International Journal of Management, IT & Engineering
Vol. 9 Issue 9, September 2019,
ISSN: 2249-0558 Impact Factor: 7.119
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Distributed Denial of Service Mitigation in Software Defined Networking

Godfrey Nengo Kihamba

Odira Elisha Abade

Abstract

Software Defined Networking is a new networking architecture that is fast becoming popular and is already widely deployed across the world even by large companies like Google, Facebook etc. A centralized device called the controller has the whole view of the network making it easy to manage and control the underlying infrastructure. This offers many advantages and also vulnerabilities. This paper looks at the serious threat of protocol based DDoS attacks. Attackers can take advantage of protocol functioning to exploit a network. This research explored how we could take advantage of the SDN architecture to detect and stop these attacks before serious damage is done on the network.

Keywords:

SDN;
DDoS;
REST API;
SYN Flooding;
Ping Flooding.

Author correspondence:

Godfrey Nengo Kihamba,
University of Nairobi, School of Computing and Informatics.
Email: godfrey.nengo@students.uonbi.ac.ke

Elisha Abade,
University of Nairobi, School of Computing and Informatics.
Email: abade@uonbi.ac.ke

1. Introduction

A Software Defined Network is one in which switches do not process incoming packets (Open Networking Foundation (ONF), 2017). SDN's main characteristic is the separation of the network into three layers. Layer 1 is the infrastructure layer constituting unintelligent network equipment such as switches. These simply drop and/or forward packets depending on the flow information installed onto them by the next layer – the control layer. The control layer links applications running on a layer above to the lower layer which is the infrastructure layer. The final and top layer is the application layer; it's composed of applications that communicate with lower layers through an application programming interface.

Incoming packets are assessed by devices in the infrastructure layer. They always look for a match and the corresponding action e.g. forward, drop etc. If no match is found, the packet will then be sent to the controller for further processing.

The Software Defined Network architecture is shown in the diagram below:

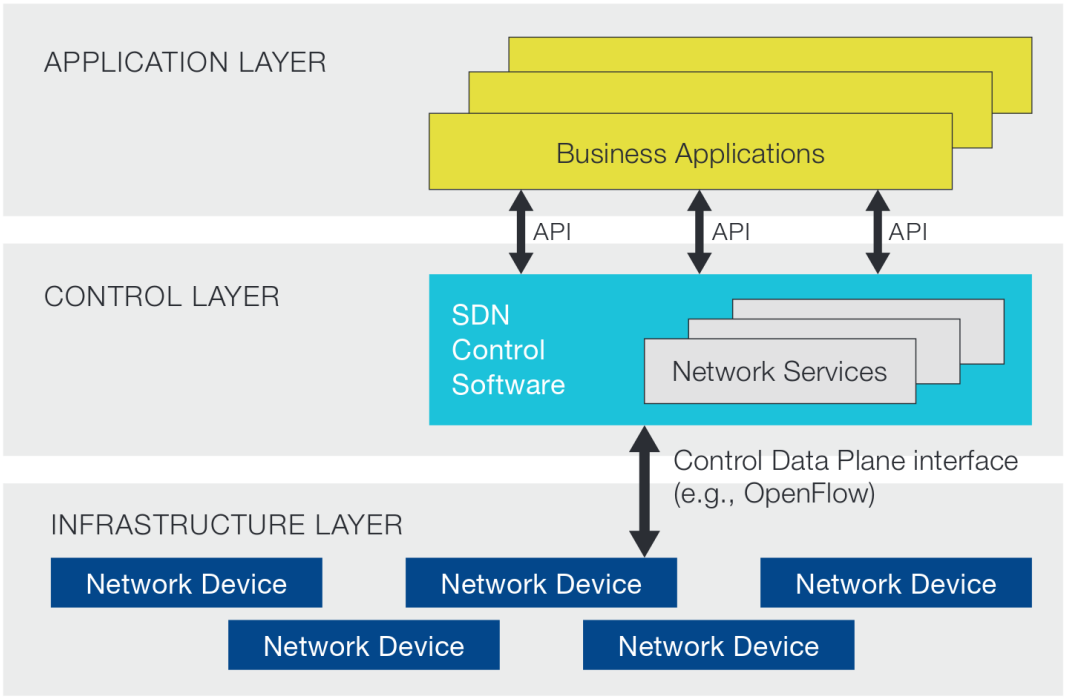

Figure 1 A High Level Schema of the SDN Network

1.1 DOS and DDOS Attacks

Denial of Service attacks are usually originated by malicious individuals with the aim of interrupting regular or standard functioning/operation of a service or application. In case the malicious user incorporates many machines/hosts in the attack, then we refer to it as a distributed denial of service.

Detecting DDoS based on flooding is a major challenge for security of the internet now since the attack relies heavily on large resource imbalance between the internet consisting on millions of devices and the limited resources on a victim host when handling unusually large number of requests that are not genuine resulting in normal user requests not being processed because resources are exhausted. (Rodrigo, Edjard, & Alexandre, Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow, 2010).

1.2 DDOS Types

Classification of these attacks can be done using the traffic characteristics, amount of packets/data used, and what weakness was targeted on the host. Based on these, we can have three distinct types of DDoS i.e. volumetric, protocol based and finally application based attacks. (Kesavan, 2016)

Volume based DDoS - Usually have a large size of traffic that could even be in hundreds of Gbps, they do not require a similarly large traffic to be originated by the attacking hosts. This characteristic enables them to be very simple to generate.

DDoS using Protocols - These mainly use vulnerabilities in network and transport layers.

DDoS using applications - Malicious users in these attacks are experts having intricate knowledge of how the targeted application operates. Generated packets for attack are usually safe but target the OSI application layer; it triggers a process in the layers inner working which takes up all the resource resulting in unavailability. This is why it is nearly impossible to tackle this kind of attacks.

There are other different forms of DDoS attacks e.g. User Datagram Protocol flooding, SYN Flooding using TCP, Ping flooding, smurf flooding etc.

1.3 Distributed Denial of Service Detectin and Mitigation

There are well established mechanisms for protection against DDoS and mitigation of DDoS attacks when they happen to occur. (Haris, Ahmad, & Ghani, 2010). They involve:

Prevention of DDoS, Detecting DDoS, Filter-based mechanisms and Determination of DDoS sources.

Firewalls based on destination addresses combined with filtering of packets has widely been used with exceptional success. (Hussain, Heidemann, & Papadopoulos, 2003).

Network based DDoS detection involves placing a number of monitors over the entire network. These monitoring agents have predefined signatures; the monitors constantly attempts to match traffic passing through the network against the signatures. This activity does not disrupt the traffic flow.

This detection method does not feature any learning but relies on analysis developed externally to come up with required signatures to be used by monitoring agents.

