. This can result in identity theft and private information leakage. Since our model is proposed to protect domains against SYN Flood attack, the only thing that other domains can have access during collaboration is SYN packets and these packets do not contain any sensitive information causing private information leakage. Therefore, this is not a concern for our model.

D. Scalability

Since there might be new domains joining the collaborative network in order to serve/receive service the network will grow by time. Scalability can thus be a crucial challenge for CoFence. In our model, we define the scalability as the ability of handling a growing amount of incoming packets to domains and the potential of the collaborative network to be expanded in order to accommodate that growth. Therefore, we need to make CoFence scalable, so that domains can collaborate regardless of how big the network and the amount of incoming packets are. In order to address such challenge, we need to use networking technology solutions (hardware-based offload and network technologies) focused on eliminating potential network bottlenecks associated with network packet processing [10], [23].

E. Robustness

CoFence and its credit evaluation model can effectively improve network collaboration and protect networks against DDoS attacks. However, the credit evaluation itself may become the target of attacks and be compromised. In this section, we describe common attacks and provide defense mechanisms against them.

1) Domain Server's Vulnerabilities: can raise some critical security and privacy concerns. Among all network security vulnerabilities insecure network architecture and unprotected communication lines (communication overlay component) are the most important ones. To address such vulnerabilities we can define a set of security requirements as a part of service agreement and new domains should fulfill the requirements before they join the network. These requirements can include careful system maintenance (e.g. applying software patches), deployment (e.g. the use of proper firewalls, access controls and encryption equipment) and auditing (both during development and throughout the deployment lifecycle) to protect the network against possible vulnerabilities.

2) Collusion Attacks: it occurs when a group of domains (malicious ones) cooperate together to do a set of harmful attacks to other domains. There are two main collusion attacks. The first scenario is one in which a group of malicious domains lie to a normal domain (let us call it domain A) that requests help. For example, if the incoming packet flow from domain A to malicious ones is not from a DDoS attack, they report it as an attack and vice versa. The solution for this attack can be verifying the reports from helper domains. The second potential attack is when colluding malicious domains do not serve a specific domain in case of receiving help request from that domain. This way the colluding domains can shut down the victim domain. This attack scenario can be addressed by a well prepared service agreement and quality of service measurement.

3) Betrayal Attacks: it occurs when a normal domain (a domain with high credit) suddenly turns into a malicious one and starts sending false help requests to all neighbor domains. The collaboration performance can be influenced by this attack. We already addressed this attack in our model. The amount of impact that a domain can have on other domains in the network has a direct relation with its gained credit through its interaction with others. In our model, domains can not gain a high credit unless they behave safe and normal for a long time. On the other hand, they also lose their credit if they change their behavior from normal to malicious.

4) Tamper Attack: it occur when a user manually manipulates their local data and credits in order to mislead the system or other users. In CoFence, the only way that a domain gain credit is to help others when they are under attack. The credit of a domain is evaluated and stored by other domains. Thus, because domains do not have access to their credits they cannot manually manipulate their credits. However, if a domain (let's call it A) manipulates another domains' credit (let's call it B) on its domain, it will only change its resource allocation to other domains without benefiting itself. Therefore, there is no incentive for A to change the credits of other domains. This indicates that CoFence is resilient against Tamper attack.

F. Network Interoperability

Since different domains can use different infrastructure and software, it can cause inconsistency. In order to establish a fully effective collaborative network, domains should be able to interact with each other regardless of their hardware heterogeneity. Our model can use a network *interoperability standard* to achieve network interoperability, so that domains can communicate (requesting/receiving help) [16], [24]. The standards include application, data, and management interoperability.

