# Denial of Service Attacks Prevention using Traffic Pattern Recognition over Software-Defined Network

Steven Schmitt[1], Farah I. Kandah[2,*]

[1,2] Computer Science and Engineering Department
University of Tennessee at Chattanooga,
Chattanooga, TN 37403

Recent trends have shown a migration of software from local machines to server-based services. These service-based networks depend on high up-times and heavy resistance in order to compete in the market. Along with this growth, denial of service attacks have equally grown. Defending against these attacks has become increasingly difficult with the growth of Internet of Things and the different varieties of denial of service attacks. For this, our research offers a solution of implementing software-defined networking and real-time metric based techniques to mitigate a denial of service attack within a smaller time window than other comparable solutions. The use of our method offers both efficient attack handling and also flexibility to fit a variety of implementations. The end result being a network that can automatically adapt against new attacks based on previous network activity.

Received on 13 May 2018; accepted on 14 May 2018; published on 23 August 2018

**Keywords:** Denial of Service attack, Software-defined networks, prevention.

doi:10.4108/eai.23-3-2018.155334

## 1. Introduction

Networks are implemented with the core ideas of network integrity, confidentiality, and availability. The security of a network depends on the state of these core ideas, and how they are implemented. As network technologies see advancements in transfer speeds and computational power, the ability to implement flexible and effective security solutions has become difficult. The availability of networks are under constant threat by attacks focusing on bringing down the service or network itself. Defending against denial of service (DoS) attacks has become important in an environment where users rely on these services on a daily basis. The DoS attack in general seeks to bring down network services by flooding a service with dummy traffic with the goal of overloading the service, bring it down and affect its availability. An example of such attacks is presented in Fig. 1. As a service becomes unavailable due to DoS attack, the hosting companies will be significantly affected through the loss in profit and their number of customers. The development of software-defined networking (SDN) has brought network security solutions that will help in solving these issues.

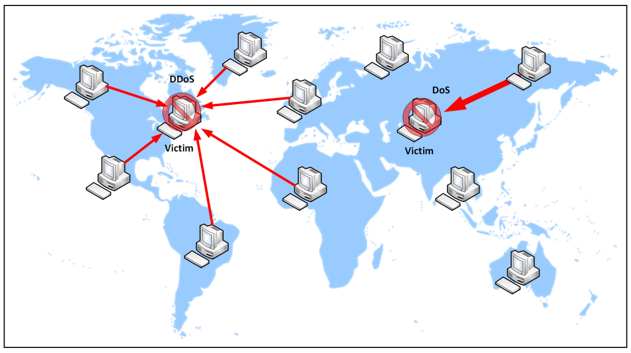

**Figure 1.** DoS vs DDoS

Software-defined networking is the process of virtualizing network hardware to facilitate the flexibility and scalability of software implementations. The idea

*Corresponding author. Email: farah-kandah@utc.edu is to connect hosts through the use of virtual switches and virtual controllers that can be used to automate network functions programmatically. A virtual switch serves the same purpose as a hardware switch by creating a topology to connect hosts while handling basic traffic flow. The virtual controller delegates flows to the switches and handles network-wide operations. The two elements of virtual controllers and virtual switches allow complete control over a network that can adapt to threats. In this work, we present a novel DoS mitigation scheme based on SDN. This work serves to implement a SDN that uses the adaptability and programmability features to defend against different DoS attacks.

The remainder of this paper is organized as follows: We will discuss the related work in Section 2 followed by our motivations and contributions in Section 3. Our problem statement will then be outlined in Section 4. This section will consist of detailed definitions for both the simulations and technologies used as well denial of service attacks. Our proposed denial of service prevention scheme will be presented and discussed in Section 5, followed by our analysis and performance evaluation in Section 6. We will summarize and conclude this paper in Section 7.

## 2. Related Works

The details of how denial of service attacks are handled and defend against are outlined in [1]. There are a variety of current solutions exist today that involve specialized hardware switches, and server load balancing. Solutions such as AutoSlice serve to handle load balancing in order to reduce the overall strain on the network [2]. This is accomplished by evenly distributing traffic throughout the network when large flows are detected. The downside of applying this method to defend against DoS is that it can create network-wide strain during larger denial of service attacks. During a large DoS attack, the solution might propagate the attack across the entire network.