1.4 Firewalls

Firewalls are network security systems used to control the flow of traffic that is inbound or outbound between networks. They analyze traffic by looking at characteristics such as layer 2/3 headers - hardware address, source/destination addresses.

Two broad categories of firewalls exist - network layer firewalls and application layer firewalls.

Network layer firewalls - This type of firewall is also known as packet filtering firewall. Packets are processed and analyzed based on ports or protocol in use, MAC address, layer three source and destination address.

Application based firewalls - These types of firewalls operate at the top layer of the network i.e. application layer.

1.5 Problem Statement

There has been a lot of research focusing combatting DDoS attacks in Software Defined Networks. SDN has been used in conjunction with other technologies such as SOMs (self-organizing maps); support vector machines, entropy, statistical methods etc. to detect and mitigate DDoS attacks.

However, these methods seem to be generalized and mainly address the volumetric attacks. This research aims to build on previous work that already addresses volumetric attacks, to find a solution for protocol based attacks. Combating these attacks entails understanding the working of the various attacks and designing a specific solution geared towards detecting and mitigating the specific attack.

1.6 Relevant Literature on the Subject

(Mousavi, 2014), used entropy to detect DDoS attacks. Entropy (Shannon-Wiener index) is defined as the measurement of randomness with regards to a randomly varying data and for this research, traffic traversing over the network using destination internet address.

(Dao, Park, Park, & Cho, 2015) Uses a statistical method based on the fact that majority of users send a minimum of five packets to any destination at a time. Users considered suspicious will send less than five packets at a time.

In (Phan, Bao, & Park, 2016) traffic characteristics of flows to a server in an ISP core network in the event of normal traffic flow and during a DoS/DDoS attack are analyzed. A combined usage of support vector machines (J. & Cristianini, 2000) and Self Organizing Map (Kohonen, 1997) enhances the efficiency of classifying flows for the traffic running on the network. For this setup, SVM is deployed as a tool for supervised learning model that is capable of analyzing given data and notice patterns. Self-organizing map is known as an efficient algorithm for the purpose of un-supervised learning in artificial neural networks. This collaboration of SVM and SOM takes advantage of each algorithm used; not just for generation of accurate decisions but to ultimately bring down time spent processing. Additionally, SVM and SOM collaboration is applied in a proposal to limit distributed denial of service attack consequences.

(Rodrigo, Edjard, & Alexandre, Light-weight Distributed DoS Flood Attack Detection Using NOX & OpenFlow, 2010) Used SOM for distributed denial of service detection. Three modules are used in the solution.

1. Flow collector - periodically collects information of flow tables installed on switches. These communications are done over a secure channel.
2. Feature extractor - picks flows obtained from above. Isolates and processes characteristics pertinent to distributed denial of service detection; stores them in 6 tuples to be sent to the classifier.
3. Classifier - used to analyze 6 tuple to determine if it matches distributed denial of service attack or normal traffic. The classification could be achieved using either statistical or learning methods. For this research, SOM was used.

1.7 Proposed Solution

Below is a simple conceptual framework of our solution:

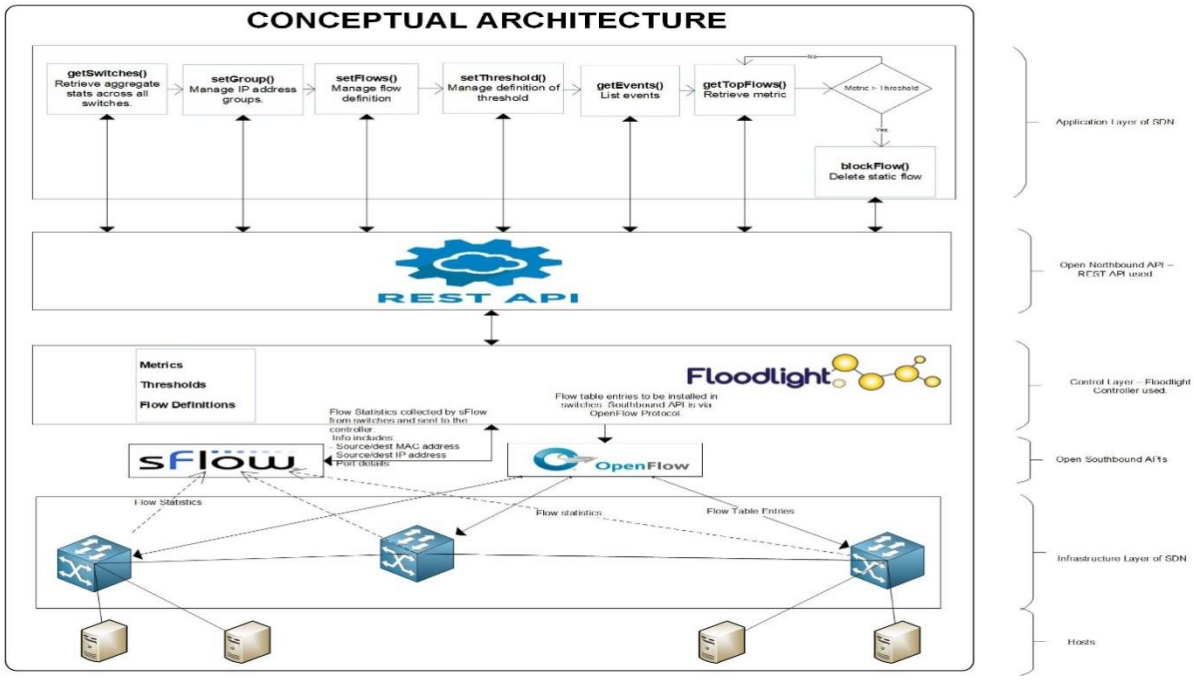

Figure 2 Conceptual Architecture

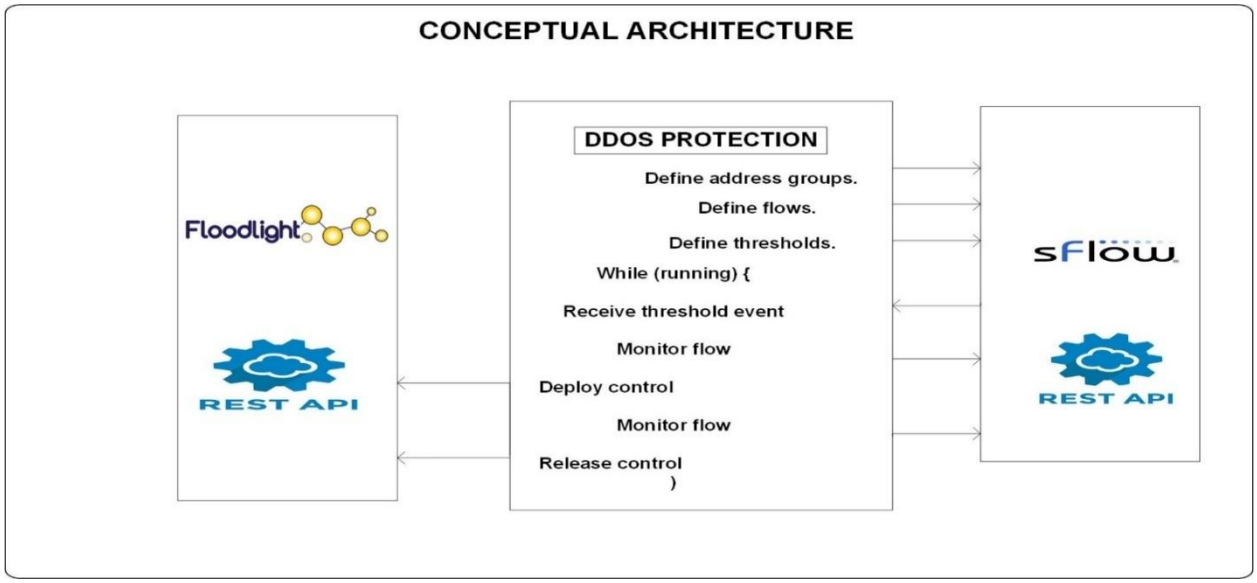

Figure 3 Conceptual Architecture

The conceptual architecture above depicts how the system will defend against Ping Floods and TCP SYN attacks. The system defends when there is a malicious flow by deleting the flow.

1. Definition of address groups – traffic is categorized into internal and external. Internal traffic is represented as an address block and is the address space used on the hosts and switches of the SDN. In our case it's the address block 10.0.0.0/8. External traffic is any other traffic that might be accessing our network. We define the whole address space as external traffic i.e. 0.0.0.0/0.
2.Flow definition - We define the flows by assigning names to packet characteristics identified and implemented in grouping the same packets into flows or keys, a unique value to associate the flow with and finally a filter for selection of specific traffic.
3. Defining thresholds - We define thresholds of packets that could signal a breach e.g. 10000 packets per second for a given flow.
4. Get threshold events - The application keeps polling for events in order to obtain notifications in case of new events.
5. Monitoring of flows - The flow is monitored and more information regarding the flow is obtained. This information is extracted from the event plus the agent, metrics and data sources.

6.Deploying control – after monitoring and determining that thresholds have been exceeded, the controller installs flows that result in traffic from the identified source being dropped.
7. Monitoring of flows – the flow is continuously monitored to ensure that the control measures took effect on traffic.
8. Releasing control - After a given time, the control action is stopped so that flow table entries engaged in blocking the attacker are released. If the attack happens again, another event will be generated and new control implemented.

2. Research Method

The research method used is experimentation with a virtual environment. An application was also developed to detect and mitigate protocol based DDoS attacks.

To meet our prototype design objective, the following were identified:

A basic PC with the following specification was found to be adequate for the hardware platform. Intel Core i7-5500U CPU. 2.4 GHz processor. 16GB Random Access Memory. 10/100/1000 Mbps network interface.

Hypervisor – Virtualbox
OS – Ubuntu Linux
SDN Controller – Floodlight.
A network emulator – mininet.
A traffic generation – nping, ping and hping3.

Network Topology

A topology consisting of one server and thirty four PC clients. Among the PCs we will have two acting as DDoS attackers.

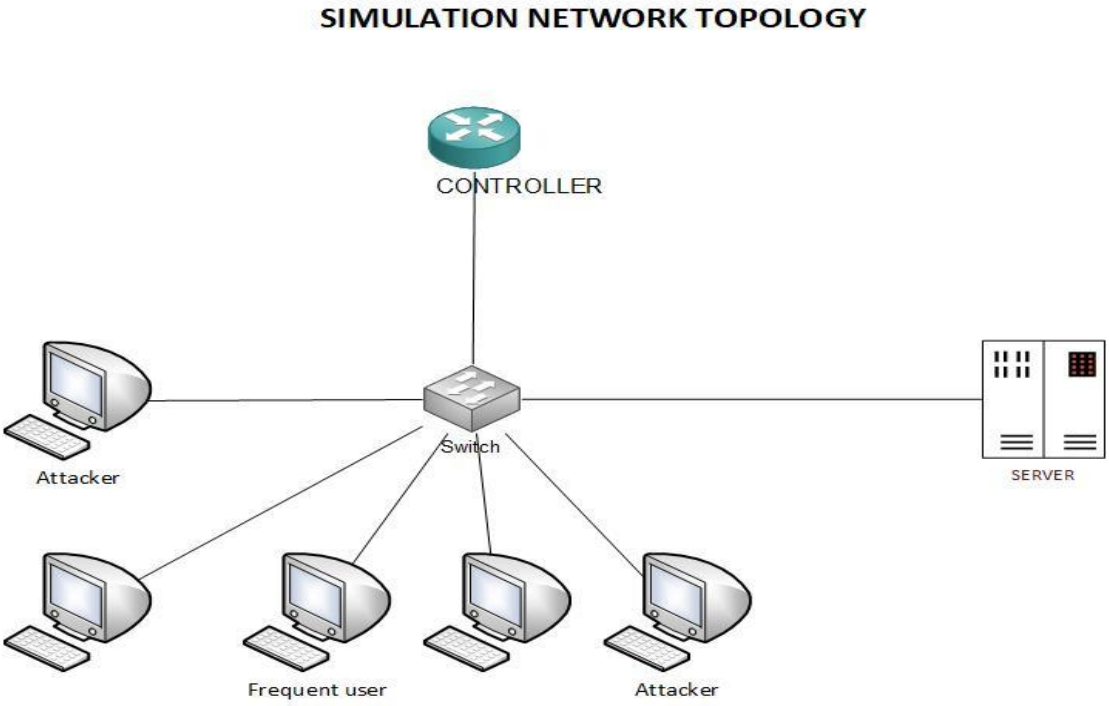

Figure 4 Simulation Network Topology

Application Development
Language – node.js and JSON
Software Development Model – Waterfall Software Development.

3. Results and analysis

In this section, it is explained the results of research and at the same time is given the comprehensive discussion.

