DDoS Attack Detection under SDN Context

Yang Xu and Yong Liu
Department of Electrical and Computer Engineering
New York University
Brooklyn, New York, 11201
Email: yx388@nyu.edu, yongliu@nyu.edu

***Abstract*—Software Defined Networking (SDN) has recently emerged as a new network management platform. The centralized control architecture presents many new opportunities. Among the network management tasks, measurement is one of the most important and challenging one. Researchers have proposed many solutions to better utilize SDN for network measurement. Among them, how to detect Distributed Denial-of-Services (DDoS) quickly and precisely is a very challenging problem. In this paper, we propose methods to detect DDoS attacks leveraging on SDN's flow monitoring capability. Our methods utilize measurement resources available in the whole SDN network to adaptively balance the coverage and granularity of attack detection. Through simulations we demonstrate that our methods can quickly locate potential DDoS victims and attackers by using a constrained number of flow monitoring rules.**

I. Introduction

Recently, Software Defined Networking (SDN) has emerged as a new network management platform, which draws lots of attentions from both academia and industry. The centralized control platform fundamentally changed the traditional distributed network management paradigm. SDN offers plenty of opportunities for new network management methods. Network measurement is among one of the most important tasks in network management. Through SDN, network admins can flexibly install any flow rule in any controlled switch as long as it has additional Ternary Content-addressable memory (TCAM) [1]. In the current Openflow specification [2], there are packet count and byte count fields in each flow rule. If a flow rule is matched by a packet, the packet count and byte count fields of that rule will increase accordingly. Using this feature, we can install some rules specifically for network measurement in SDN switches.

Distributed denial-of-service (DDoS) attacks [3] make online services unavailable by overwhelming victims with traffic from multiple attackers. As more and more businesses migrate their operations online, DDoS attacks have caused significant financial losses [4]. There are reports showing that the frequency of DDoS attacks has become higher and higher recently [5]. Thus, how to effectively and quickly detect DDoS attacks is one of the most important problems for network measurement. Since DDoS attackers are by nature distributed across the whole network, coordinated network-wide monitoring is necessary for efficient DDoS detection. For timely detection and mitigation, DDoS detection should also react quickly to the onset of traffic anomaly. SDN central controller can quickly install and adapt measurement rules on all switches in a coordinated fashion. This makes SDN an ideal platform for DDoS detection. In addition, after DDoS attackers are detected, SDN controller can immediately install blocking rules to drop attack traffic for prompt DDoS attack mitigation.

Current DDoS attacks have various forms, e.g., consumption of computational resources, disruption of configuration information, etc. For different types of DDoS attacks, there are different DDoS detection methods [3]. In this study, we focus on how to detect large volume DDoS attacks, in which more than thousands of attackers transfer packets to a victim to overwhelm the victim's access bandwidth. Thus, large traffic rate is one important feature for this type of DDoS attacks. Besides, previous research [6] has showed that traffic rate deviation/asymmetry is another important feature of DDoS attacks. In a DDoS attack, usually there will be huge rate difference between flows coming into a victim server and flows going out of the victim. If we only consider the traffic rate without observing the rate asymmetry between two directions, we may falsely tag legitimate large-rate flows, e.g., data transfers between data centers, as DDoS attack flows.

SDN switches utilize TCAM as their lookup memory, because of its fast lookup speed. But since TCAM is very expensive and very power-consuming, the TCAM size for each SDN-enabled switch is very limited. Contemporary SDN switch can only store around $3,000$ rules. It is impossible to record flow statistics for the whole network at the finest IP pair granularity. Thus, to utilize SDN to detect DDoS attacks, we should address the following challenges:

1) *How to capture the traffic rate feature as well as the traffic rate deviation/asymmetry feature to achieve high detection precision?*
2) *How to collaboratively utilize limited TCAM available on all switches to monitor the whole network?*

To address the first challenge, for each suspected victim IP range, we have to install a pair of rules to capture both the flows going into the range and the flows coming out of the range. And we have to make sure the range granularities of the pair of rules are consistent. To address the second challenge, we coordinate monitor rule placement on all switches to efficiently utilize all TCAM entries available in the whole network to maximize the coverage and minimize the granularity of detection. We further propose an adaptive procedure to dynamically zoom in the potential victim and attacker IP ranges and zoom out the normal IP ranges. Furthermore, we develop a *Sequential Method* as well as a *Concurrent Method* to do victim and attacker detection. Finally, we evaluate our proposed methods through simulations to demonstrate their advantages as well as the potential weaknesses.

The rest of the paper is organized as follows. Section II covers the related work. Section III gives an overview of our system. We present our sequential and concurrent detection methods in Section IV and Section V respectively. We compare the two methods in Section VI. Section VII introduces the classification method we used in our detection. Section VIII presents our experiment results. The paper is concluded in Section IX.

II. Related Work

Characteristics of DDoS attacks have been widely studied. Researchers have proposed various methods, e.g., covariance analysis, cluster analysis, wavelets, to detect attacks [3], [7], [8], [9], [10]. In network measurement, some papers studied how to spread the load of measurement across the whole network [11], [12], [13]. Our DDoS detection work builds on top of the existing SDN proposals [14], [2], [15], [16]. There are some work on how to do measurement under SDN environment [13], [17], [18], [6], [19]. Among them, [18] discusses the tradeoff between detection accuracy and resource consumption. [17] proposes a measurement framework based on sketch. [19] studies how to detect heavy hitters on a single switch. [13] proposes an adaptive flow counting method for anomaly detection. They install rules across the network and adaptively change rule granularity to do anomaly detection. In our method, we also utilize all switches in the whole network. But our methods capture the asymmetry feature of DDoS attacks to achieve higher detection accuracy. Industry has also proposed SDN based DDoS attack detection, e.g., the Defense4All solution [20] in OpenDayLight [21]. The basic idea is to collect statistics from some locations in the network to identify the anomaly traffic. For those suspicious traffic, they will be diverted to a scrubbing center to do further detection and flow cleaning. The drawback of this method is that it introduces additional delay to the traffic, which will degrade user quality-of-experience (QoE) of delay-sensitive services. [6] also studies how to utilize SDN to do DDoS detection. But they assume that the installed rules could always reach the finest granularity, which cannot hold in reality due to the limited TCAM sizes. We use Self Organizing Mapping (SOM) [22] as our DDoS attack detection classifier. Some work discussing how to utilize SOM to do anomaly or intrusion detection can be found in [23], [24].