Singh *et. al.* in [3] offered another solution by the use of a buffer to create a waiting window before blocking the traffic. In their proposed solution, as soon as a suspicious traffic is identified, the authors proposed to increase the queue buffer at the switches and see whether the traffic keeps coming. During this process the attacking hosts are requested to decrease their traffic rate. If the attack persists, the host is then blocked by the network. Unfortunately, this slowly allow potential attacks to consume more bandwidth over time. Although this prevention strategy offers a solution to identify large legitimate traffic, it is still vulnerable to large distributed attacks. This is because of the large timing window the prevention algorithm allows before blocking an attacking host.

Both solutions stated above also have a large increase in processing overhead on the network controller. This comes as a large cost for smaller networks that may not be able to handle the increase in workload. The costs are based on fundamental network architecture problems involving limited bandwidth and security implementations [4] [5].

For many companies a complex security solution for DoS is not an option due to the lack of resources or the incompatibility with their current systems. To lessen the resource requirement in order to better secure a network, cheaper solutions to monitoring networks have been found such as sFlow [6]. Technologies such as sFlow allow for low resource monitoring that can be adapted to improve security needs [7] [8]. Monitoring the network allows the polling of different metrics at the switch or host level, such as the number of messages exchanged between network entities, the bandwidth consumption, the number of dropped packets, etc ... that can be used to recognize the traffic pattern in the network, which can then be easily implemented in scripting solutions. By nature a network is suppose to transmit data, and the denial of service attack is exploiting this core element, which makes it hard to completely defended against [1][4][5]. The idea of denial of service prevention is to identify and mitigate attacks in a timely manner. This will minimize the time frame in which the targeted network is affected. In this work, we intend to refine this defense against denial of service using more adaptable scheme that offers rapid detection and mitigation.

## 3. Motivations

The current customer climate demands a high degree of reliability and availability from network services that are used on a daily basis. This exponential increase in network traffic has put a strain on older network architecture models that fail to handle large data flows without down time. As we observed that current solutions to denial of service attacks through the use of load balancing and buffer limits for the networks fall short of delivering *quick identification* and *mitigation* of attacking hosts [1][3], we focused our work to contribute a more rapid solution to denial of service attack compared to the previously stated solutions as well as lower processing workloads for the virtual network devices.

## 4. Problem Statement

The main issue in current solutions to DoS attacks involves decreased availability. During a DoS attack affected networks are effectively cut off from the Internet rendering the services being provided unusable [1]. Our research seeks to improve on this solution by allowing for increased service availability by recognizing traffic pattern through smartly analyzing the network traffic and packets. This will improve the overall network uptime while also maintaining network performance during attacks.

Throughout our discussion, we will be using the following definitions:

**Definition 1.** A Denial of Service (DoS) attack is a network threat that seeks to affect the availability of a network or service by introducing fake traffic to the target. □

**Definition 2.** A Distributed Denial of Service (DDoS) attack is a DoS attack where a multitude of attackers are used to send a synchronized attack on a target. □

**Definition 3.** A virtual switch is a software-based implementation of a hardware switch in a network. It often serves as a node to route traffic between hosts on a network. □

**Definition 4.** A virtual controller is a software-based implementation of a hardware controller in a network. A controller often serves as a primary point of control that can designate work to network switches. □

We define our **S**oftware-defined networking **D**enial of service **M**itigation (SDM) problem that our work aims to solve as follows: *Given a network under DoS attack, our scheme seeks to detect and mitigate any DoS attack taking into consideration the improvement of the efficiency and the speed of mitigating denial of service attacks.*

To achieve this we will use traffic pattern recognition using metric based traffic analysis to rapidly identify attacking hosts while allowing for legitimate traffic to continue to be processed.

## 5. Software-defined networking Denial of service Mitigation

Our research focuses on mitigating two different types of denial of service attacks. This is done through a single algorithm that uses live network metrics to gauge the state of the network. By basing decisions off previous network states, our DoS Mitigation Scheme is able to defend against several attacks. Our Software-defined networking DoS mitigation scheme is presented in Algorithm 1. Our scheme begins with initializing the network through identifying any OpenFlow switches ($S$) on the network and their available ports (Lines 2-5). By doing this we can map the network topology allowing for better flow control. Once this is done the execution phase of the scheme will begin (Lines 6-18). This phase polls each OpenFlow switch for live metrics. During normal traffic states these metrics are used to set the aggregate threshold ($T_r$) for each metric such as packet rate. Once this is complete, every polling cycle will check the current metrics against the metric thresholds. Any conflict during this step will push the scheme into the update phase. This phase consists of identifying the traffic flows that breached the thresholds by tracing the packets back to the host. Once found, the host ($h$) will be removed from the OpenFlow switch's flow tables and the switches will be set to drop all packets from the host. This continues for a set timeout interval ($T_i$) at which time the host can re-enter the network.