3.1 Ping Flood Simulation

Traffic on the network before attack:

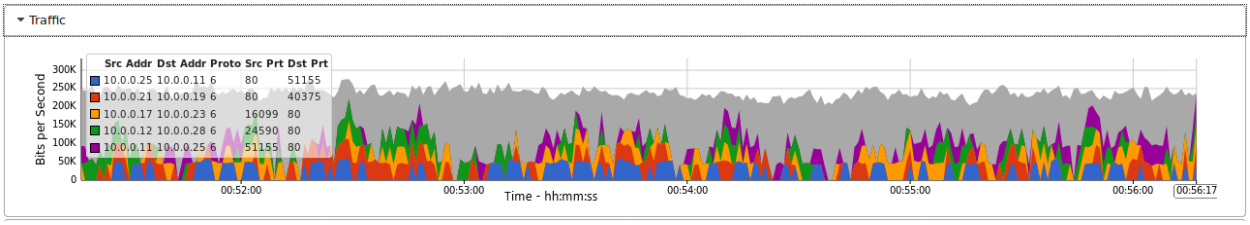

Figure 5 Normal traffic

Figure 5 shows normal traffic on the virtual network. This is before the ping flood attack is simulated. This traffic is generated by the commands nping commands shown previously.

The different colors indicate traffic from one host to the other e.g.

Blue: ICMP traffic from 10.0.0.25 to 10.0.0.11.

Red: ICMP traffic from 10.0.0.21 to 10.0.0.19

Orange: ICMP traffic from 10.0.0.17 to 10.0.0.23

Green: ICMP traffic from 10.0.0.12 to 10.0.0.28

Maroon: ICMP traffic from 10.0.0.11 to 10.0.0.25

The gray background is a default setting for the sflow graphs just showing the time the data is captured e.g. for night time traffic.

The graph shows normal conditions and traffic as per our simulation. Hosts exchanging traffic mostly via ICMP among themselves.

At this point the attack has not started yet. The switch has flow rules installed by the controller that allow traffic to be exchanged among all hosts

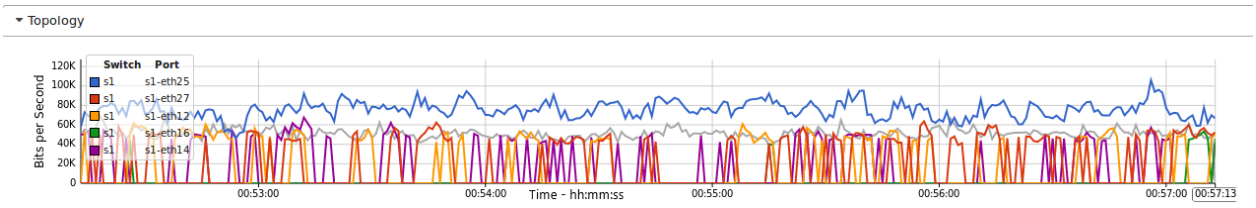

Figure 6Normal topology traffic

Figure 6 shows normal topology traffic on the simulated network. This is generated by the nping commands to simulate normal traffic.

The graph indicates traffic as seen on the mininet switch in our topology. Different colors are used for each host connected on a port on the switch.

Blue: Traffic on port 25 of the switch for host 10.0.0.25

Red: Traffic on port 27 of the switch for host 10.0.0.27

Yellow: Traffic on port 12 of the switch for host 10.0.0.12

Green: Traffic on port 16 of the switch for host 10.0.0.16

Maroon: Traffic on port 14 of the switch for host 10.0.0.14

This is traffic generated by the nping commands.

All traffic is allowed since the flow tables have not yet blocked any IP addresses. The SDN controller installed the flow tables in the switch which allow this traffic to go through.

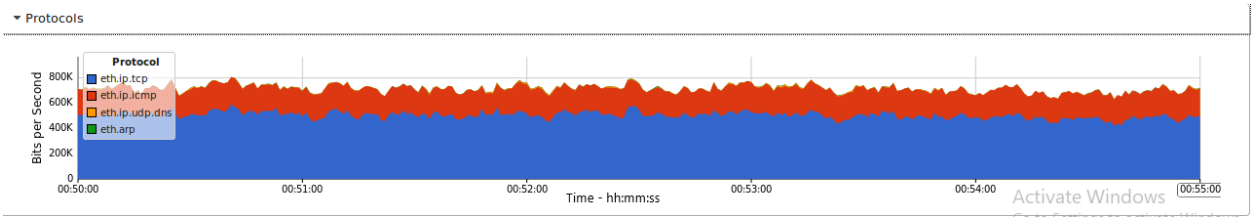

Figure 7 Normal protocol traffic

Figure 7 shows protocols running on the network. TCP traffic is the most common with ICMP, UDP and ARP as shown above.

The traffic was generated by nping commands on the various hosts. Different protocols on the network have been represented by different colors on the graph. The traffic running on the network represented by different colors is:

Blue: IP TCP traffic on the network.

Red: IP ICMP traffic.

Yellow: IP UDP DNS traffic.

Green: ARP traffic.

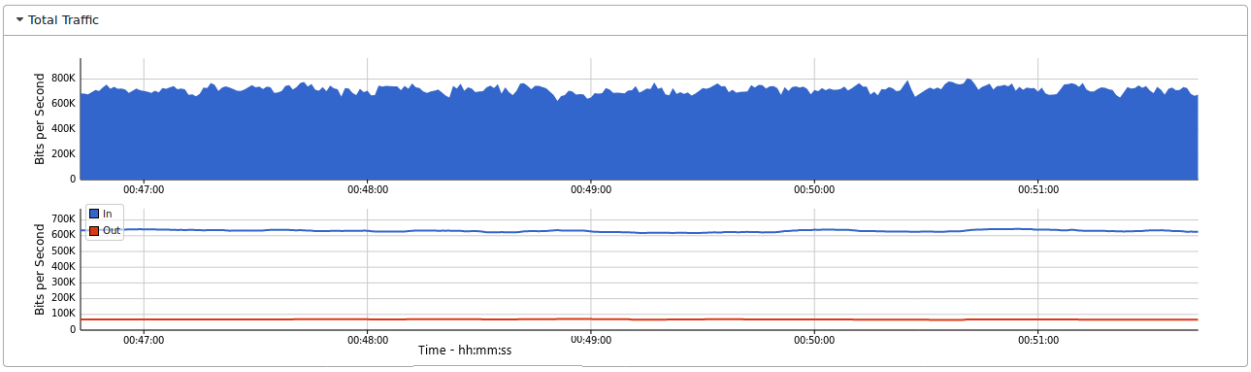

Figure 8 Normal traffic

Figure 8 shows all the total traffic on the simulated network under normal conditions.

The top graph shows total traffic which is just below 800kbps and is shown in blue graph. On the lower graph, incoming traffic is shown using a blue line at just over 600kbps and outgoing traffic is shown using a red line and is just below 100kbps.