III. System Overview

Generally, DDoS attack defense consists of two procedures: victim detection and post-detection. We will describe each of the procedures in details in the following.

The aim of victim detection procedure is to quickly and correctly detect DDoS attack victims. As stated in the introduction, the key to correctly identify DDoS attack victims is to jointly consider the flow volume feature and the flow rate asymmetry feature. To capture these two features, for any potential victim IP we should have measurement rules to record the total flow rate coming to this IP as well as the total flow rate going out of this IP. Since TCAM size is limited, we cannot install the above measurement rules for all individual IP addresses in the whole network. Thus, initially we can only observe flow volume and flow rate asymmetry for large IP ranges. If our captured features for these large IP ranges show potential DDoS attacks, we will adaptively zoom in to find the precise victim IP address. Otherwise, we will adaptively zoom out to save TCAM size for detecting other potential victim IP ranges. Due to TCAM size limit, we may never find the precise IP addresses of victims if TCAM sizes are not large enough. The victim detection procedure finishes with the smallest possible IP ranges containing potential victims.

Another procedure is post-detection procedure. There are two ways to react to the detected DDoS attacks. One way is to do passive processing, e.g., contacting the user of the victim IP and asking him to migrate his normal service to a new IP. But usually the migration process may take some time and it also wastes resources of the victim server. Another way is to do active processing, e.g., network admins find the attacker IP addresses and install rules in Openflow switches to drop packets from attackers to the victim. Through this way, it saves network resources and doesn't affect normal operation of the victim user. Like the victim detection procedure, detection of attackers is conducted adaptively. This is the post-detection procedure we will study in this paper. And we call this procedure as attacker detection procedure.

The above two procedures could be done either sequentially or simultaneously, which results in two different detection methods. We will discuss these two different methods in more details in the following two sections.

IV. Sequential Method

The general work flow of the Sequential Method is shown in Figure 1. We start with the initial rule partition/placement, followed by the victim detection. After victims are identified, attacker detection procedure is conducted. We will describe the components in detail in this section.

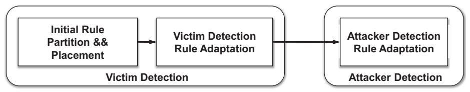

Initial Rule Partition && Placement
Victim Detection Rule Adaptation
Attacker Detection Rule Adaptation
Victim Detection
Attacker Detection

Fig. 1: Work Flow of Sequential Detection

A. Victim Detection

1) Initial Rule Placement: Initially, if we don't have any prior knowledge about whether a DDoS attack has happened nor what IP ranges contain the victim servers, the DDoS detection system need to monitor all IPs in the system. To make detection fast and accurate, we want to make the IP range granularity for monitoring as small as possible. Given the limited flow table sizes on all switches, our design objective is to minimize the maximum granularity among all monitored IP ranges. In other words, we don't want to have a large number of IP addresses monitored by a single rule. Besides, for each monitored IP range, we need to measure the total rate of traffic going into all IP addresses in that range, as well as the total rate of traffic going out from all IP addressed in that range, to capture the flow rate asymmetry feature.

Following the SDN rule definition, each IP range is defined using the common prefix of all addresses in that range. Secondly, given a source IP range R^s and a destination IP range R^t, we want routing paths between all possible source and destination pairs between the two ranges $\{\langle s,t\rangle, \forall s \in R^s, \forall t \in R^t\}$, go through the same sequence of SDN switches in the network. To measure the flow rate asymmetry, we additionally want routing paths in the reverse direction $\{\langle t,s\rangle, \forall t \in R^t, \forall s \in R^s\}$ also follow the same sequence of SDN switches. We assume that we are working on a PoP-level topology, and all traffic between two PoP routers go through the same sequence of SDN switches. Then we can initialize the coarse IP ranges using the sets of IP addresses behind all PoP routers. In case the IP set behind a PoP router cannot be exactly summarized using any prefix matching rule, we will further divide the set into subsets until each subset can be exactly summarized using a prefix matching rule. If the routing assumption doesn't hold, we also need to further divide PoP level IP ranges until hosts in between each pair of ranges go through the same sequence of SDN switches. After this operation, we call each pair $\{R^s, R^t\}$ of source and destination IP ranges as a flow f.