**Algorithm 1** SDM: Software Defined networking DoS Mitigation Scheme

```
 1: Set the controller ($C$) to listen for connections;
 2: for each switch ($S_i$) connected to $C$ do
 3:     Assign switch ID
 4:     Identify available port(s)
 5: end for
 6: Define polling interval: $I \leftarrow \beta$;
 7: Set $T_r \leftarrow \frac{\sum Rate}{Ports}$
 8: for every $I$ do
 9:     for each switch ($S_i$) connected to $C$ do
10:         if $\frac{\sum Curr_{Rate}}{Ports} > T_r$ then
11:             $Set_{hosts} \leftarrow$ flow sources;
12:             for each $h_i \in Set_{hosts}$ do
13:                 $T_i \leftarrow$ timeout for $h_i$ to rejoin
14:                 Block $h_i$ flow for $T_i$
15:             end for
16:         end if
17:     end for
18: end for
```

### 5.1. Mitigating Single-Host Denial of Service Attacks

One aspect of our DoS Mitigation Scheme focuses on mitigating a denial of service attack that originates from a single attacker. This attacker will have a single target host with the goal of bringing down any services running on the host. Since denial of service attacks can be based in overworking common network functionality there are various ways they could be executed. Amongst all of these variation there are common traits and characteristics of a single host denial of service attack. These characteristics can then be used along side monitoring network activity. In order to properly handle denial of service attacks real network activity must be identified versus malicious activity. Our SDM scheme does this by monitoring packet rate, packet size, and flow path.

To illustrate our algorithm, let us consider the following example by defining a network flow that contains common properties of a denial of service attack. This network flow can then be monitored at the network switch level through sFlow to gain real time metrics that will allow us to recognize network traffic patterns. Through the REST API and JSON handling, we will be able to keep an up to date list of all network flows and their load on the network. Once the current flows of the network are mapped, thresholds can be defined based on the network activities. The threshold will be set to signify network flows that are acting outside of normal network metric ranges. Note that, the thresholds can be adapted based on the host network's common activity. This makes our SDM Scheme be adapted across a variety of networks settings.

Let us consider the *packet rate* of each flow on the network as the first threshold to be monitored, and let us define a network flow with a packet rate exceeding 1000 packets per second as being malicious. Note that, any flow that passes this threshold is collected by sFlow and sent off to the Floodlight Controller for analysis. Besides that *packet rate*, we use *packet size*, which is compared across packets to check for variance. Without this variance, it can be assumed that the network flow could be malicious. With this comparison along with the packet rate monitoring threshold, our SDM Scheme is able to rapidly identify and stop a single host denial of service attack by detecting large increases of packet rate over a short time period.

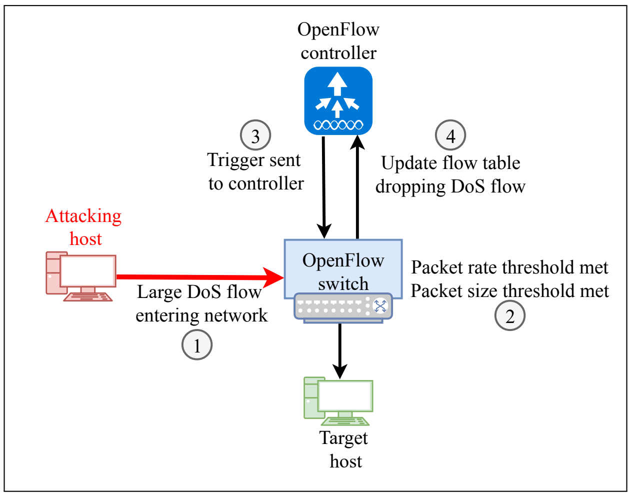
OpenFlow controller
3
Trigger sent to controller
4
Update flow table dropping DoS flow
Attacking host
OpenFlow switch
Packet rate threshold met
Packet size threshold met
2
Large DoS flow entering network
1
Target host