VII. 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 requesters so that the resource is distributed efficiently, fairly, and with incentives. In order to make the resource allocation optimized, we utilized the *stakelberg* game model for collaboration. To make the collaboration fair, we proposed a QoS framework under which domain networks should agree with. 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] (Dec. 2016). *Arbor DDoS Detection and Protection*. [Online]. Available: http://security.arbornet works.com/protection

[2] (Dec. 2016). *Atlas Q2 2015 Update*. [Online]. Available: http://www.slideshare.net/Arbor_Networks/atlas-q2-2015finall

[3] (Dec. 2016). *Biggest Internet Attack in History Threatens Critical Systems*. [Online]. Available: http://www.ibtimes.co.uk/biggest-internet-attack-history-threatens-critical-infrastructure-450969

[4] (Dec. 2016). *Prolexic Routed: A DoS and DDoS Protection for Internet-Facing Applications, Network, Data Center Infrastructure and Network Bandwidth Into the Data Center*. [Online]. Available: https://www.akamai.com/us/en/solutions/products/cloud-security/prolexic-routed.jsp

[5] (Dec. 2016). *The Simpy Simulator: Event Discrete Simulation for Python*. [Online]. Available: https://simpy.readthedocs.io

[6] (Dec. 2016). *Xyzbooter Ltd*. [Online]. Available: https://booter.xyz/

[7] A. Abdelsadiq, C. Molina-Jimenez, and S. A. Shrivastava, "A high-level model-checking tool for verifying service agreements," in *Proc. IEEE 6th Int. Symp. Service Oriented Syst. Eng. (SOSE)*, Dec. 2011, pp. 297–304.

[8] N. Beigi-Mohammadi, C. Barna, M. Shtern, H. Khazaei, and M. Litoiu, "CAAMP: Completely automated DDoS attack mitigation platform in hybrid clouds," in *Proc. 12th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2016, pp. 136–143.

[9] V. DeMiguel and H. Xu, "A stochastic multiple-leader Stackelberg model: Analysis, computation, and application," *Oper. Res.*, vol. 57, no. 5, pp. 1220–1235, 2009.

[10] W. Ding, W. Qi, J. Wang, and B. Chen, "OpenSCaaS: An open service chain as a service platform toward the integration of SDN and NFV," *IEEE Netw.*, vol. 29, no. 3, pp. 30–35, May 2015.

[11] C. Cui *et al.*, "Network functions virtualisation (NFV)," in *Proc. SDN OpenFlow World Congr.*, Düsseldorf, Germany, 2014, pp. 1–20.

[12] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic DDoS defense," in *Proc. 24th USENIX Secur. Symp. (USENIX Security)*, Washington, DC, USA, 2015, pp. 817–832.

[13] P. Ferguson and D. Senie. *Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing (BCP 38)*, accessed on Dec. 2016. [Online]. Available: http://tools.ietf.org/html/rfc2827

[14] C. J. Fung and B. A. McCormick, "VGuard: A distributed denial of service attack mitigation method using network function virtualization," in *Proc. 11th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2015, pp. 64–70.

[15] R. Greenwell, X. Liu, and K. Chalmers, "Semantic description of cloud service agreements," in *Proc. Sci. Inf. Conf. (SAI)*, Jul. 2015, pp. 823–831.

[16] S. Grid, C.-P. S. P. Office, Energy, and E. Division. (Sep. 2014). *NIST Framework and RoadMap for Smart Grid Interoperability Standards, Release 3.0*. [Online]. Available: http://www.nist.gov/smartgrid/upload/NIST-SP-1108r3.pdf

[17] F. Guenane, M. Nogueira, and A. Serhrouchni, "DDOS mitigation cloud-based service," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, vol. 1. Aug. 2015, pp. 1363–1368.

[18] 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 *Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jun. 2016, pp. 431–436.

[19] A. John and T. Sivakumar, "DDos: Survey of traceback methods," *Int. J. Recent Trends Eng. Technol.*, vol. 1, no. 2, pp. 241–245, Nov. 2009.

[20] S. Khandelwal. *602 Gbps! This May Have Been the Largest DDoS Attack in History*, accessed on Dec. 2016. [Online]. Available: http://thehackernews.com/2016/01/biggest-ddos-attack.html

[21] K. Kritikos *et al.*, "A survey on service quality description," *ACM Comput. Surv.*, vol. 46, no. 1, p. 1, Jul. 2013.

[22] S. Lim, J. Ha, H. Kim, Y. Kim, and S. A. Yang, "A SDN-oriented DDoS blocking scheme for botnet-based attacks," in *Proc. 6th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2014, pp. 63–68.