After the initial rule setup, we determine the victim IP range granularity for measurement and the monitor rule placement for each flow. Generally, there are two ways to do rule management. The first way is to dedicate one measurement rule solely for one potential victim IP range. It means that if we want to split victim IP range, we only split the victim IP range and keep the attacker IP range the same as the initial partitioned ones generated by rule partition and placement method. For example, assuming that two flows $A \rightarrow B$ and $B \rightarrow A$ exist in the network, we can use rules $A \rightarrow B$ and $B \rightarrow A$ to observe the potential victim IP ranges A and B. If now we want to zoom in both victim IP range A and victim IP range B, the rule split method will generate rules $A_1 \rightarrow B$, $A_2 \rightarrow B$, $B \rightarrow A_1$ and $B \rightarrow A_2$ solely for detecting potential victims in IP range A. And rules $A \rightarrow B_1$, $A \rightarrow B_2$, $B_1 \rightarrow A$ and $B_2 \rightarrow A$ will be generated to solely detect potential victims in IP range B. The second way of rule organization method is that each rule is used to monitor both the source victim IP range and the destination victim IP range. It means that if source IP range and destination IP range are all suspected as victims, we not only split destination victim IP range but also split source victim IP range for one rule. For example, when we decide to split A and B using this method, we will generate $A_1 \rightarrow B_1$, $A_2 \rightarrow B_1$, $A_1 \rightarrow B_2$, $A_2 \rightarrow B_2$, $B_1 \rightarrow A_1$, $B_2 \rightarrow A_1$, $B_1 \rightarrow A_2$ and $B_2 \rightarrow A_2$ to monitor IP ranges A and B. If we want to further split both A and B into more ranges to do monitoring, the second way will generate more monitor rules compared to the first way. In general, if we want to split both A and B into k ranges to do monitoring, the first approach will generate $2k$ rules to monitor A and $2k$ rules to monitor B, a total of $4k$ rules; while the second approach will generate $2k^2$ rules to simultaneously monitor A and B, which is much larger than the first approach when $k > 2$. In the Sequential Method, we use the first rule organization method as it would generate fewer rules to detect victim. In our Concurrent Method, we use the second rule organization method, which we will discuss in details later.

Employing the first rule organization method, we use Algorithm 1 to do the initial rule partition and placement. We first set up one rule for each flow f to be monitored. [1] Let N_f be the number of monitor rules associated with flow f. Thus, initially we have $N_f = 1$ and place a single monitor rule for f on some switch on the path from IP range R^s to IP range R^t. Since we are focusing on the victim detection at this stage, we only worry about the granularity of the monitored victim IP range, which we define as its monitor granularity. First, we set the monitor granularity G to be a large value G_{upper}. We then try to reduce G iteratively. In each iteration we find the maximum granularity G_{max} among all the monitor IP range. Then, for each rule with the maximum monitor granularity G_{max}, we divide its monitor IP ranges into two halves, and replace it with two new monitor rules, one for each half of the victim IP range as described in the previous paragraph. As a result, the maximal granularity is reduced to $G_{max}/2$. After rule split, it is possible that the monitoring rules for different monitor IP ranges are the same. To save TCAM space, we will delete redundant rules. Then, we can update the number of monitor rules N_f for each flow. We will check whether the monitor rules can be placed to TCAM of all switches in the network. If there is a feasible placement, we try to do another iteration to refine the IP range granularity. Otherwise, the minimum maximum granularity is G_{max} and monitor rule placement is the one found in the previous iteration.

Algorithm 1 Initial Rule Partition and Placement

```
1: $F$ : set of flows; $R$ : set of monitor rules; $N_f$ : number of monitor rules for flow $f$;
2: Initialize monitor granularity to be $G_{upper}$
3: Initialize monitor rules $R$ for flows in $F$ at granularity $G_{upper}$; Calculate $N_f, \forall f \in F$;
4: **if** Placement($N_1, \cdots, N_F$) is not Feasible **then**
5:     Raise Error
6: **end if**
7: **while** 1 **do**
8:     $G_{max}$ = maximum victim range granularity of all rules in $R$
9:     $\hat{R} = R$
10:    **for** all rules $r \in \hat{R}$ with victim range granularity $G_{max}$ **do**
11:        Partition victim IP range into two halves;
12:        Replace $r$ with two new rules in $\hat{R}$;
13:    **end for**
14:    Remove redundant rules and Update $\{N_f, \forall f \in F\}$ based on $\hat{R}$;
15:    **if** Placement($N_1, \cdots, N_F$) is not Feasible **then**
16:        Return $G_{max}$ and the previous rule set $R$ and the associated allocation
17:        Break
18:    **end if**
19:    $R = \hat{R}$;
20: **end while**
```

2) Rule Placement Feasibility Check: The key question for Algorithm 1 is that given switch space requirement $\{N_f, \forall f \in F\}$, how to decide whether the monitor rule placement is feasible or not. This problem could be transformed to the classical maximum flow problem and then be solved by the Ford-Fulkerson algorithm [25]. In Figure 2, we illustrate the virtual graph for the rule placement check. Basically, the virtual graph consists of four types of nodes: start node S, terminate node T, flow nodes $\{f_i, i \in F\}$ and switch nodes

[1]Keeping in mind, to detect the rate asymmetry, for each IP range pair $\langle A, B\rangle$, we will have two flows to be monitored: $A \rightarrow B$ and $B \rightarrow A$.

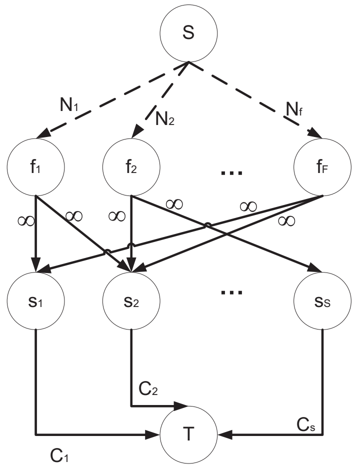


Fig. 2: Virtual Graph for Rule Allocation Check

$\{s_j, j \in S\}$. Start node S is connected to all flow nodes f_i and the link capacity between S and f_i is the required switch space N_i. Then, flow node f_i is connected to all switch nodes s_j that the flow traverses in the real network. And the capacity for those links are infinity. At last, a switch node s_j is connected to the terminate node T and the capacity between these two is the available TCAM rule space capacity C_j on switch s_j. Then, after getting this rule allocation network, the original problem becomes whether the maximum flow between S to T equals to $\sum_{f \in F} N_f$. This problem could be solved by utilizing the classic Ford-Fulkerson algorithm. The idea is that as long as there is a path from the start node to the terminate node, with available capacity on all edges in the path, we will send flow along one of these paths. We will try to find another path until no path is available. Then, after the program is terminated, we will know whether the placement is feasible or not. And if it is feasible, we can also check how many rule space each flow needs on each switch in the network.