**Figure 2.** DoS Attack Mitigation Scheme

This scheme is presented in Fig. 2, which outlines a generalized idea of how our SDM scheme handles a denial of service flow entering the network. The important part of our SDM scheme is the ability for other flows to still function during the mitigation process, which allows for regular network activity to continue while the attack is being handled. In step 1 of the flow chart we see a large DoS flow entering the network. The flow is then handled by the OpenFlow switches (step 2), where they can detect when a threshold is met. Once a threshold is met, a message will be sent to the Floodlight controller (step 3), which in turn replies with instructions to all network switches to drop the detected flow (step 4). With this, our SDM scheme can minimize the amount of malicious traffic the target host receives.

## 5.2. Mitigating Distributed Denial of Service Attacks

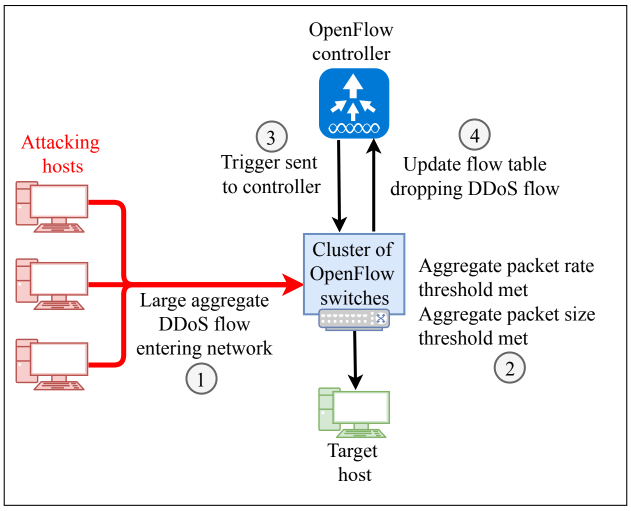
OpenFlow controller
Attacking hosts
3
Trigger sent to controller
4
Update flow table dropping DDoS flow
Cluster of OpenFlow switches
Aggregate packet rate threshold met
Aggregate packet size threshold met
2
Large aggregate DDoS flow entering network
1
Target host

**Figure 3.** DDoS Attack Mitigation Scheme

Another form of denial of service attack is the distributed denial of service attack. This attack follows the same principle of a single host denial of service attack, but instead of using a single host, the attack is distributed across multiple attackers. Spreading the attack across multiple hosts makes it difficult to identify the source of the attack. Conventional methods of blacklisting an attacker's IP address no longer work due to the large amount of sources [1]. Our SDM scheme can further detect a distributed attack by focusing on aggregating the flow data instead of singling out a single flow. This can be done by tracking the same metrics previously tracked, such as *packet rate* and *packet size*. With this, the thresholds are also adjusted to better handle aggregate network data. The threshold for our SDM scheme for packet rate can then both monitor a single flow exceeding the threshold as well as the summation of multiple smaller flows exceeding the threshold.

We will use Fig. 3 to provide a visualization of how our SDM scheme handles distributed denial of service attack. Similar to Fig. 2, step 1 shows a large DDoS flow from multiple attacking hosts entering the network. Once this attack triggers the threshold across the OpenFlow switches in step 2 a message can then be sent to the Floodlight controller in step 3. This JSON message contains all suspect flows that have entered the network during the time frame of the attack. The Floodlight controller then replies to drop all suspect flows blocking all communications for a short period of time with the attacking hosts.

Another aspect of our SDM scheme is the timing window of the metrics captured. For this, a constant buffer of metrics will be kept and refreshed with the most current sFlow data. By only focusing on current sFlow metrics, it allows normal network flows to not be flagged as suspect. Once our defined thresholds are triggered by the aggregate metrics, flows will then be dropped by the Floodlight controller. The selected dropped flows will be chosen based on their *packet rate* and *packet size*. Once a flow is dropped it will then be allowed to reconnect with the network after a set amount of time.

## 6. Analysis and Performance Evaluation

In this section we will present the performance of our proposed SDM scheme and discuss how it handles both types of denial of service attacks. In our implementation, we used Mininet to build the network topology [9], which is presented in Fig. 4.

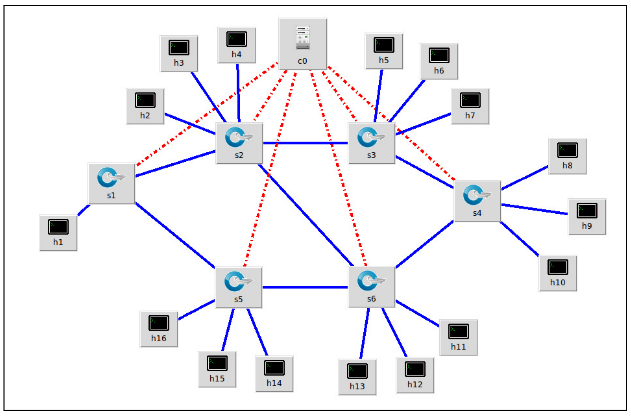

**Figure 4.** Test network architecture – Mininet

For gathering results sFlow was used to capture real-time data for the network. The D-ITG traffic generator was used to create scripts that simulate both normal network activity, single-host denial of service, and distributed denial of service attacks [10]. The network typologies created in Mininet consists of simple tree-based typologies along with a larger topology to simulate the distributed denial of service attack.

In Fig. 4 OpenFlow switches are denoted as **s1** through **s6** while hosts are denoted as **h1** through **h16**. For this example **c0** represents the remote Floodlight controller used for our network. For the sake of this research ping floods will be used to test the resilience of the network to denial of service attacks. These attacks will be combined with UDP flows to show that our algorithm is able to mitigate a variety of denial of service attack types.

### 6.1. Analysis and Evaluation of a Single-Host DoS Attack

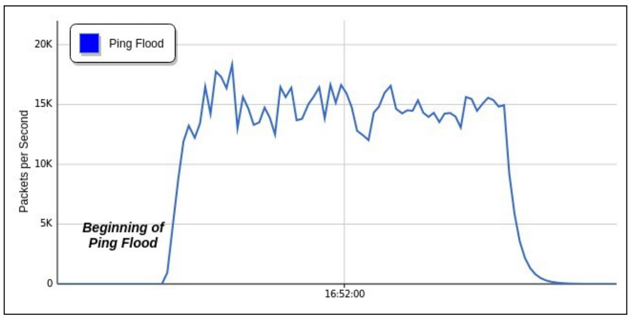

**Figure 5.** Unprotected DoS Flow

Using the above topology in Fig. 4, our SDM algorihm was put into place with a packet rate threshold of 1000 packets per second. In this test individual flows will be monitored to see if each reach the designated threshold. Along with the packet rate threshold, packet size is also monitored to gauge how much variance each flow has. The first run of our algorithm consisted of an isolated flow where a single DoS attack is sent through the network without legitimate traffic. For this the attacking host sends a ping flood to the target host through several OpenFlow switches. The resulting network flow is presented in Fig. 5.

As seen in Fig. 5, once the denial of service attack begins, it quickly reaches an upwards of 15,000 packets per second. This heavy traffic on a switch disrupts any normal network traffic that may be on the switch. Fig. 6, also shows further behavior of the denial of service attack flow. The majority of the packets during the denial of service attack are under the Internet Control Message Protocol (ICMP) protocol. This helps further identify the attack as a ping flood. The overall effect of the denial of service attack on the network load is shown in Fig. 7.

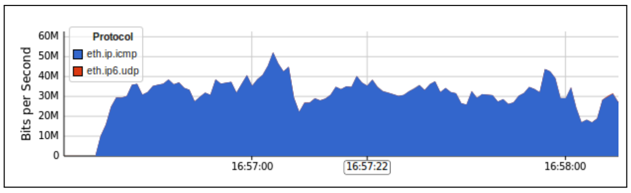

**Figure 6.** Unprotected DoS Flow Packet Analysis

As seen in the preceding figures a single-host denial of service attack can be enough to generate substantial network load. We can now compare these results with the results of our denial of service prevention algorithm (SDM). Our results for this are presented in Fig. 8. It can be seen that as soon as the packet rate approaches the threshold of 1000 packets per second the flow begins

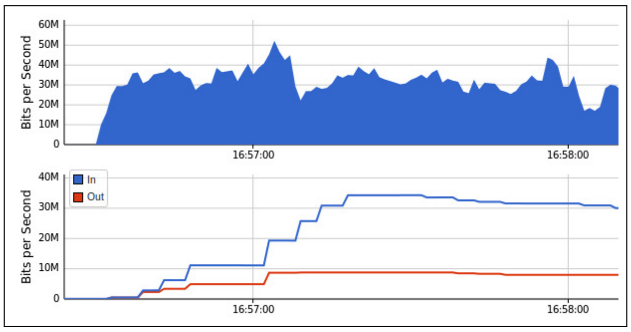

**Figure 7.** Unprotected DoS Flow Network Load

to be mitigated. From this point a JSON message is produced containing the source and the destination for the flow along with any metrics pertaining to the flow itself. Upon receiving the JSON message the Floodlight controller can then push an update to all OpenFlow switch flow tables allowing for the denial of service traffic to be dropped and the host to be blacklisted.

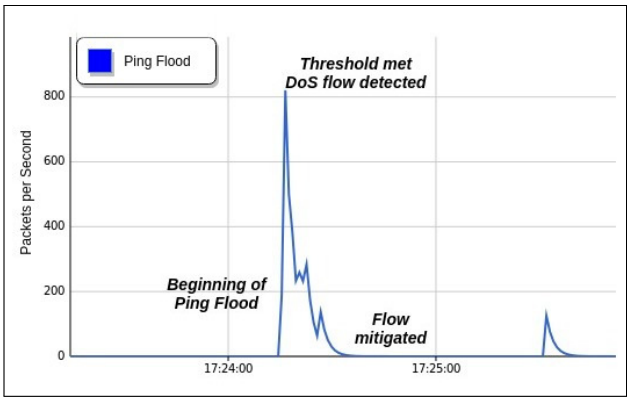

**Figure 8.** Protected DoS Flow

In this example our scheme is able to quickly handle the ping flood produced by the attacking host. For the attacking host to continue communicating within the network a JSON message will need to be sent to remove the host from being blacklisted. For our SDM this functionality is automated and it occurs after a set amount of time. To further test our SDM scheme and its performance in identifying DoS flows, the same ping flood was sent during normal network operations. D-ITG traffic generator was used to send a stream of UDP data packets across the network. During this time the ping flood was sent to the target host to disrupt the UDP traffic flow.

Fig. 9 outlines our scheme's performance during the attack. Once the ping flood begins, a large influx of ICMP packets enters the network. Our algorithm quickly identifies this flow as malicious, and it is able to smartly drop the packets while maintaining the UDP data flow. The use of sFlow metrics and the Floodlight controller allows for fine-grained control over network flows allowing normal network traffic to remain untouched.

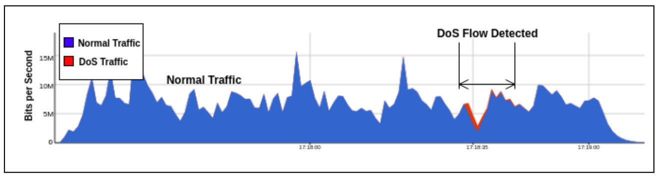

**Figure 9.** Protected UDP Flow

In comparison to the prevention scheme presented in [3], our solution is able to quickly identify and mitigate attacking hosts. By using a metric-based approach we were able to avoid the waiting times that are implemented by Singh *et.al.*'s algorithm that requests hosts to verify that they are sending legitimate traffic. Along with this, Singh *et.al.*'s scheme causes increased overhead for the virtual controller. During larger scale attacks this overhead can lead to further delays and network strain. Our SDM scheme avoids this by performing the detection and processing through an sFlow server, and only communicating with the Floodlight Controller when flow handling updates are needed.

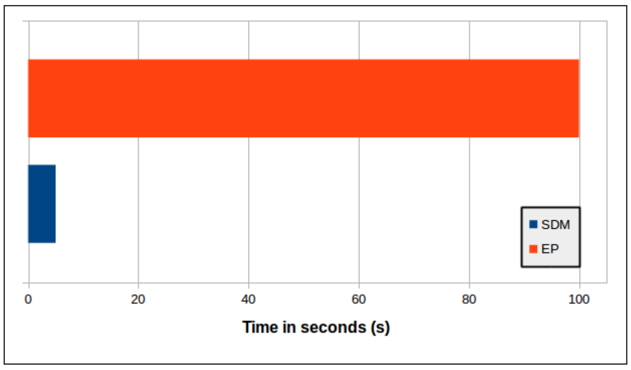

**Figure 10.** Average Mitigation Time: SDM vs EP

As seen in Fig. 10, on average our method of denial of service mitigation performs faster than Singh *et.al.*'s method. This is due to the added processing time Singh *et.al.*'s method takes to identify and mitigate flows. Before a flow is dropped, Singh *et.al.*'s method allows for at least a 100 second response time from the attacker. We argue in our scheme that this adds too much delay in mitigating the attack, which could lead to a decrease in the network performance during the attack.