CPU usage on the server under normal conditions

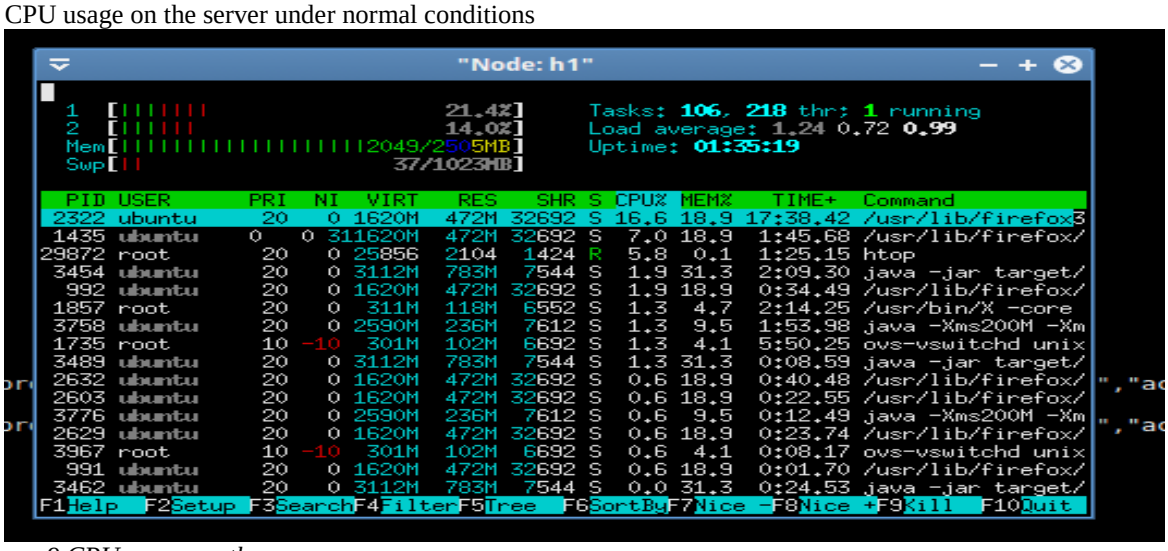

Figure 9 CPU usage on the server

Figure 9 shows the CPU usage on the web server under normal conditions. The figure shows a value of 16.6%.

Htop command was used to check resource usage on the web server. The command helps administrators monitor processes running on the system along with their full command lines.

Figure 10 htop CPU output

With regards to the above screenshot, numbers 1 to 2 are CPU cores for the web server and the progress bars next to them describe their usage. The progress bars contain multiple colors. The colors have the following meanings:

Blue: Shows the percentage of CPU used by low priority processes. (nice > 0)

Green: Indicates percentage of CPU used for processes owned by normal users.
Red: Displays percentage of CPU used by system processes.
Orange: Shows percentage of CPU used by IRQ time.
Magenta: Indicates percentage of CPU consumed by Soft IRQ time.
Grey: Percentage of CPU consumed by IO Wait time.
Cyan: Percentage of CPU consumed by Steal time.

Just below the CPU stats we have information on memory and swap usage. A similar progress bar with different colors is used. The colors have the following meanings:
Green - percentage of RAM utilized by memory pages
Blue - percentage of RAM used by buffer pages
Orange - percentage of RAM used by cache pages

Figure 11 Tasks, threads

Htop's output also indicates number of tasks and threads running on the system. In our case, we have 106 tasks with 218 threads but only 1 process is running.

Tasks represent all open processes. However, not each open process consumes CPU all the time. There are a number of states in which a process could exist:

R: Running – process is actively using CPU.
T/S: Traced/Stopped – process is currently in stopped or paused state.
Z: Zombie or defunct – process has completed execution but still has an entry in the process table.
S: Sleeping – Most common state for many processes. Generally, processes are in the sleep state for most of the time and perform small checks at a constant interval of time, or wait for user input before it comes back to running state.

The load average is the system's load average. The three values are for the last minute, last five minutes and the last 15 minutes.

Uptime value refers to system's uptime since last reboot.

Figure 12 htop detailed process information.

PID – Process ID number.
USER – process owner.
PRI – process priority as viewed by the Linux kernel.
N – Process priority reset by the user or root.
VIR – virtual memory that a process is consuming.
RES – physical memory that a process is consuming.
SHR – shared memory that a process is consuming.
S – Current state of a process.
CPU% – percentage of CPU consumed by each process.
MEM% – percentage of Memory consumed by each process.
TIME+ – time since process execution has started.
Command – full command execution in parallel to each process.

At the moment the CPU is at 16.6% indicating that the system is not overloaded. Usage for CPU cores 1 and 2 are also low at 21% and 14% respectively.

Ping Flood Attack is launched from h2 using hping3 utility.

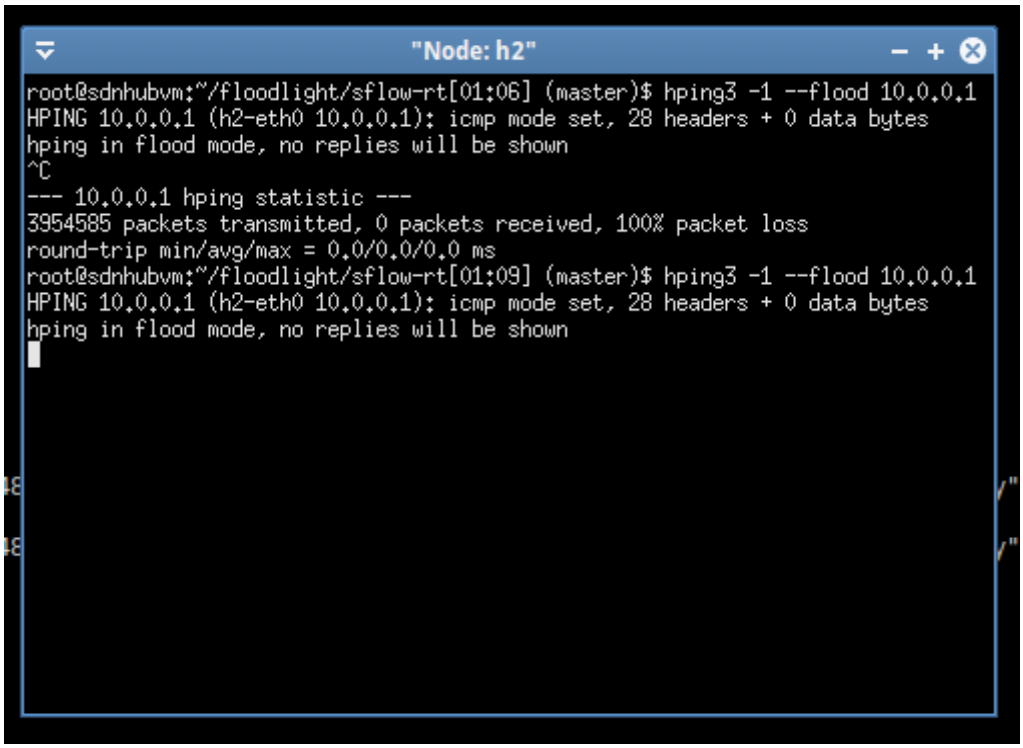

Figure 13 Ping flood attack launched

Figure 13 shows a ping flood attack initiated towards h1, the web server. The command used is hping3 -1 –flood 10.0.0.1.