3) Detection Rule Adaptation: After the initial rule placement, we will run DDoS attack detection algorithms to estimate the victim likelihood of each IP range in rule set. For the IP ranges having no sign of being attacked, we could use courser-grained rules to replace the finer-grained rules. For the IP ranges with high likelihood of under DDoS attack, we can utilize the available and/or newly released rule space to install finer-grained rules to monitor them. This adaptation is called spatial adaptation in [13]. Besides spatial adaptation, we could also change the rule fetching period and do temporal adaptation as stated in [13]. In this article, we mainly focus on spatial adaptation.

In Algorithm 2, we illustrate our adaptation algorithm to iteratively locate the potential victim IP ranges. For each destination IP range d in the current monitor rule set, we collect its traffic statistics and use DDoS attack victim classifier C_v (see details in Section VII) to calculate one value $C_v(d)$. If $C_v(d) = False$, d is not identified as a potential victim. If its sibling $Sib(d)$ is also not a victim[2], we can increase the observation IP range by replacing monitoring rules for d and $Sib(d)$ with monitoring rules for their common parent $Parent(d)$. If $C(d) = True$, d is a probable DDoS attack victim. We will try to refine the observation granularity for d by replacing monitoring rules for d with monitoring rules for its two children[3]. And if the granularity level is already the finest, we will do attacker detection. Each time we refine observation granularity, we need to utilize Algorithm 1 to decide whether the adaptation is feasible or not. In the current adaptation process, for each refinement or contraction process, we only increase or decrease the granularity by one prefix bit. We could also try larger adaptation steps. But if the adaptation step is too large, we may waste many TCAM sizes for false positive alarms.

Algorithm 2 Victim Detection Rule Adaptation (D)

```
1:  $D$ : set of victim ranges monitored by rules in $R$;
2:  **while** 1 **do**
3:      **for** $\forall d \in D$ **do**
4:          collect packet statistics $\mathcal{F}(d)$
5:          use victim classifier $C_v$ to calculate $C_v(d)$ to decide
            whether $d$ is under attack or not;
6:      **end for**
7:      **for** $d$ in $D$ **do**
8:          **if** $C_v(d) = False$ and $C_v(Sib(d)) = False$ **then**
9:              *Contraction:* add monitor rule for victim range
                $Parent(d)$, remove monitor rules for victim ranges
                $d$ and $Sib(d)$;
10:         **end if**
11:         **if** $C_v(d) = True$ and $G(d) \neq 32$ **then**
12:             *Refinement:* add monitor rules for the victim ranges
                $Child(d)$, remove monitor rule for victim range $d$;
13:         **end if**
14:     **end for**
15:     **if** Refined Rule Set Infeasible? **then**
16:         return list of victim ranges with $C_v(d) = True$
17:     **end if**
18: **end while**
```

B. Attacker Detection Procedure

After locating the potential victim IP ranges, we will start the attacker detection procedure. The attacker detection procedure also works in an adaptive fashion. Details of the algorithm are listed in Algorithm 3. For each range s, we will use attacker classifier $C_s(s)$ (see details in Section VII) to identify whether the attacker is within this ranges s. If yes, we will zoom in range s to explore the IP further. If the range is already at the finest level or the switches do not have enough TCAM space for the refined rule set , we will return the source IP or the source IP ranges directly.

V. Concurrent Method

In the previous sequential method, only after detecting the exact victim IP ranges, we will start the attacker detection. Another way of DDoS attack detection method is to do victim detection and attacker detection concurrently. The work flow of the Concurrent Method is shown in Figure 3. The basic idea

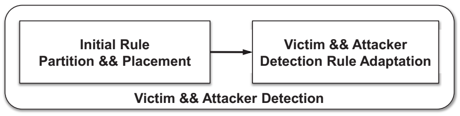
[2]A sibling range of a $/x$ range d is the other $/x$ range sharing the same $/(x-1)$ prefix with d.

[3]The two children of a $/x$ range d is the two IP ranges sharing the same $/x$ prefix with d and the $(x+1)$-th bit is 0 and 1 respectively.

Algorithm 3 Attacker Detection Rule Adaptation (S)

```
1: $S$ : set of IP ranges sending traffic to victim IP ranges;
2: while 1 do
3:     for $\forall s \in S$ do
4:         collect packet statistics features $\mathcal{F}(s)$
5:         use attacker classifier $C_s$ to calculate $C_s(s)$ to decide
           whether $s$ is an attacker or not;
6:     end for
7:     for $s$ in $S$ do
8:         if $C_s(s) = True$ then
9:             if $G(s) = 32$ then
10:                Return $s$
11:            else
12:                Refinement: add monitor rules for the attacker
                   ranges $Child(s)$, remove monitor rule for at-
                   tacker range $s$;
13:            end if
14:        end if
15:    end for
16:    if Refined Rule Set Infeasible? then
17:        return list of attacker ranges with $C_s(s) = True$
18:    end if
19: end while
```

is that if a victim IP range is being suspected under DDoS attack, not only we should refine the measurement granularity for the victim range, but also we should simultaneously refine the measurement granularity for all IP ranges that are classified as attackers of the victim, so that we can identify the potential attackers in the mean time. The simultaneous attacker and victim range refinement procedures are conducted in both the initial rule placement and the subsequent rule adaptation. Similar to the Sequential Method, the first step of the Concurrent Method is to do initial IP range separation to form flow f. Then we will do the initial rule partition and placement. Different from the Sequential Method, rules are not solely organized based on the victim IP ranges, and we use the second rule organization method as described in Section IV-A1. Our initial rule partition and placement are the same as in Algorithm 1. We also try to minimize the maximum monitoring granularity. But the rule refinement is different from the sequential method. Due to the simultaneously splitting of source and destination IP ranges, we will generate more monitoring rules. Therefore, it is expected that the final victim IP ranges obtained in the Concurrent Method will have coarser granularities than those obtained in the Sequential Method.



Initial Rule Partition && Placement
Victim && Attacker Detection Rule Adaptation
Victim && Attacker Detection

Fig. 3: Work Flow of Concurrent Detection

After the initial rule partition and placement, we will go to the concurrent rule adaptation process, as described in Algorithm 4. In the adaptation process, for any modification done for monitoring rule of $A \rightarrow B$, we will also do the corresponding modification to the monitoring rule of the reverse flow $B \rightarrow A$ to make sure that the new formed flows are still organized in pair. For rule $A \rightarrow B$, if at least one of A and B is identified as potential victim, we will do rule split, as depicted in Table I. If neither A nor B is a potential victim, we will try to do rule merge. For one rule t, *source sibling merge* means that we merge source range of t with its sibling range and *destination sibling merge* means that we merge destination range of t with its sibling range. Under these definitions, we could do rule merge as Table II states. The details of rule adaptation is listed in Algorithm 4. We will use classifier C_v to identify the potential victims, and for each victim d, we use classifier C_{sd} to identify the suspicious attackers of victim d. After the concurrent adaptation ends, we will find the potential victims as well as suspicious attackers attacking those victims.

TABLE I: Rule Split for rule $A \rightarrow B$

Condition	Rule Split
Neither A or B is victim	No Split
A, B are both victim	Split both A and B
B is victim, A is not attacker for B	Split B
B is victim, A is attacker for B	Split both A and B

TABLE II: Rule Merge for rule $A \rightarrow B$

Condition	Rule Merge
A or B is victim	No merge for A, B
A, B and their siblings are not victim	Source and destination sibling merge
A and its sibling are not victim	Source sibling merge
B and its sibling are not victim	Destination sibling merge

Algorithm 4 Concurrent Rule Adaptation R

```
1: $R$ : set of current monitor rules;
2: $D$ : set of victim ranges monitored by rules in $R$;
3: $S_d$ : set of attacker ranges monitored for victim ranges $d$
   by rules in $R$;
4: while 1 do
5:     for $\forall d \in D$ do
6:         collect packet statistics $\mathcal{F}(d)$
7:         use victim classifier $C_v$ to calculate $C_v(d)$ to decide
           whether $d$ is under attack or not;
8:     end for
9:     for $\forall s_d \in S_d$ where $C_v(d) = True$ do
10:        collect packet statistics features $\mathcal{F}(s_d)$
11:        use source classifier $C_{sd}$ to calculate $C_{sd}(s_d)$ to
           decide whether $s$ is an attacker for victim $d$ or not;
12:    end for
13:    for $r \in R$ do
14:        $s_d$=SourceRange(r); $d_d$=DestinationRange(r);
15:        if $C_v(s_d) = True$ or $C_v(d_d) = True$ then
16:            Do rule split according to Table I;
17:        else
18:            Do rule merge according to Table II;
19:        end if
20:    end for
21:    if Splited Rule Set Infeasible? then
22:        return list of potential victim ranges with correspond-
           ing suspicious attacker ranges
23:    end if
24: end while
```

VI. Comparison of Two Methods

In this section, we discuss the pros and cons of the above two proposed methods. Both methods can identify DDoS attack victims as well as attackers precisely given large enough TCAMs on all switches. If TCAM sizes on switches are constrained, both methods can only detect the victims and attackers at coarse IP range granularities.

At the initialization stage, both methods will try to make the IP observation ranges as small as possible. The Sequential Method can reach finer victim observation IP ranges, since the Concurrent Method use more TCAM space to monitor possible attacker IP ranges. Thus, if the objective is to quickly identify the victims, the Sequential Method is more preferable, as it can find the finest victim IP ranges under the TCAM size constraints. If the objective is to quickly find the victims as well as the attackers, the choice between the two depends on the TCAM capacities. If the TCAM capacities are pretty large, Concurrent Method is more preferable, as it can quickly find the victims along with the attackers. On the other hand, if the TCAM capacities are very constrained, it is likely that the Concurrent Method will exit at a very coarse observation granularity, while at least the Sequential Method can pinpoint the victims precisely. Thus, the preferred method under various conditions can be summarized as in Table III.

TABLE III: Method Selection under Various Conditions

TCAM Size Limit	Victim Detection	Attacker and Victim Detection
Small Size	Sequential	Sequential
Medium size	Sequential	Sequential or Concurrent
Large size	Sequential or Concurrent	Concurrent

VII. Classification Method

In the above two DDoS detection methods, we will use classifiers to detect victims as well as attackers. In this section, we will introduce how we do classification in more details.

A. Feature Selection

To do accurate classification, we first need to select proper features. For victim identification and attacker identification, we will use different feature representations. But as previously stated, we will capture both the flow volume feature as well as the asymmetry feature for both cases.

1) Victim Identification Features: We will select victim identification features based on the destination range. Assuming that we have traffic flow f_{ji} from range j to range i, and the number of IP addresses in range i is $Gra(i)$, then the victim feature for range i can be expressed as the followings:

1) *Packet Count per Destination ($\mathcal{P}$): describe the average number of packets to each destination IP in that range:*
   $\mathcal{P}_i = \sum_j Pkt_{ji}/Gra(i)$
2) *Byte Count per Destination ($\mathcal{B}$): describe the average number of bytes to each destination IP in that range:*
   $\mathcal{B}_i = \sum_j Byte_{ji}/Gra(i)$
3) *Packet Count Asymmetry per Destination ($\mathcal{PA}$): describe the average packet count asymmetry for each destination IP in that range:*
   $\mathcal{PA}_i = (\sum_j Pkt_{ji} - \sum_j Pkt_{ij})/Gra(i)$
4) *Byte Count Asymmetry per Destination ($\mathcal{BA}$): describe the average byte count asymmetry for each destination IP in that range:*
   $\mathcal{BA}_i = (\sum_j Byte_{ji} - \sum_j Byte_{ij})/Gra(i)$

Among the above four features, feature $\mathcal{P}$ and $\mathcal{B}$ only consider the volume flowing to a potential victim. When DDoS attack happens, values of these two features become very large. Feature $\mathcal{PA}$ and feature $\mathcal{BA}$ quantify the traffic asymmetry of DDoS attack. When DDoS attack happens, compared to the incoming flows, flows going out of a victim would be much smaller. Thus, we can also observe a large value for these two asymmetry features.

2) Attacker Identification Features: Using the previous notations, for a potential victim IP range j, the attacker identification features to identify whether IP range i has attackers for range j could be expressed as follows:

1) *Packet Count per Source ($\mathcal{P}$): describe the average number of packets from a host in IP range i to a host in victim IP range j:*
   $\mathcal{P}^a_{ij} = Pkt_{ij}/(Gra(i) * Gra(j))$
2) *Byte Count per Source ($\mathcal{B}$): describe the average number of bytes from a host in IP range i to a host in victim IP range j:*
   $\mathcal{B}^a_{ij} = Byte_{ij}/(Gra(i) * Gra(j))$
3) *Packet Count Asymmetry from Source ($\mathcal{PA}$): describe the average packet numbers asymmetry to victim IP range:*
   $\mathcal{PA}^a_{ij} = (Pkt_{ij} - Pkt_{ji})/(Gra(i) * Gra(j))$
4) *Byte Count Asymmetry from Source ($\mathcal{BA}$): describe the average bytes numbers asymmetry to victim IP range:*
   $\mathcal{BA}^a_{ij} = (Byte_{ij} - Byte_{ji})/(Gra(i) * Gra(j))$

Similar to the features used in victim identification, features $\mathcal{P}^a$ and $\mathcal{B}^a$ characterize the large volume feature while features $\mathcal{PA}^a$ and $\mathcal{BA}^a$ characterize the asymmetry feature of volume-based DDoS attacks.

B. Classifiers

After forming the above features, we will use Self Organizing Mapping (SOM) [22] as our classifier to do classification. SOM is an unsupervised artificial neural network that describes a mapping from a n-dimensional data space to a lower-dimensional map space. The result of the transformation is that data with similar statistical features would be gathered close to each other in the map space. Generally, the algorithm runs as follows:

1) *Randomize node's weights in the map space.*
2) *Choose one input vector I from the data space.*
3) *Calculate Euclidean distance between input vector and all map's nodes' weight vector. Find node with smallest distance, label this node as winner node.*
4) *Update the nodes in the neighborhood of winner node so that the Euclidean distance between their individual weight vector and the input vector becomes smaller.*

5) *repeat procedure 2 - 4 until weight vector has no significant change.*

Due to the space limit, we refer interested readers to [22] for implementation details of the SOM transformation process. Since SOM's training utilizes competitive learning, it is robust even when the training data are noisy. After the training, it could find the hidden relations between input data. Thus, we utilize this technique as our classifier. In our implementation, we will form different SOMs for victim detection and attacker detection.

VIII. Experiments

Due to the lack of access to complete traces of real DDoS attacks, we build a simulator to evaluate our methodology. Internet2's network topology [26] is employed as our simulated network. We select 19 different class C IP addresses as the existing IP addresses in the network. We assign each class C IP address to a PoP as their ingress and egress points. Flow routing among different PoPs follows the shortest path policy. In this section, we will present our simulation results.

A. Importance of Asymmetry Feature

Firstly, we want to study how the asymmetry feature affects the detection accuracy. In our experiments, we generate the following four different classes of transmissions:

1) *Attack Transmission ($\mathcal{A}$): flows from $10,000$ randomly picked source IPs to the victim IP, with sending rate from each source IP randomly distributed within ($30kbps$, $70kbps$) and receiving rate of each source IP randomly distributed within ($1kbps$, $4kbps$);*
2) *Normal Large Volume Transmission ($\mathcal{N}_1$): From one source IP to another destination IP, with sending rate within ($300mbps$, $700mbps$) and receiving rate randomly within ($300mbps$, $700mbps$);*
3) *Normal Small Asymmetry Transmission ($\mathcal{N}_2$): From one source IP to another destination IP, with sending rate randomly distributed within ($30kbps$, $70kbps$) and receiving rate randomly distributed within ($1kbps$, $4kbps$), the number of sources sending traffic to the same destination is no more than 100;*
4) *Normal Small Symmetry Transmission ($\mathcal{N}_3$): From one source IP to another destination IP, with sending rate randomly distributed within ($30kbps$, $70kbps$) and receiving rate randomly distributed within ($30kbps$, $70kbps$), number of sources sending traffic to the same destination is no more than 100.*

Then, we use the previously described classification methods to identify victims. For normal condition, we randomly mix $\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{N}_3$ to form normal transmission. For attack condition, we choose one IP as the victim and generate $\mathcal{A}$ for that IP. Besides $\mathcal{A}$, we also generate $\mathcal{N}_1$ for other IP pairs as normal background traffic. For each transmission, packet size is randomly selected in the range of ($500bytes$, $1500bytes$). In our experiments, we totally generate $5,000$ normal conditions and $5,000$ attack conditions. We use the features stated in Section VII-A1. We try two different feature combinations: two-feature combination $\{\mathcal{P},\mathcal{B}\}$ and four-feature combination $\{\mathcal{P}, \mathcal{B}, \mathcal{PA}, \mathcal{BA}\}$. The first feature combination doesn't consider the traffic asymmetry of DDoS attacks while the second combination does. For the victim IP selected, we calculate features at different prefix lengths. For example, given victim IP is IP and prefix length is a, the features at prefix a would be the features for IP range IP/a. After getting features at different prefix lengths, we will generate different SOMs at different prefix lengths. We choose 80% of the features from normal conditions and attack conditions as the training set and use the residual 20% features as the test set. Table IV shows the parameters of our SOM.

TABLE IV: SOM Parameters

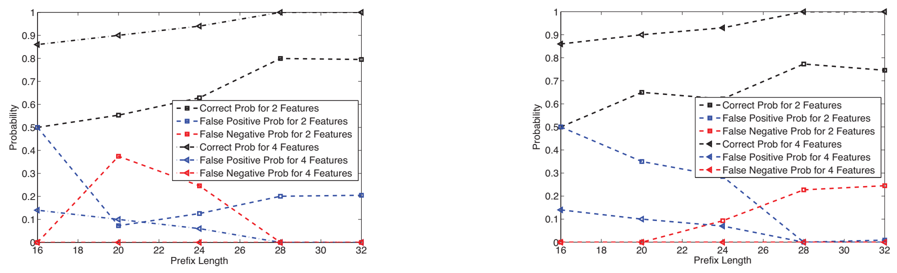
Parameter	Value
Neutron Dimension	20*20
Initial Learning Rate	0.05
Initial Neighbor Radius	20
Iteration Limit	1000

The victim detection performance of SOM classifier using different feature combinations under different IP prefix lengths is shown in Figure 4(a). In that figure, the first observation is that the four-feature combination performs much better than the two-feature combination, which means that the asymmetry feature is very important to achieve high detection accuracy. Another observation is that as the prefix length increases, SOM's detection accuracy also increases. This can be explained as that attack transmission $\mathcal{A}$ is hard to be differentiated from transmission $\mathcal{N}_2$ if multiple flows are monitored by a single rule. This confirms with our monitoring rule placement guideline: finer granularity leads to higher accuracy. The third observation is that SOM is very accurate at victim detection. When using the four-feature combination, even when the prefix length is 16, the detection accuracy is larger than 85%. The only error is false positive, which means that we won't miss any potential victim. We also test the attacker detection performance using SOM classifier as illustrated in Figure 4(b). It also shows the similar trend as in victim detection.

Summary: Simulation results demonstrate that capturing asymmetry feature is important to achieve high detection accuracy. Using suitable features, SOM classifier can achieve very accurate detection performance. The features obtained from finer granularity will make the detection more accurate.

B. Performance of Two Detection Methods

Then, we compare the performance of the previous two proposed detection methods. In our simulator, we set one IP as the DDoS attack victim and we randomly select N ($N > 500$) IPs as attackers for the victim. We implemented two proposed detection methods in the simulator. In our simulations, we assume that classifier can correctly identify victims or attackers in any IP range in any step. In the previous subsection, we did observe high classification accuracy when using suitable classifier and features. The only error that our classifier produces is few false positives. Given those false alarms, our detection methods will waste some extra switch TCAM to monitor the non-existing victims/attackers at some iteration. And our detection might completes with a coarser detection granularity due to the wasted switch TCAM. For the clarity of presentation, we don't introduce those errors in our simulations. In each adaptation round, we either expand an IP range to its direct parent range or split it into its direct children



(a) Victim Detection (b) Attacker Detection

Fig. 4: SOM Classifier Performance at Different Prefix Lengths.

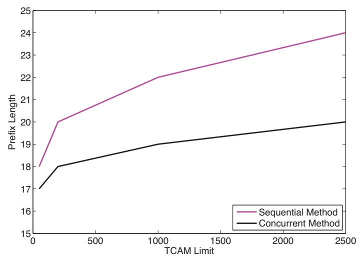


Fig. 5: Prefix Length for Observation after the Initial Rule Placement.

ranges. Then, we will vary switch TCAM sizes and attacker number to study the performance of the two methods.

Each method starts with initial rule placement. In this step, the aim is just to minimize the maximum observation IP range. In Figure 5, we show the prefix length of the observed IP range using the two methods under different TCAM sizes. As TCAM size increases, the prefix length of the monitored IP ranges becomes larger, i.e., the granularity of IP range observation becomes finer. Among the two methods, Concurrent Method has the coarser observation granularity, since it has to utilize some TCAM to simultaneously observe potential attackers.

After the initial rule placement, each method will adaptively change its rule ranges to detect victims as well as attackers. In Figure 6(a) and Figure 6(b), we list the maximum victim IP prefix length that the detection algorithms can reach and the number of iterations to get that IP prefix length. Using Sequential Method, we will always find the precise victim IP. As TCAM sizes become larger, Sequential Method can detect victim with fewer iterations. This is because that the initial rule placement already gives us a fine observation IP range to start with. In contrast, since Concurrent Method will install lots of rules to detect attackers simultaneously, the victim detection performance depends on the number of attackers. When the number of attackers is very large($>$500 attackers), victim detection only stops at a coarser IP range granularity. Under such condition, as TCAM sizes become larger, victim detection IP prefix length increases and the number of iterations to get this finer victim IP range also increases. Only under the condition with a small number of attackers ($=$ 500 attackers) and large TCAM sizes ($>$1000), we can find the precise victim IP. The number of iterations to find this precise victim IP would decrease as TCAM sizes become larger, due to the same reason for Sequential Method.

In Figure 7(a) and Figure 7(b), we plot the results of attacker detection. For the reachable prefix length, the lines of the two methods are almost coincided with each other. Performance of the two methods are almost identical. Only with a small number of attackers ($=$ 500 attackers) and large TCAM sizes ($>$1,000), both methods could find the precise attacker IPs. Otherwise, both methods only find coarse attacker IP ranges. In Figure 7(b), the relation between the number of iterations and TCAM sizes is similar to the case of victim detection by using Concurrent Method. Besides, we show that Concurrent Method detects attackers much quicker than Sequential Method, as the later one has to use lots of iterations to detect victim first.

Summary: Experiment results show that if the priority of DDoS detection is to find victims, Sequential Method is preferable, as it can detect finer potential victim IP ranges given limited TCAM sizes. If the objective of DDoS detection is to find victims as well as attackers, Concurrent Method is preferable if TCAM sizes are abundant, as it finds victims and attackers more quickly.

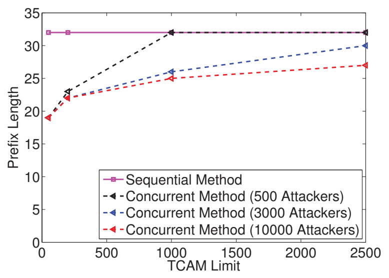
IX. Conclusion

In this paper, we studied how to utilize SDN to detect DDoS attacks. Our methods capture the flow volume feature as well as the flow rate asymmetry feature. We propose Sequential Method as well as Concurrent Method to adaptively change the flow monitoring granularities on all switches to quickly locate the potential victims and suspicious attackers. Through simulations, we demonstrated that our methods have high detection accuracy by capturing the flow asymmetry feature. In addition, we further compared the performance and applicability of our two proposed methods under different TCAM size limits. For future work, we plan to refine our detection methods, and evaluate our methods with packet traces collected from real DDoS attacks. And we will implement our proposed methods in the Openflow platform.

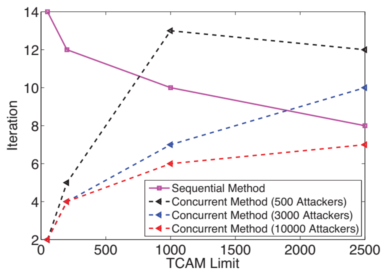
References

[1] "Tcam wiki," http://en.wikipedia.org/wiki/TCAM.

[2] "Openflow specification," https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf.

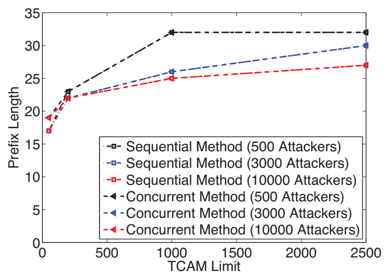


(a) Achieved Prefix Length

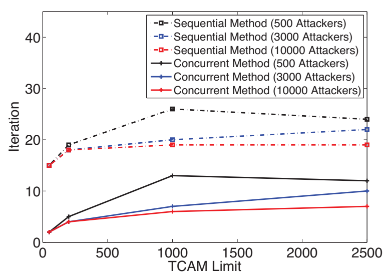


(b) Number of Iterations

Fig. 6: Victim Detection Performance Comparison between Sequential Method and Concurrent Method.



(a) Achieved Prefix Length



(b) Number of Iterations

Fig. 7: Attacker Detection Performance Comparison between Sequential Method and Concurrent Method.

[3] 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.

[4] "Ddos attack loss," http://blog.radware.com/security/2013/05/how-much-can-a-ddos-attack-cost-your-business/.

[5] "Neustar annual ddos attacks and impact report," https://www.neustar.biz/resources/whitepapers/ddos-protection/2014-annual-ddos-attacks-and-impact-report.pdf.

[6] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*. IEEE, 2010, pp. 408–415.

[7] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to ddos attack detection and response," in *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, vol. 1. IEEE, 2003, pp. 303–314.

[8] S. Jin and D. S. Yeung, "A covariance analysis model for ddos attack detection," in *Communications, 2004 IEEE International Conference on*, vol. 4. IEEE, 2004, pp. 1882–1886.

[9] K. Lee, J. Kim, K. H. Kwon, Y. Han, and S. Kim, "Ddos attack detection method using cluster analysis," *Expert Systems with Applications*, vol. 34, no. 3, pp. 1659–1665, 2008.

[10] L. Li and G. Lee, "Ddos attack detection and wavelets," *Telecommunication Systems*, vol. 28, no. 3-4, pp. 435–451, 2005.

[11] V. Sekar, M. K. Reiter, W. Willinger, H. Zhang, R. R. Kompella, and D. G. Andersen, "csamp: A system for network-wide flow monitoring." in *NSDI*, vol. 8, 2008, pp. 233–246.

[12] C.-W. Chang, G. Huang, B. Lin, and C.-N. Chuah, "Leisure: A framework for load-balanced network-wide traffic measurement," in *Architectures for Networking and Communications Systems (ANCS), 2011 Seventh ACM/IEEE Symposium on*. IEEE, 2011, pp. 250–260.

[13] Y. Zhang, "An adaptive flow counting method for anomaly detection in sdn," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 25–30.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[15] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4d approach to network control and management," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 41–54, 2005.

[16] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. M. Parulkar, "Can the production network be the testbed?" in *OSDI*, vol. 10, 2010, pp. 1–6.

[17] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in *Proceedings 10th USENIX Symposium on Networked Systems Design and Implementation, NSDI*, vol. 13, 2013.

[18] M. Moshref, M. Yu, and R. Govindan, "Resource/accuracy tradeoffs in software-defined measurement," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 73–78.

[19] L. Jose, M. Yu, and J. Rexford, "Online measurement of large traffic aggregates on commodity switches," in *Proc. of the USENIX HotICE workshop*, 2011.

[20] Defense4All, "Homepage," https://wiki.opendaylight.org/view/Defense4All:Tutorial.

[21] OpenDaylight, "Homepage," https://www.opendaylight.org/.

[22] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[23] M. Ramadas, S. Ostermann, and B. Tjaden, "Detecting anomalous network traffic with self-organizing maps," in *Recent Advances in Intrusion Detection*. Springer, 2003, pp. 36–54.

[24] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "A hierarchical som-based intrusion detection system," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 4, pp. 439–451, 2007.

[25] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, no. 3, pp. 399–404, 1956.

[26] "Internet2's network topology," https://www.internet2.edu/media/medialibrary/2013/07/31/Internet2-Network-Infrastructure-Topology.pdf.