Overall, our single-host denial of service prevention algorithm was successful in identifying and mitigating large denial of service flows. This was done through the use of highly adaptable software-defined network techniques. In the following section a similar algorithm

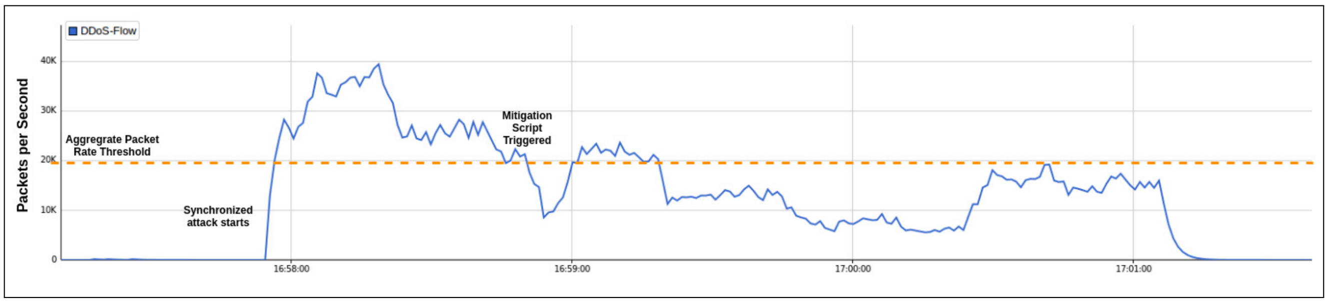

**Figure 11.** Synchronised attack – Protection Enabled

is implemented to handle distributed denial of service attacks.

## 6.2. Analysis and Evaluation of a Distributed DoS Attack

To simulate a distributed denial of service attack several hosts on our test network were scripted to send a synchronized attack on a single host using various traffic types. A D-ITG script was used to synchronize the attack across multiple hosts. Hosts were chosen across the network to involve multiple network switches in the simulation. This allowed us to test the ability of our system to collect aggregate switch data while watching for spikes. Both ICMP and UDP traffic were used in our simulated attack to further test the ability of our algorithm to identify suspect flows of various traffic types. Along with this the D-ITG script produced a uniform distribution for packet rate during the simulation. Other network traffic distributions can further be tested to probe for weaknesses in detection. Due to processing constraints, fluctuations in packet rate from each attacking host can be expected, but on average were set at a fixed rate of 5000 packets per second.

The idea behind our mitigation scheme is to use the sudden increase of network activity as an indicator of which flows are suspect. For this the Floodlight controller for our network will keep an aggregate total for several metrics that will update with only the most current values. This will create a timing window to monitor the network. The same type of script we used in our single-host denial of service attack can be used to select suspect flows and send triggers to drop them once identified. A threshold of 20,000 packets per second was used to trigger the mitigation script. This threshold was chosen based on normal network activity, and can be adjusted based on an individual network's needs. For our simulation 20,000 packets per second allows for normal network traffic to continue without triggering mitigation. Fig. 11 displays our distributed denial of service script in action against a multi-host attack on a single target.

Once our script is implemented with sFlow, we can see that the aggregate packet rate sees a spike once the distributed denial of service begins. Once this threshold window is detected, suspect network flows are dropped and blocked for a set period of time. As seen in Fig. 11, once the aggregate packet rate drops below the threshold, the rate at which flows are dropped decreases. This is done to avoid dropping legitimate network traffic. Even if attacking flows still persist the attack as a whole has been mitigated.