10.0.0.1 is the IP address of the web server.

The attack has been sourced from host 2.

CPU usage on the server

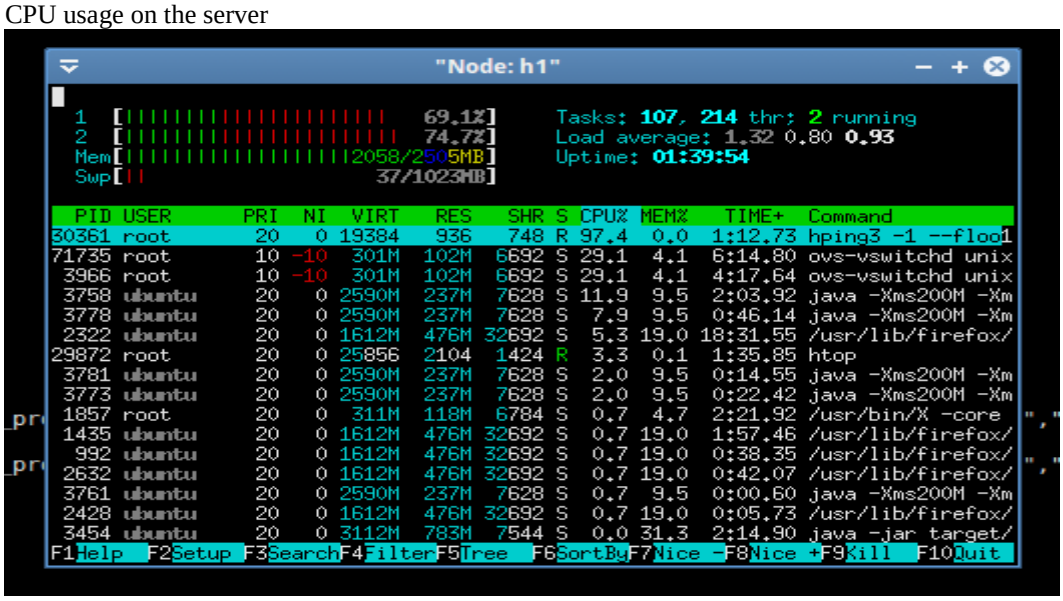

Figure 10 CPU usage on the server during the attack

Figure 11 shows that the CPU usage on the web server has shot up to 97.4% during the ping flood attack.

Both CPU cores 1 and 2 have also had their usage go up to 69% and 74% respectively. Memory and swap usage has not changed much.

Tasks and threads running also remain almost the same – 107 tasks and 214 threads with only 2 processes running at the moment.

At this stage the SDN controller has no intelligence to detect the attack and stop the same. So the web server is exposed and overwhelmed thus making the services available.

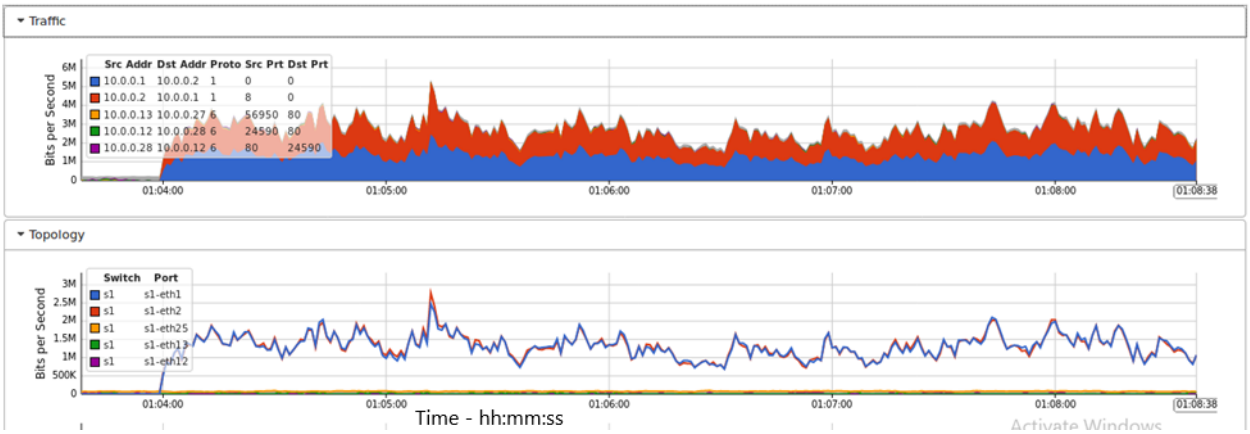

Figure 11 Traffic on network

Figure 12 shows traffic on the network spiking up when the ping flood attack is initiated.

The graph shows traffic on the network and is consistent with our simulation as it shows traffic shown in blue originating from 10.0.0.1, the web server to 10.0.0.2, the attacking machine.

Red indicates traffic from the attacking machine, 10.0.0.2 to the web server, 10.0.0.1.

The lower graph shows traffic as it passes through the mininet switch. The blue line represents port 1 on the switch on which the web server is connected and the red line is for port 2 on which the attacking machine is connected.

The two graphs also indicate a spike in traffic from the normal 600kbps to almost ten times at 6Mbps.

In this instance, the SDN did not adapt to the conditions of the network i.e. detect the DDoS and stop it while allowing normal traffic to continue flowing. The malicious ping flood continued thus overwhelming the web server hence making it unresponsive. Its services are then made inaccessible. Normal users cannot access the web page because the malicious users have made it unable to respond to legitimate requests.

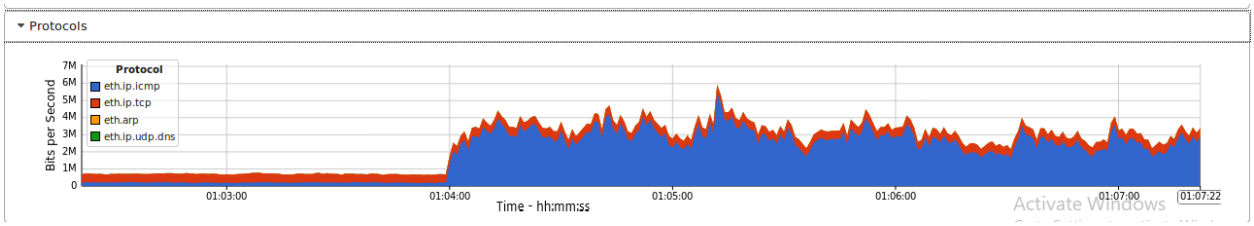

Figure 12 Protocol traffic during ping flood attack

Figure 13 shows protocol traffic on the network also increasing during the simulated attack. As indicated on the image, most of the traffic is ICMP traffic. Ping flood is based on ICMP traffic. This is shown by the blue on the graph.

Red color represents IP TCP traffic, Yellow is for ARP traffic while green indicated UDP DNS traffic on the simulated network.

The graph also shows a sudden increase in the ICMP traffic. It is at this point that the ping flood started.

The traffic continues unabated because at this moment the controller is not capable of detecting the DDoS and then stopping the same. Thus the server will end up being overwhelmed thus making the service unavailable i.e. the web service.

The SDN network still has no intelligence or an application to help it determine the legality of traffic currently traversing the network. Malicious traffic is allowed to flow through the network thus causing disruptions to other normal operations on the network. The controller does not detect the malicious traffic hence it cannot instruct the switches to drop this DDoS attack that is ongoing.

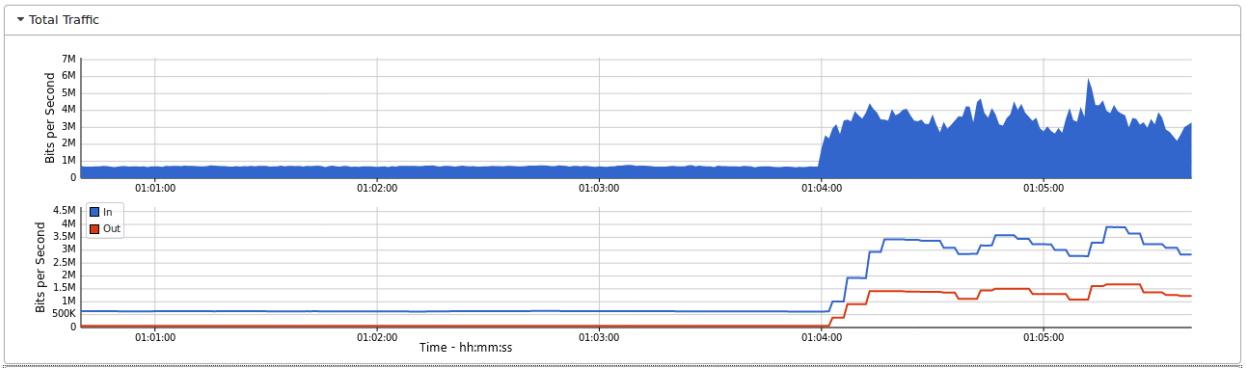

Figure 13 Traffic on network

Figure 14 is a graph of total traffic running on the network. Traffic flow was normal until when the ping flood attack was initiated leading to the traffic shooting up as shown in the image above.

The lower graph shows total incoming traffic using the blue line and total outgoing traffic using the red line.

This diagram also indicates that the SDN, being unable to detect the malicious traffic, allowed total traffic to explode and go up unabated. At this point, all traffic is seen as normal traffic since it has no mechanism to differentiate between normal and malicious traffic.

The lower section of the graph also shows traffic coming in and going out increasing. The SDN allowed all traffic to come in and be responded to by the server. If this continues, it will reach a point where the server will be overwhelmed hence making it impossible to respond to legitimate queries. This is due to inability of the controller to put a stop to the attack by instructing the switches to drop the malicious traffic.

Running the mitigation

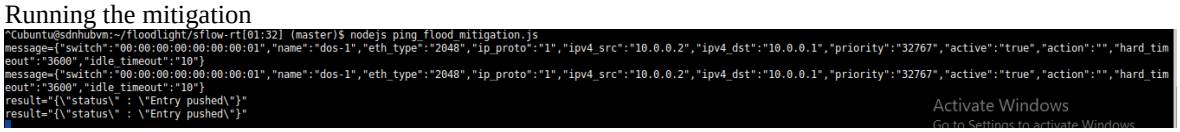

Figure 14 Ping flood mitigation script

Figure 15 shows the mitigation script being run to detect and stop the ping flood attack.

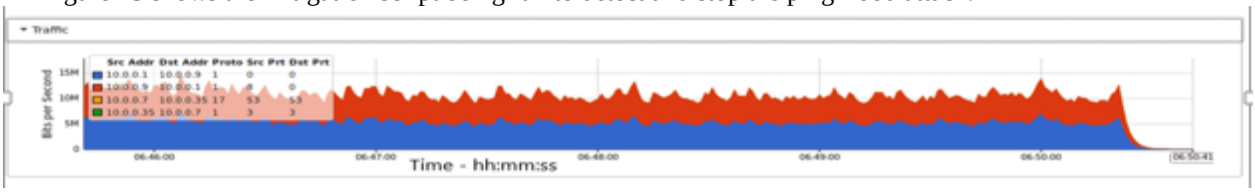

Figure 15 Traffic drop after ping flood mitigation

After the mitigation script is run, the ping flood traffic is now dropped by the switch after new flow entries are installed by the controller into the switch.

Blue indicates traffic from the web server towards the attacking machine 10.0.0.9 while red shows the reverse traffic originating from the attacking machine 10.0.0.9 destined to the web server, 10.0.0.1.

When the mitigation script is applied, the DDoS traffic drops sharply as shown in the graph. The SDN controller was able to detect the source of the malicious traffic. It then installed flows in the switch that ensure all traffic from the malicious source is dropped.

The application gives the SDN intelligence to differentiate between legitimate and malicious flows. Once malicious flows are detected, the controller installs flow rules into the switches instructing the switches to drop malicious traffic. In this case, any traffic originating from the IP address 10.0.0.9.

Figure 16 Topology traffic after mitigation

Figure 18 shows total topology traffic falling after the mitigation script is executed. The graph shows traffic on the switch. Blue line indicates traffic on port 1 of the switch. This is the port on which the web server is connected.

Red line shows traffic on port 9 on which the attacking host is connected.

Once the mitigation is executed, the SDN is able to detect the malicious traffic and drop the same thus eliminating the DDoS.

The sharp drop in traffic on the switch is a result of the switch dropping packets deemed malicious by the controller. When the DDoS attack was detected, the controller immediately created rules that instruct the switch to drop any packets from the malicious host. Thus ensuring that the server is safe and continues to operate optimally.