[23] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[24] Office of the National Coordinator for Smart Grid Interoperability. (Jan. 2010). *NIST Framework and RoadMap for Smart Grid Interoperability Standards, Release 1.0*. [Online]. Available: http://www.nist.gov/publicaffairs/releases/upload/smartgrid-interoperability-final.pdf

[25] G. Pibiri, C. McGoldrick, and M. Huggard, "Expected quality of service (eQoS) a network metric for capturing end-user experience," in *Proc. IFIP Wireless Days (WD)*, Nov. 2012, pp. 1–6.

[26] J. J. Santanna *et al.*, "Booters—An analysis of DDoS-as-a-service attacks," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 243–251.

[27] J. Soldatos, E. Vayias, and G. Kormentzas, "On the building blocks of quality of service in heterogeneous IP networks," *IEEE Commun. Surveys Tuts.*, vol. 7, no. 1, pp. 69–88, 1st Quart., 2005.

[28] R. Vogt, J. Aycock, and M. J. Jacobson, Jr., "Army of botnets," in *Proc. NDSS*, 2007, pp. 111–123.

[29] H. Wang, C. Jin, and K. G. Shin, "Defense against spoofed IP traffic using hop-count filtering," *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, pp. 40–53, Feb. 2007.

[30] H. Wang, D. Zhang, and K. G. Shin, "Detecting SYN flooding attacks," in *Proc. 21st Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 3. Jun. 2002, pp. 1530–1539.

[31] K. Xiong, "Secure network bandwidth provisioning for quality of services (QoS) guarantees," in *Proc. IEEE 12th Int. Conf. Ubiquitous Intell. Comput., IEEE 12th Int. Conf. Auto. Trusted Comput., IEEE 15th Int. Conf. Scalable Comput. Commun. Assoc. Workshops (UIC-ATC-ScalCom)*, Aug. 2015, pp. 444–451.

[32] Y. Xu and Y. Liu, "DDoS attack detection under SDN context," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[33] K.-Y. Chen, A. R. Junuthula, I. K. Siddhrau, Y. Xu, and H. J. Chao, "SDNShield: Towards more comprehensive defense against DDoS attacks on SDN control plane," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Oct. 2016, pp. 28–36.

[34] A. Yaar, A. Perrig, and D. A. Song, "Pi: A path identification mechanism to defend against DDoS attacks," in *Proc. Symp. Secur. Privacy*, May 2003, pp. 93–107.

Bahman Rashidi received the master's degree in computer engineering from the Iran University of Science and Technology, Tehran, Iran, in 2014. He is currently pursuing the Ph.D. degree in computer science with Virginia Commonwealth University. His research interests include distributed systems, mobile systems, and privacy issues in smartphone devices. Being the top student in the program, he graduated with the Distinguished Master's Student of the year in research award for two consecutive years in 2012 and 2013. He was a recipient of the Outstanding Early-Career Student Researcher Award from the VCU Computer Science Department in 2015.

Carol Fung received the bachelor's degree and the master's degree in computer science from the University of Manitoba, Canada, and the Ph.D. degree in computer science from the University of Waterloo, Canada. Her research interests include collaborative intrusion detection networks, social networks, security issues in mobile networks and medical systems, security issues in next generation networking, and machine learning in intrusion detection. She was a recipient of the Young Professional Award in IEEE/IFIP IM 2015, the Alumni Gold Medal of the University of Waterloo in 2013, the best dissertation awards in IM2013, the best student paper award in CNSM2011, and the best paper award in IM2009.

Elisa Bertino was a Professor and the Department Head with the Department of Computer Science and Communication, University of Milan. She has been a Visiting Researcher with the IBM Research Laboratory, Microelectronics and Computer Technology Corporation, Rutgers University, and Telcordia Technologies. She is currently a Professor of Computer Science with Purdue University. She is a fellow of the ACM. She received the IEEE Computer Society 2002 Technical Achievement Award and the IEEE Computer Society 2005 Kanai Award. She serves as the EiC of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING.