An OpenFlow User-Switch Remapping Approach for DDoS Defense

Qiang Wei[1,2], Zehui Wu[1,2], Kalei Ren[1,2] and Qingxian Wang[1,2]
[1] State Key Laboratory of Mathematical Engineering and Advanced Computing
Zhengzhou, 450000 - China
[e-mail: wuzehui2010@foxmail.com]
[2] China National Digital Switching System Engineering and Technological Research Center
Zhengzhou, 450000 – China
[e-mail: wuzehui2010@foxmail.com]
*Corresponding author: Zehui Wu

Received January 13, 2016; revised April 19, 2016; revised June 13, 2016; accepted July 22, 2016; published September 30, 2016

Abstract

DDoS attacks have had a devastating effect on the Internet, which can cause millions of dollars of damage within hours or even minutes. In this paper we propose a practical dynamic defense approach that overcomes the shortage of static defense mechanisms. Our approach employs a group of SDN-based proxy switches to relay data flow between users and servers. By substituting backup proxy switches for attacked ones and reassigning suspect users onto the new proxy switches, innocent users are isolated and saved from malicious attackers through a sequence of remapping process. In order to improve the speed of attacker segregation, we have designed and implemented an efficient greedy algorithm which has been demonstrated to have little influence on legitimate traffic. Simulations, which were then performed with the open source controller *Ry*u, show that our approach is effective in alleviating DDoS attacks and quarantining the attackers by numerable remapping process. The simulations also demonstrate that our dynamic defense imposes little effect on legitimate users, and the overhead introduced by remapping procedure is acceptable.

Keywords: Cyber security, dynamic defense, software defined network, DDoS attack, greedy algorithm

This research was supported by National High Technology Research and Development Program of China [2012AA012902] and National Science Fund for Distinguished Young Scholars [61402526].

http://dx.doi.org/10.3837/tiis.2016.09.027 **ISSN : 1976-7277**

1. Introduction

Distributed Denial-of-Service (DDoS) attacks are derived from DoS attacks, by expanding one-to-one relationship attack mode to many-to-one relationship attack model, which pose an immense threat on the Internet [1]. Comparing with the first quarter of 2014, the number of reported DDoS attacks increased by 100% in the same period of 2015. The industries of game and finance are among worst victims. And since each attack could cause damage as much as $50 thousand, victims have to pay for ransom to avoid the financial loss. So there is a pressing need for practical DDoS defense approaches.

A certain amount of mechanisms and approaches have been proposed in the past few decades to stop or alleviate DDoS attacks [2].

Filtering-based approaches [3, 4, 5, 6] intend to block attack traffic from the protected network by deploying kinds of filters. However, this kind of defense assumes that the attack traffic and the legitimate traffic have some different features. Thus, if the clever attackers use botnet which could hide the difference between attack traffic and legitimate traffic to attack the target network, filtering-based approaches would fail.

Capability-based mechanisms change the passivity of the above filtering-based approaches and employ a different philosophy that requires senders to obtain the receivers' explicit permission before transmitting packets to them. Capability-based mechanisms concentrate on the use of resource by the receivers to prevent DDoS attacks, which are practical active defense approaches [7, 8]. However, these mechanisms are excessively dependent on the processing capability of the routers and the other underlying physical facilities.

To eliminate underlying physical facilities constraints, secure overlay networks, such as Tor [9] have been applied to supply traffic filtering and tolerance enhancement [10, 11]. However, the above methods still belong to static approaches which could be break or bypass by some skillful attackers.

To address the shortage of the above static defense approaches, other mechanisms use hidden layer to mitigate DDoS attacks [12, 13, 14]. The DDoS attack flow is transmitted by the dynamically reallocated hidden nodes between the attacker and the target. In [14], researchers propose a moving target DDoS defense by using hidden proxy nodes. However, since the IP addresses of these hidden proxy nodes are static, clever attackers could send probing packets to get the topology of the hidden proxy nodes. Besides, it is costly to upgrade the existing physical network facilities, which is necessary for these hidden layer approaches.

Based on previously proposed DDoS defense mechanisms, we propose a user-switch remapping approach which uses the dynamic programmable property of SDN to provide dynamic defense. The result of experiment shows that the implementation is able to mitigate DDoS attack traffic as well as segregate the attackers effectively.

The main contributions of our work are:

(1) We first propose a user-switch remapping approach which uses SDN to provide dynamic defense and attacker segregation. Static defending approaches have shortages of depending on the support of extra functionalities from Internet facilities or requiring virtual networks to defend large attacks. Instead of traditional approaches, our dynamic defense employs SDN-based controller to dynamically remap user-switch connection and by switch rotation we could mitigate the DDoS traffic as well as segregate the attackers.

(2) We propose a method to provide an uninterrupted service for legitimate traffic when the network is under DDoS attack. Control and forwarding separation is the essential propriety of SDN, by which we could dynamically generate new routing paths to transmit the legitimate traffic, thus insuring the continuous service for legitimate users.

(3) We implement our dynamic defense using dynamic programmable property of SDN. SDN provides dynamic programming interfaces through OpenFlow northbound APIs, and we are the first to implement this dynamic defense approach practically.

The rest of the paper is organized as follows. Related work is discussed in Section 2.We describe our threat model and assumptions in Section 3. We show the architecture of our approach and discuss the procedure of DDoS dynamic defense in Section 4. We present our user-switch remapping approach and its implemental algorithm, as well as the complexity analysis in Section 5. We give the implementation and evaluation of our work in Section 6. We give our conclusion in Section 7.

2. Related Work

Currently, researches on the detection and defense of DDoS attacks based on SDN are mainly divided into the following aspects:

Luke McHale et al. [15] proposed a flow re-classification method to determine whether the SDN network is under DDoS attack through analyzing the traffic from the same source in a SDN network. Since the attack traffic is assumed to come from the same source, the method is not capable of detecting and preventing DDoS attacks.

Braga et al. [16] proposed a lightweight DDoS attack detection method. By extracting the six-tuple associated with the DDoS attack, which is then dimensionally reduced with self-organizing map (SOM), the DDoS attacks are eventually detected.

Paul Smith et al. [17] implemented a dynamic management framework based on SDN, which can defend a variety of security threats, including DDoS attack and worm propagation. However, the framework can only deal with small-scale DDoS attacks right now.

Seungwon Shin et al. [18] combined source address authentication, connection migration and other methods to defend DoS attacks in SDN network. However, the method cannot defend DoS attack from application layer, nor can it defend attacks based on UDP or ICMP protocol.

Crossfire attack is a DDoS attack aiming at links rather than servers, which becomes more difficult to detect and defend compared with the traditional DDoS attacks. Dimitrios Gkounis et al. [19] implemented a prototype system based on SDN network. It proves that

the dynamic configuration characteristic of the SDN network can be used to defense crossfire attack, but the measurement precision and overhead are not tested yet, so its practicability remains to be investigated.

Moreno Ambrosin et al. [20] proposed a method called LineSwitch. LineSwitch acts for all TCP connection from given IP. Once it detects the SYN flood attack, it will add the corresponding IP to the blacklist and prevent all TCP packets from this IP. This method can effectively prevent the SYN flood attack against SDN network with little expense, and dynamically adapt to the changes of network structure. But it cannot defend DDoS attack too.

Compared with the above approaches, the dynamic DDoS defense method proposed in this paper changes the static character of traditional defense methods, makes targets of DDoS attack change dynamically, and increases the difficulty of the attack, thus reducing the attack surface.

3. Threat Model and Assumptions

We focus on DDoS defense with SDN, and our threat model consists of the following two types: (a) Targeting web servers deployed in SDN. As for this threat model, we employ SDN to protect the web servers. (b) Targeting SDN-based controllers or switches. With regard to this model, we use SDN to mitigate the attack traffic to safeguard itself.

Simultaneously, our first hypothesis is the accessibility of a SDN-based controller with enough computing power and network bandwidth to instantiate a large number of proxy switches. The second assumption is that the secure channel between controllers and switches is reliable, which would not be attacked by DDoS. Besides, the flow inside the SDN includes both the attack traffic and the legitimate traffic, since the attackers and legitimate users are in the same network. The last assumption is that there exists intrusion detection system in the target network to detect DDoS attacks, because we focus on DDoS defense rather than detection. To simulate DDoS detection, we deploy a traffic detector on each proxy switch, and if the traffic outnumbers our threshold, we assume the occurrence of DDoS attacks.

4. Dynamic DDoS Defense Architecture

Our approach employs a group of SDN-based proxy switches to relay data flow between users and servers. By substituting backup proxy switches for attacked ones and reassigning suspect users onto the new proxy switches, innocent users are isolated and saved from malicious attackers through a sequence of remapping process. We will discuss the remapping algorithm in Section 5. In the following sections, we first give an overview of our dynamic DDoS defense approach, and then present the main content of our work.

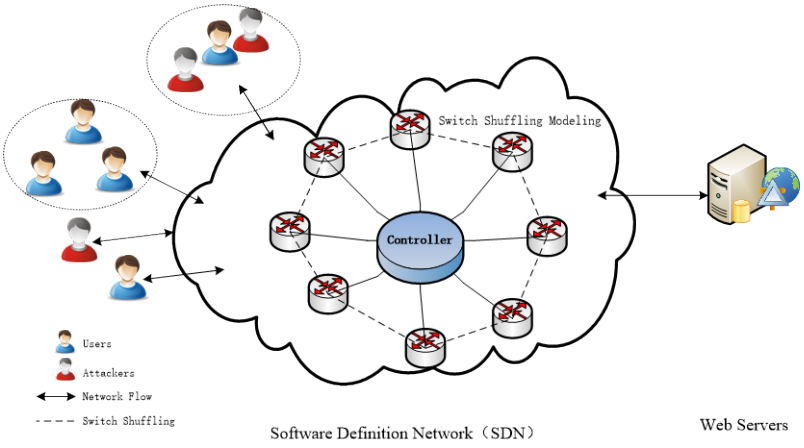


Fig. 1. Overview of Dynamic DDoS Defense Architecture

4.1 Architecture Overview

Fig. 1 shows the whole architecture of our dynamic defense approach which consists of three inter-connected components: users, software defined network, and web servers.

As we can see from **Fig. 1**, the network traffic that users send to the web servers is transmitted by the SDN which provides protection for the target web servers. Under normal state of the network, users visit the web servers by their DNS or IP addresses, and the visiting traffic is forwarded by the SDN-based switches through the routing paths configured by the SDN-based controllers. However, under abnormal network state, if the traffic detectors deployed on each proxy switch have detected DDoS attacks, the SDN-based controllers replace attacked proxy switches with backup proxy switches and reassign suspect users onto the new proxy switches so as to segregate attackers. And the legitimate traffic will still be forwarded by the original proxy switches, which could produce little influence on legitimate users.

4.2 Switch Layering

We divide SDN-based switches into two layers, the proxy layer and the hidden layer, as shown in **Fig. 2**. The proxy layer is constituted by proxy switches which receive the network flow sent by users directly, and will be used in our switch remapping process to segregate the attackers. The hidden layer employs SDN-based switches to relay packets that transmitted from the proxy layer according to the routing paths configured by SDN-based controllers. We deploy the traffic detectors on proxy switches, and set the threshold of them according to the simulation requirements through SDN-based controllers. When DDoS attacks happen, the controllers employ user-switch remapping approach to dynamically configure the proxy switches for segregating attackers and legitimate users. After that, the legitimate traffic will be forwarded to the hidden layer, and then relayed to the destinations.

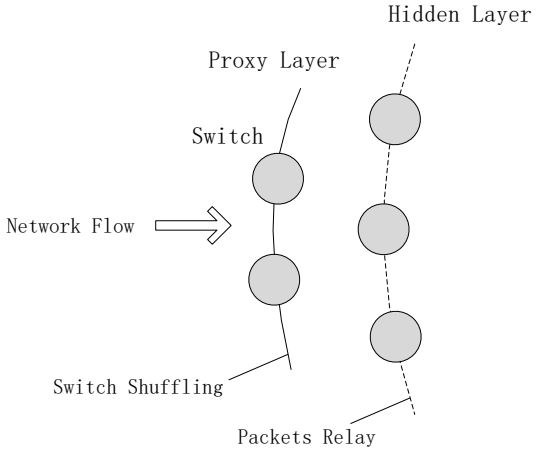


Fig. 2. Hierarchical Classification of SDN-based Switches

4.3 Remapping Approach

Before presenting our Remapping algorithm, we first give a simple example to illustrate the procedure of our user-switch remapping process. In [12], researchers discussed how to filter the insiders using proxy shuffling method, which inspires us to propose our approach. However, their shuffling method needs the support from an authentication server and hidden proxy nodes. We modified and improved their method by employing SDN-based switches and controllers. Our approach can be illustrated by **Fig. 3**.

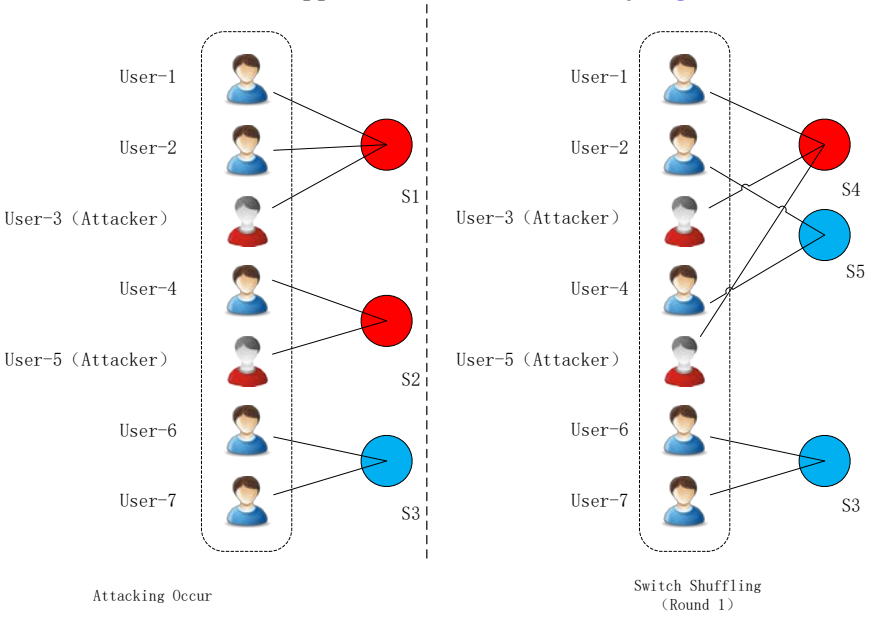


Fig. 3. A Simple Case of Switch Remapping

Initially, seven users (including attackers, denoted by (*user-1* ⋯ *user-7*) are randomly mapped to three proxy switches (denoted by *S1*, *S2* and *S3*), as suggested by the left figure: *user-1*, *user-2* and *user-3* are mapped to proxy switch *S1*, *user-4* and *user-5* are mapped to *S2*, and *user-6* and *user-7* are mapped to *S3*. In this example, *user-3* and *user-5* are attackers which bring an attack to proxy switch *S1* and *S2*. After detecting the attack, the controller reacts by replacing *S1* and *S2* with new proxy switches *S4* and *S5*. Users and attackers previously mapped to *S1* and *S2* are remapped to *S4* and *S5*. One possible mapping scheme maps *user-1*, *user-3* and *user-5* to *S4*, while mapping *user-2* and *user-4* to *S5* as suggested by the right figure. In this case, *S5* was not under attack and the users on *S5* are saved. In the opposite, *S4* remains attacked and its users are still suspected. The mapping from *user-6* and *user-7* to *S3* remains the same on the account of their escaping from the attack. As a result of this remapping, only *S4* will be involved in the next round of remapping.

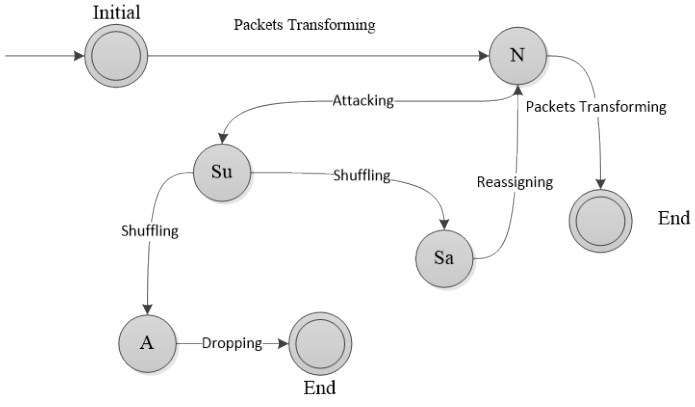


Fig. 4. Migration of Proxy Switch States

From the above discussion of this simple example, we know that there are four states of proxy switches: *Normal (N)*, *Attacked (A)*, *Suspect (Su)*, *Saved (Sa)*. The migration diagram of the four states is shown in **Fig. 4**. When the attack occurs, the switch is transferred from the normal state to the suspect state. After remapping process, the current state is transferred to other two states: saved and attacked, while the former state will be switched to normal state after routing redirection. Then the saved users and their packets will be forwarded by hidden layer, while the packets coming from attacked switches will be dropped directly in order to reduce the traffic caused by DDoS attack.

5. Switch Remapping Modeling

5.1 Problem Definition

To mitigate DDoS attacks as quickly as possible, and also to quarantine attackers over time, we have to design a remapping algorithm to distinguish and isolate most innocent users per

remapping process. Therefore, we provide the main definitions required in this algorithm as follows. First, we list all the used variables and their representation in **Table 1**.

Table 1. Variables and Rrepresentation

Variables	Representation
N_{sum}	the total number of users of the network
N_A	the number of attackers
N_{su}	the number of suspect users when attack happens
N_{sa}	the number of saved users after one round of remapping process
N_{us}	the number of unsaved users after one round of remapping process
$E(N_{sa})$	the expected value of N_{sa}
S	the total number of proxy switches
S_j	the number of users that connect to proxy switch j
p_j	the probability that proxy switch j is not under attack
$\vec{A}$	the remapping scheme of user-switch

Definition 1:

N_{sum} denotes the total number of users of the network. N_A denotes the number of attackers, and N_{su} denotes the number of suspect users when attack happens. N_{sa} denotes the number of saved users after one round of remapping process, and N_{us} denotes the number of unsaved users after one round of remapping process, which means this users are still suspect. So we have the following equations: $N_{sum} = N_A + N_{su}$ and $N_{su} = N_{sa} + N_{us}$.

We aim at calculating the expected value of N_{sa} (denoted as $E(N_{sa})$), and find a method that maximize the expected value given S available shuffling proxies.

Definition 2:

S denotes the total number of proxy switches, and S_j denotes the number of users that connect to proxy switch j. p_j is the probability that proxy switch j is not under attack. So we have the following equation: $p_j = \dfrac{C_{N_{sum}}^{N_A - S_j}}{C_{N_{sum}}^{N_A}}$.

According to definition 1 and 2, the expected value can be calculated by Equation (5.1).

$$E(N_{sa})=\sum_j^S p_j * S_j = \sum_j^S \frac{C_{N_{sum}}^{N_A-S_j}}{C_{N_{sum}}^{N_A}} * S_j \tag{5.1}$$

As a consequence of Equation (5.1), increasing the amount of proxy switches (denoted as S) also improves the whole size of user group which we want to save in each remapping process (denoted as $E(N_{sa})$) in each remapping process. The extreme situation is $S \geq N_{sum}$ while every user can be remapped to a proxy switch and it is obviously to segregate attackers. However, in most cases, the user population would greatly outnumber the proxy switches $S << N_{sum}$. So the problem of calculating $E(N_{sa})$ can be transferred to the following optimization problem:

Given the total number of proxy switch S, the vector $\vec{A}$ represents the remapping scheme of user-switch, for example, $A_1 = 2$ means proxy switch S_1 connects with two users. Hence, Equation (5.1) can be transferred to calculating $\sum_{j=1}^{S} \overrightarrow{A_j} = N$, which is an optimization/maximization problem. And in our work, we adopt a greedy approach described in Section 5.3 to produce a quick and near-optimal solution.

5.2 Estimating the Number of Attackers

From the above analysis we know that the number of attackers is a variable quantity, which has direct influence on the user-switch mapping scheme and affect the calculating of $E(N_{sa})$. So we use a maximum-likelihood estimation (MLE) as an estimate to the number of attackers before we give our greedy algorithm.

We use X to denote the number of proxy switches that are not under attack. And in a particular attack we assume $X = m$, according to the inclusion-exclusion principle of balls-and-urns model [21] we have the following Equation (5.2).

$$\begin{aligned} p(X=m) = {} & p(X \geq m) - C_{m+1}^{m} p(X \geq (m+1)) \\ & + C_{m+2}^{m} p(X \geq (m+2)) - \\ & \cdots + (-1)^{S-m} C_{S}^{m} p(X \geq S) \end{aligned} \tag{5.2}$$

We employ the set $\mathrm{U} = \{u_1, u_2, ..., u_m\}$ to represent the proxy switches that are not under attack, and $\sum_{j=1}^{m} S_j$ denotes the summation of all the proxy switches that are not under attack. So the number of attacker can be denoted as $N_{sum} - \sum_{j=1}^{m} S_j$, and we have Equation (5.3)

$$p(X \geq m) = \frac{\sum_{U}^{m} C_{N_{sum} - \sum_{j=1}^{m} S_j}^{N_A}}{C_{N_{sum}}^{N_A}} \tag{5.3}$$

Hence, according to Equation (5.2) and Equation (5.3), we could obtain the value of N_A.

With the value of N_A, we could give our greedy algorithm of user-switch remapping approach.

5.3 Greedy Algorithm for Switch Remapping

In this portion, we discuss the solution of user-switch remapping approach using greedy algorithm, and analysis the method of reducing the computational complexity to linear algorithm by employing Stirling Approximation. In [12], we know that when $N_A \leq S$, evenly distributing all users to all proxy switches is optimal. However, the result is opposite ($N_A >> S$) when refer to DDoS attacks. Hence, we employ greedy algorithm instead of evenly distributing method.

It is obviously that if each proxy switch connects to fewer users, then the probability that proxy switches are under attack decrease as well. But the number of saved users falls along with the probability after each round of remapping process. Therefore, the appropriate number of users that should connect to each proxy switch should be the value that maximized the following equation.

$$E(S_j) = \frac{C_{N_{sum} - S_j}^{N_A}}{C_{N_{sum}}^{N_A}} * S_j \tag{5.4}$$

The detailed procedure of greedy algorithm used in computing the user-switch mapping process is shown in Algorithm 1. Because the value of N_A affects the optimal value of S_j, for a specific N_A, all probable values of S_j could be enumerated and the parameter k which maximized $p_j * S_j$ will then be chosen. This procedure is described in *MaxSwitch* of Algorithm 1.

This computation will exit when any of the following three prerequisites is met: (a) There are more proxy switches left than users, under this condition each users will be mapped to a proxy switch; (b) There is only one proxy switch left, under this condition all remaining users will be allocated to a proxy switch; and (c) The expected number of remaining attackers is rounded to 0, under this condition there are no attackers, and all remaining users will be evenly allocated.

Algorithm 1 is a recursive algorithm, which calls *MaxSwitch()* to obtain the scheme used in remapping process according to Equation (5.4). And the variable *proxyAssign* denotes the number of proxy switch when remapping with parameter. *proxRem*, *userRem* and *attackRem* all represent the remaining number of proxy switch, users and attackers respectively.

The enumeration approach can find the optimal value of S_j in $\Theta(N_{sum} \cdot N_A)$ time complexity. Therefore, as N_{sum} and N_A get larger, the running time of the subroutine *MaxSwitch* will become notably longer. To address this problem, we employ Stirling Approximation $n! \approx \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$ to reduce the computational complexity to linear.

We assume $N_A << N_{sum}$, then we get $E(S_j) = \left(1 - \frac{N_A}{N_{sum}}\right)^{S_j} * S_j$.

Let $S_j = \frac{N_{sum} * x}{N_A}$, we have

$$E(S_j) = \left(1 - \frac{N_A}{N_{sum}}\right)^{\frac{N_{sum} * x}{N_A}} * \frac{N_{sum} * x}{N_A} = e^{-x} \frac{N_{sum} * x}{N_A} \tag{5.5}$$

After getting the derivation of Equation (5.5), we could apply approximation $\lim_{n \to \infty}(1 - 1/n)^n \approx e^{-1}$ on this equation. Then, we have $\frac{\partial E(S_j)}{\partial x} = \frac{e^{-x} * N_{sum}}{N_A} * (1 - x)$. From Equation (5.5), the derivation of $E(S_j) = 0$ if and only if $x = 1$. That is, $x = 1$ maximizes $E(S_j)$. Therefore, $E(S_j) = \frac{N_{sum}}{N_A * e}$, and the function *MaxSwitch()* described in algorithm 1 could be cut off, and the final computational complexity is reduced to $\Theta(N_A)$.

Algorithm 1 GreedyRemap // Greedy algorithm for computing user-to-proxy switch remapping process

Input: user, attacker, proxySwitch
Output: numRemap
1: **def** GreedyRemap(*user, attacker, proxySwitch*)
2: **if** *user* $\leq$ *proxySwitch* **then**
3: Allocate 1 proxy switch to user;
4: **else if** *proxySwitch* == 1 **then**
5: Allocate all the users to the proxy switch;
6: **else if** *attacker* == 0 **then**
7: Evenly distribute user over proxy switch;
8: **else**
9: *k* = MaxSwitch(*user, 0, user-1, attacker*);
10: *proxyAssign* = $\lfloor user / k \rfloor$;
11: **if** *proxAssign* $\geq$ proxySwitch **then**
12: *proxAssign* = *proxySwitch* – 1;
13: **end if**
14: *proxRem* = *proxySwitch* – *proxAssign*;
15: *userRem* = *user* – *proxAssign***k*;
16: *attackRem* = Shuffle(*attacker***userRem*/*user*);
17: GreedyShuffle(*userRem, attackRem, proxRem*);
18: **end if**
19**: def** MaxSwitch(*user, left, right, attacker*)
20: *max* = 0;
21: *maxSwitch* = 0;
22: **for** *i* = *left* → *right* **do**
23: *saved* = $C_{user-i}^{attacker} / C_{user}^{attacker}$;
24: **if** *saved* > *max* **then**
25: *max* = *saved*;
26: *maxSwitch* = I;
27: **end if**
28: **end for**
29: **return** *maxSwitch*;

6. Simulation and Evaluation

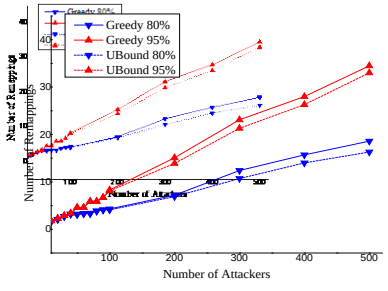
In this section, we evaluate the performance of user-switch remapping approach and its DDoS dynamic defense capability. In the former evaluation, we implemente user-switch remapping algorithm on *MATLAB*, and simulate the number of users, attackers and proxy switches. But for clearer comparison, these factors are kept constant.

When evaluating DDoS dynamic defense capabiliy, we implemente the DDoS dynamic defense module in open source SDN controller *Ryu*[22] and open source OpenFlow switch *Open vSwitch*[23], and the module is loaded when Ryu starts. Ryu is one of the dominant commercial SDN controllers, and is implemented with Python which is in

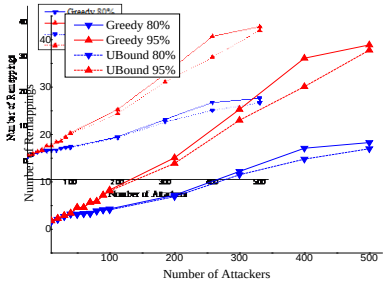
conformity with the programming language of our DDoS dynamic densese module. *Open vSwitch* is implemented on software, which allows us to create switches by simply copying the switch virtual machines. *Ryu* and *Open vSwitch* help us reduce the difficulty and complexity of the evaluation.

6.1 Performance Evaluation

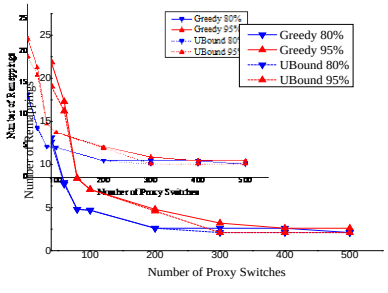
In this part, we implemente user-switch remapping approach on *MATLAB* and simulate the number of users, attackers and proxy switches. These factors do not need to remain constant, but we keep them constant during each simulation in order to make the comparison easier. In each simulation, the percentage of users and attackers are randomly selected from all users by *Mersenne twister*[24], our random number generator. We also suppose that the attackers only attack the proxy switches which they are connected to, and there are sufficient bandwidth for attackers to compromise an attacked proxy switch.



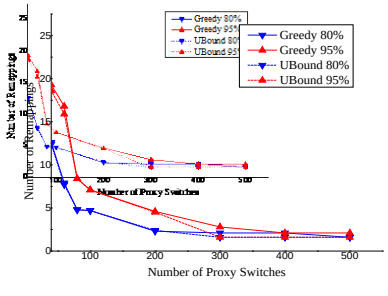
(a) Varing the number of Attackers under 10K clients, 100 proxy switches



(b) Varying the number of attackers under 100K clients, 100 proxy switches



(c) Varying the number of proxy switches under 10K clients, 100 attackers



(d) Varying the number of proxy switches under 100K clients, 100 attackers

Fig. 5. The relationship among the number of remappings, attackers and proxy switches

Fig. 5 shows the relationship among the number of remappings, the number of attackers and the number of proxy switches. The solid lines represent the relationship among the number of remappings, attackers and proxy switches when algorithm 1 segerates 80% and 95% of the attackers. And the dotted lines represent the theoretical upper bound(caculated by Formula 4.1). The simulation results show that when the percentage of segeragated attackers remain constant, the number of remappings and proxy switches increase as the number of attackers climbs up.

In **Fig. 5(a)** and **Fig. 5(b)**, the number of users and proxy switches remain constanst, and the number of attackers increases from 10 to 500. In **Fig. 5(c)** and **Fig. 5(d)**, the number of users is the same as that in **Fig. 5(a)** and **Fig. 5(b)**, the number of attackers is 100, and the number of proxy switches increases from 40 to 500. Every simulation runs 10 times and we use the average value as the result.

Fig. 5 shows that the result of algorithm 1 is close to the theoretical upper bound, so our method can be regarded as the near-optimal approach. As showed in **Fig. 5(a)** and **Fig. 5(b)**, when the percentage of segeragated attackers remain constant, the number of remappings and the number of attackers increase as linear function approximately. **Fig. 5(c)** and **Fig. 5(d)** show that the number of remappings decreases as the number of proxy switches increases, and when the number of proxy switches is fewer than that of attackers, the change is obvious. However, when the number of proxy switches is more than that of attackers, the change becomes gentle.

6.2 DDoS Dynamic Defense Capability Evaluation

We have developed a prototype of user-switch remapping approach, using Python with approximately 3000 lines. The approach is developed as an application module of *Ryu*, with *Ryu* 3.6 and OpenFlow 1.1.0, and the module is loaded when *Ryu* starts. The proxy switches are implemented on *Open vSwitch* with the version of 2.3.1(OVS for short in **Fig.6**).

In order to evaluate user-switch remapping approach, we operate the prototype under the simulation environment in **Fig.6**. In **Fig.6**, OVS-1 to OVS-5 are proxy switches, and OVS-6 to OVS-10 are forwarding switches. *Ryu* is the management and configuration center of all switches, and runs on a machine with 2 Intel Xeon 2.13GHz CPUs and 64GB memory. We have also developed a traffic monitoring application on *Ryu* to obtain the traffic distribution information of the network periodically. In **Fig.6**, Agg1 to Agg4 are legitimate flows, and Agg5 is an attack flow. The attack flow launches DoS attack by using TCP SYN. All of the five aggregates are generated by a Kali virtual machine.

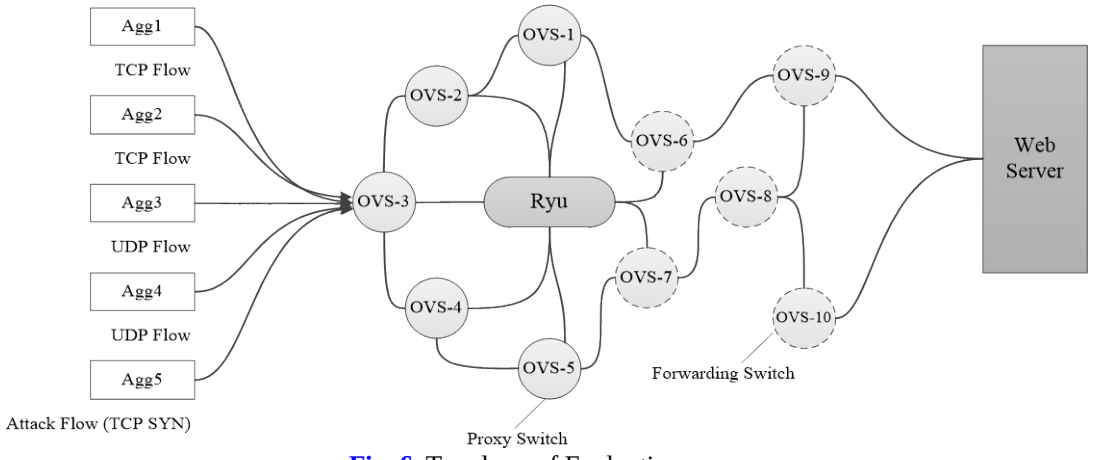


Fig. 6. Topology of Evaluation

A. DDoS Defense capability

Fig. 7 is the results of simulation with(right) and without(left) user-switch remapping approach. Agg1 to Agg4 are legitimate flows, and Agg5 is an attack flow. Agg5 starts to increase at t=13, and decrease at t=30. In **Fig. 7**(top left), we can see that without our approach, the high-bandwidth aggregate consumes the majority of all bandwidth. At the bottom left of **Fig. 7**, the packets' drop rate of legitimate flows increases when Agg5 starts to increase. **Fig. 7**(right) is the same simulation result with user-switch remapping approach. When the packets' drop rate is over the configured value 10%, our approach recognizes the attackers, and reduces the packets' drop rate by sufficiently isolating attackers. In **Fig. 7**(right), we can see that the lasting time, bandwith and packets' drop rate of the attack flow have all decreased.

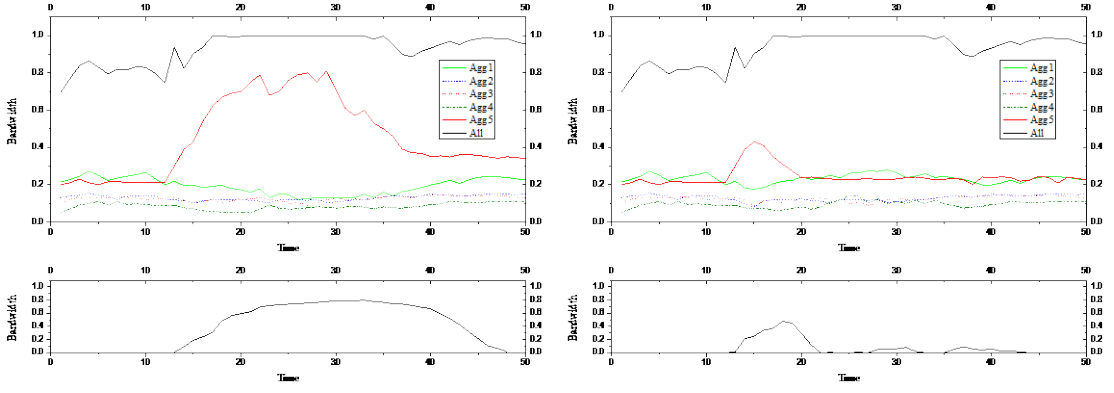


Fig. 7. The simulation with(right) and without(left) user-switch remapping approach

Fig. 8 shows the simulation results of relationship between the attack flow rate and the recovery time of the network. In the simulation, the attacker launches attack flows of

different rate to attack the proxy switches. In **Fig. 8**, we can see that the attack flow rate and the recovery time of the network increase as logarithmic function approximately. Our dynamic defense approach is able to filter attackers, so the attackers have only little effect on the network after being isolated. After the time 1200s, increasing attack flows have almost no effect on the network, so we can infer that most of the attackers have been segeragated at the time 1200s.

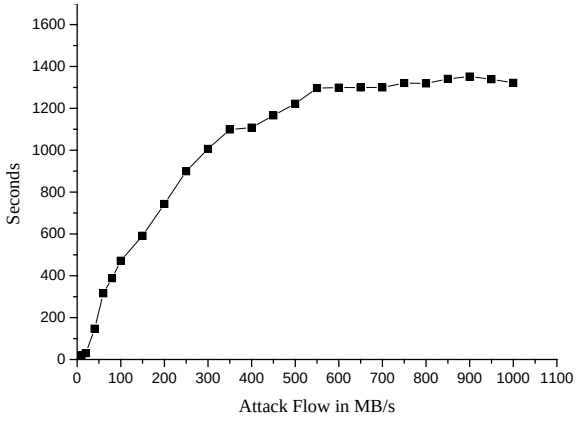
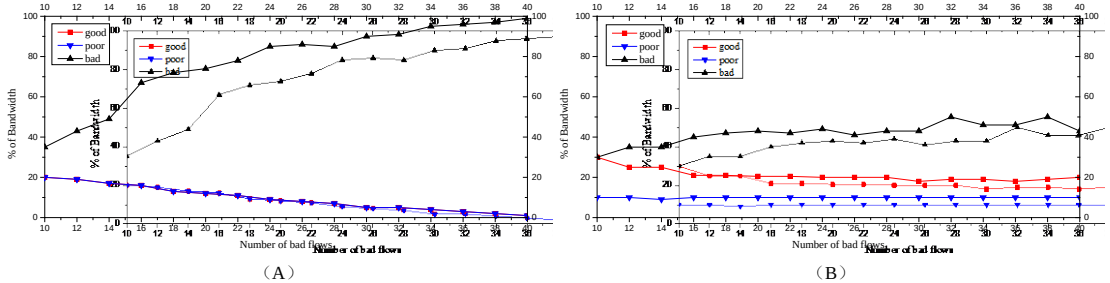


Fig. 8. The result of attack flow stress test

B. Effect on Legitimate Flows

Fig. 9 show the simulation results of bandwith consumptions of attack flows(bad), legitimate flows(poor) and legitimate flows affected by attack flows(poor). **Fig. 9**(left) does no t use user-switch remapping approach and **Fig. 9**(right) uses the aproach. In **Fig. 9**(left), the bad flows captures the majority of the band width, while the good and the poor traffic suffer. By using user-switch remapping approach, we can protect the good and the poor traffic from the bad, because the approach can differentiate the two kinds of traffic.



（A） （B）

Fig. 9. The bandwidth consumption with(right) and without(left) user-switch remapping approach

C. Overhead

The experimental results presented above indicate that using user-switch remapping approach would be effective in defense, but the communication overhead between users

and the application server increases when implementing the approach because of the proxy-based communication relay the user-switch remapping.

10 geographical locations which are far from each other are selected to conduct the test, and they form 5 testing links. The latency for both direct and relayed communications were measured for each of the 5 links, and the results are shown in **Table 2**. Mean round trip times (RTT) were obtained by bouncing short TCP messages back and forth between the end hosts of each flow 100 times. For the majority of the results, the overhead for mean RTT ranges from 15% to 30% which is acceptable.

Table 2. Latency overhead introduced by proxy indirection

	Direct	Indirect			
	RTT	**Mean RTT**	**Overhead**	**Max RTT**	**Overhead**
1	65ms	105ms	61.54%	143ms	120%
2	88ms	102ms	15.91%	129ms	46.59%
3	86ms	105ms	22.09%	132ms	53.49%
4	92ms	113ms	22.83%	131ms	42.39%
5	85ms	108ms	27.06%	119ms	40%

7. Conclusion

DDoS attacks have always been an outstanding problem for many years in cyber security. And many websites choose CDN network, or deploy their services in the cloud to facilitate the DDoS attacks [25, 26]. However, this defense method is only the confrontation of resources between attackers and defenders, which could not radically prevent DDoS attacks. In this paper we propose a practical dynamic defense approach by employing the dynamic programmable property of SDN, thus mitigating DDoS attack traffic as well as segregating the attackers effectively. In our future work, we plan to study multi-dimensional DDoS defense mechanism that combines SDN and cloud together.

Acknowledgements

This work was supported by the National High Technology Research and Development Program of China (Grant NO. 2012AA012902) and National Science Fund for Distinguished Young Scholars (Grant NO. 61402526).

References

[1] J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,” *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39-53, 2004. Article (CrossRef Link).

[2] M. Abliz, “Internet denial of service attacks and defense mechanisms,” *University of Pittsburgh, Department of Computer Science, Technical Report*, pp. 1-50, 2011.

[3] Y. Chen, X. Ma and X. Wu, "DDoS detection algorithm based on preprocessing network Traffic Predicted Method and Chaos Theory," *IEEE Communications Letters*, vol. 17, no. 5, pp. 1052-1054, 2013. Article (CrossRef Link).
[4] H. S. Kang and S. R. Kim, "sShield: small DDoS defense system using RIP-based traffic deflection in autonomous system," *The Journal of Supercomputing*, vol. 67, no. 3, pp. 820-836, 2014. Article (CrossRef Link).
[5] X. Liu, X. Yang and Y. Lu, "To filter or to authorize: Network-layer DoS defense against multimillion-node botnets," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 195-206, 2008. Article (CrossRef Link).
[6] S. M. Lee, D. S. Kim, J. H. Lee and et al., "Detection of DDoS attacks using optimized traffic matrix," *Computers & Mathematics with Applications*, vol. 63, no. 2, pp. 501-510, 2012. Article (CrossRef Link).
[7] X. Yang, D. Wetherall and T. Anderson, "TVA: a DoS-limiting network architecture," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1267-1280, 2008. Article (CrossRef Link).
[8] X. Liu, X. Yang and Y. Xia, "NetFence: preventing internet denial of service from inside out," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 255-266, 2011. Article (CrossRef Link).
[9] R. Dingledine, N. Mathewson and P. Syverson, "Tor: The second-generation onion router," in *Proc. of the 13th Usenix Security Symposium*, pp.28-39, 2004.
[10] H. Luo, Y. Lin and H. Zhang and et al., "Preventing DDoS attacks by identifier/locator separation," *IEEE Network*, vol. 27, no. 6, pp. 60-65, 2013. Article (CrossRef Link).
[11] Z. Anwar and A. W. Malik, "Can a DDoS attack meltdown my data center? A simulation study and defense strategies," *IEEE Communications Letters*, vol. 18, no. 7, pp. 1175-1178, 2014. Article (CrossRef Link).
[12] V. Kambhampati, C. Papadopolous and D. Massey, "Epiphany: A location hiding architecture for protecting critical services from DDoS attacks," in *Proc. of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 1-12, 2012. Article (CrossRef Link).
[13] C. Fachkha, E. Bou-Harb and M. Debbabi, "Inferring distributed reflection denial of service attacks from darknet," *Computer Communications*, vol. 62, pp. 59-71, 2015. Article (CrossRef Link).
[14] H. Wang, Q. Jia, F. Dan and et al., "A moving target DDoS defense mechanism," *Computer Communications*, vol. 46, no. 6, pp. 10-21, 2014. Article (CrossRef Link).
[15] L. Mchale, J. Case, P. V. Gratz and et al., "Stochastic pre-classification for SDN data plane matching," in *Proc. of IEEE 22nd International Conference on Network Protocols*, pp. 596-602, 2014. Article (CrossRef Link).
[16] R. Braga, E. Mota and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proc. of IEEE 35th Conference on Local Computer Networks (LCN)*, pp. 408-415, 2010. Article (CrossRef Link).
[17] P. Smith, A. Schaeffer-Filho, D. Hutchison and et al., "Management patterns: SDN-enabled network resilience management," in *Proc. of IEEE Network Operations and Management Symposium (NOMS)*, pp. 1-9, 2014. Article (CrossRef Link).
[18] S. Shin, V. Yegneswaran, P. Porras and et al., "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proc. of ACM SIGSAC conference on Computer & communications security*, pp. 413-424, 2013. Article (CrossRef Link).

[19] D. Gkounis, V. Kotronis and X. Dimitropoulos, "Towards defeating the crossfire attack using SDN," *arXiv preprint arXiv*, pp. 12-22, 2014.
[20] M. Ambrosin, M. Conti, F. De Gaspari and et al., "Lineswitch: Efficiently managing switch flow in software-defined networking while effectively tackling dos attacks," in *Proc. of the 10th ACM Symposium on Information, Computer and Communications Security, ACM*, pp. 639-644, 2015. Article (CrossRef Link).
[21] D. A. Sprott, "Urn models and their application—an approach to modern discrete probability theory," *Technometrics*, vol. 20, no. 4, pp. 501-501, 20(4), 1978.
[22] RYU Homepage. https://osrg.github.io/ryu/, (Access on 2016-04-11).
[23] Open vSwitch. Homepage.http://openvswitch.org/, (Access on 2016-04-11).
[24] M. Twister, "A 623-Dimensionally equidistributed uniform pseudorandom number generator-matsumoto," *Nishimura ACM Trans on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3-30, 1998.
[25] F. Al-Haidari, M. Sqalli, K. Salah, "Evaluation of the impact of EDoS attacks against cloud computing services," *Arabian Journal for Science and Engineering*, pp. 1-13, 2014. Article (CrossRef Link).
[26] B. Wang, Y. Zheng, W. Lou and et al., "DDoS attack protection in the Era of cloud computing and software-defined networking," in *Proc. of IEEE 22nd International Conference on Network Protocols*, pp. 624-629, 2014. Article (CrossRef Link).

Qiang Wei born in 1979, Master's supervisor. His main research interests include ICS security, software vulnerability analysis and SDN network security.

Zehui Wu born in 1988, doctoral candidate. His research interest includes SDN network security.

Kailei Ren born in 1992, Master Degree Candidate. His research interest includes SDN network security.

Qingxian Wang born in 1960, PhD supervisor. His main research interest includes network security.