Table 1 shows a sample of the data collected by sFlow and outlines the collection of the aggregate packet rate across all network switches. Mean packet rate is shown as approximate values due to fluctuations in packet rate caused by hosts performance. Network switches are shown as sFlow agents labeled s1 through s6 in our example. Each sFlow agent sends its metrics to the remote sFlow instance where it is analyzed further. Based on our topology presented in Fig. 4, we can see that each network switch excluding s1 had an attacking host, and that the attacking flows were unable to maintain their flows for the target at 10.0.0.1. For our attack the subnet 10.2.0.0/16 consisted of all attacking hosts across multiple OpenFlow switches. Once an individual flow is identified, a blocking signal can then be sent to the Floodlight controller which can then issue orders to drop packets pertaining to the suspect flow. By acting at the switch level, our SDM scheme is able to stop the distributed denial of service attack at the first switch it encounters.

The values and thresholds used in our model are specific to the simulated network we created. For our work to be adapted to a new network, new thresholds and values will need to be adjusted to match the common traffic load of the network.

Since our method is based on a variety of real-time metrics, it allows for a large variety of traffic types to be identified. Even more fine tuning can be added to increase the versatility in the ability for our model to detect suspect network flows. By relying on real-time network metrics our research successfully mitigated denial of service attacks. Further work and study could help improve the understanding of more

**Table 1.** Sample Network Flows

| Flow number | Protocol | IP Src | IP Dest | Agent | Mean packet rate |
|---|---|---|---|---|---|
| **5** | UDP | 10.2.0.7 | 10.0.0.1 | S3 | ~5000 |
| **6** | UDP | 10.2.0.12 | 10.0.0.1 | S6 | ~5000 |
| **7** | ICMP | 10.2.0.5 | 10.0.0.1 | S3 | ~5000 |
| **8** | UDP | 10.2.0.3 | 10.0.0.1 | S2 | ~5000 |
| **9** | ICMP | 10.2.0.14 | 10.0.0.1 | S5 | ~5000 |
| **10** | UDP | 10.2.0.13 | 10.0.0.1 | S6 | ~5000 |
| **11** | UDP | 10.2.0.9 | 10.0.0.1 | S4 | ~5000 |
| | | | | **Total** | ~35,000 |

advanced attacks to find the key indicators of their occurrence. Once this behavior is found it could quickly and efficiently be implemented in our type of mitigation model.

## 7. Conclusion

Recent trends have shown a migration of software from local machines to server-based services. These service-based networks depend on high up-times and heavy resistance in order to compete in the market. Along with this growth of network services, denial of service attacks have equally grown. With a simple set of tools, attackers could bring down one of these services. Our research in the use of software-defined networking to mitigate and detect denial of service attacks has shown that the adaptability and flexibility metrics based solutions allow for a complete solution to denial of service attacks. By relying on real-time metrics, our network was able to adapt to large flows of data quickly and effectively while maintaining services on unaffected flows.

## References

[1] Christos Douligeris and Aikaterini Mitrokotsa. Ddos attacks and defense mechanisms: Classification and state-of-the-art. *Comput. Netw.*, 44(5):643–666, April 2004.

[2] Zdravko Bozakov and Panagiotis Papadimitriou. Autoslice: Automated and scalable slicing for software-defined networks. In *Proceedings of the 2012 ACM Conference on CoNEXT Student Workshop*, CoNEXT Student '12, pages 3–4, New York, NY, USA, 2012. ACM.

[3] S. Singh, R. A. Khan, and A. Agrawal. Prevention mechanism for infrastructure based denial-of-service attack over software defined network. In *International Conference on Computing, Communication Automation*, pages 348–353, May 2015.

[4] D. Kreutz, F. M. V. Ramos, P. E. VerÃŋssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015.

[5] Kamal Benzekki, Abdeslam El Fergougui, and Abdelbaki Elbelrhiti Elalaoui. Software-defined networking (sdn): a survey. *Security and Communication Networks*, 9(18):5803–5833, 2016. SCN-16-0386.R1.

[6] sFlow Team. sFlow. http://www.sflow.org/index.php, 2017. [Online; accessed March-2017].

[7] W. Wang, X. Zhang, W. Shi, S. Lian, and D. Feng. Network traffic monitoring, analysis and anomaly detection [guest editorial]. *IEEE Network*, 25(3):6–7, May 2011.

[8] Alisha Cecil. A summary of network traffic monitoring and analysis techniques. 2010.

[9] Mininet Team. Mininet. http://mininet.org/, 2017. [Online; accessed March-2017].

[10] Alessio Botta, Alberto Dainotti, and Antonio Pescapè. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531–3547, 2012.