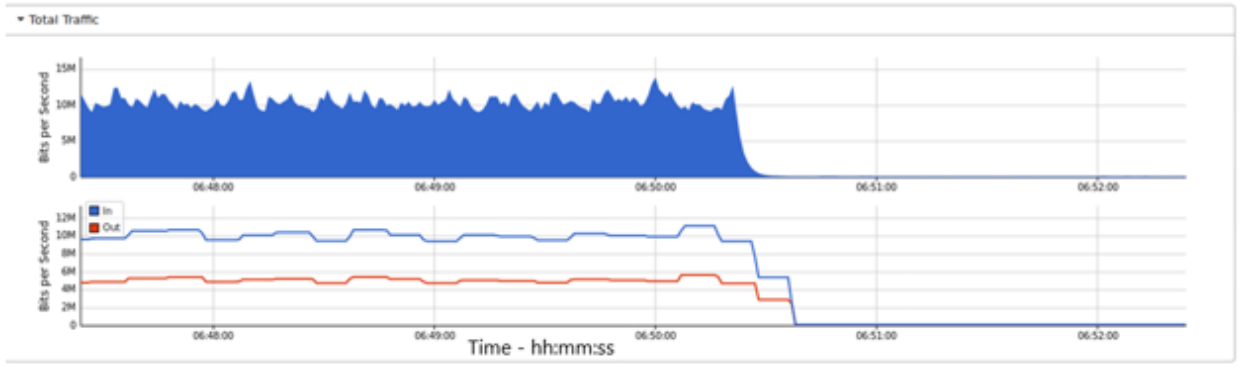

Figure 17 Total traffic after ping flood mitigation

Figure 19 shows total traffic falling after the ping flood attack has been mitigated. Most of the traffic was generated by the ping flood attack. This is shown by the graph at the top. The ping flood attack generated up to just over 10Mbps of traffic on the network.

The lower graph indicates incoming traffic using the blue line and outgoing traffic using the red line. The traffic significantly dropped after the detection mechanism was deployed. This enabled the SDN controller to detect malicious flows, install appropriate flows on the switch which made the switches drop malicious traffic thus the reduction in traffic on the network.

Figure 18 Protocol traffic after mitigation

The figure above shows protocol traffic which is mainly ICMP dropping after the mitigation has taken effect. Since we are simulating a ping flood attack, most of the traffic is ICMP and this is represented by the blue color on the graph. There is little UDP DNS traffic represented by the red color on the network.

Ping flood is composed of ICMP traffic, hence the SDN used its intelligence to detect the ping flood DDoS attack. It then installed appropriate flow rules onto the switches instructing the switch to drop any traffic from the malicious sources thus resulting in the sharp drop on the graph.

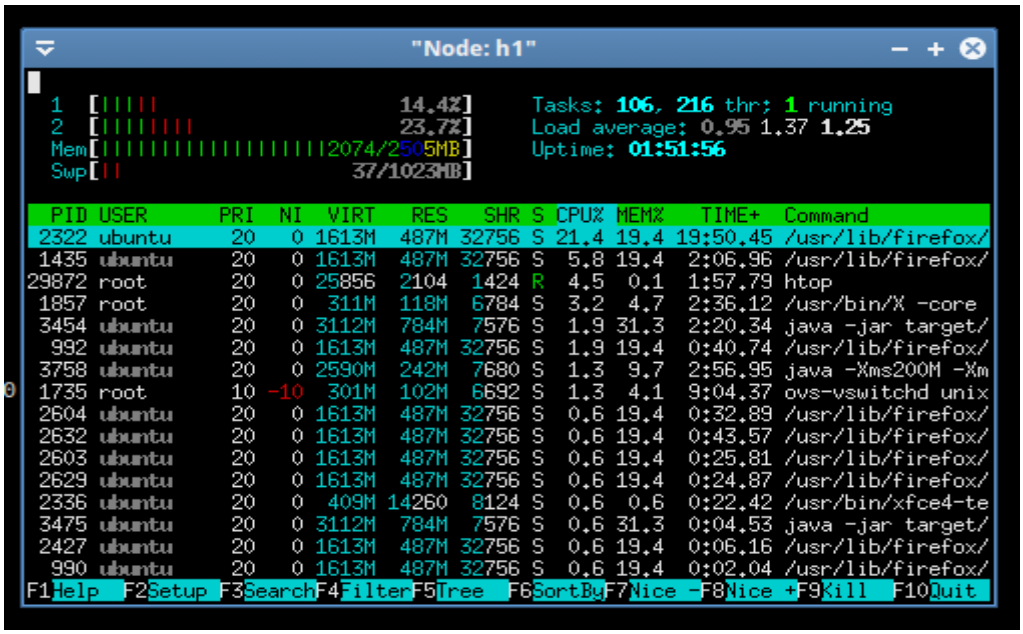

Figure 19 CPU usage after mitigation

Figure 21 shows CPU usage has gone down to normal levels. Down from 97% to 21%. This is after the ping flood attack has been detected and mitigated. Usage for CPU cores 1 and 2 has also reduced to 14% and 23% respectively. Memory usage and swap usage were not affected.

The number of tasks and threads was also barely the same.

4. Conclusion

This study was focused on detecting and preventing protocol based DoS/DDoS attacks with ping and syn flood attacks experimented with. A virtualized environment was used. The study proved that with SDN, it is possible to detect and mitigate protocol based DDoS attacks.

Further work can be done using real networking equipment. Other protocol attacks can also be studied e.g. DNS attacks, HTTP flooding etc.

Application based DoS/DDoS attacks are also an interesting field that could be explored further.

References

Cabaj, K., Wytrębowicz, J., Kukliński, S., Radziszewski, P., & Truong , K. D. (2014). SDN Architecture Impact on Network Security. *Federated Conference on Computer Science and Information Systems.* Warsaw.

Carl, G., Kesidis, G., Brooks, R. R., & Rai, S. (2006). Denial-of-service attack-detection techniques. *IEEE Internet Computing*, 82-89.

Dao, N.-N., Park, J., Park, M., & Cho, S. (2015). A Feasible Method to combat against DDoS Attack in SDN Network. *International Conference on Information Networking* (pp. 309-311). Siem Reap, Cambodia: IEEE.

Dittrich, D. (1999, December 31). *University of Washington*. Retrieved from University of Washington: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt

Floodlight. (2018). Retrieved from Floodlight: http://www.projectfloodlight.org/floodlight/

GitHub. (2017, September 03). Retrieved from GitHub: https://github.com/noxrepo/

GitHub. (2017, September). Retrieved from GitHub: https://github.com/Markus-Go/bonesi

Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., Mckeown, N., & Shenker, S. (2008). NOX: Towards an Operating System for Networks. *COMPUTER COMMUNICATION REVIEW, vol 38, no. 3* (pp. 105-110). Association for Computing Machinery (ACM).

Gupta, B. B., Joshi, R. C., & Misra, M. (2009). Defending against Distributed Denial of Service Attacks: Issues and Challenges. *Information Security Journal: A Global Perspective*, 223-248.

Haris, S. H., Ahmad, R. B., & Ghani, M. H. (2010). Detecting TCP SYN Flood Attack based on Anomaly Detection. *2010 Second International Conference on Network Applications, Protocols and Services* (pp. 240-244). Washington, DC, USA: IEEE Computer Society.

Hussain, A., Heidemann, J., & Papadopoulos, C. (2003). A Framework for Classifying Denial of Service Attacks. *Applications, technologies, architectures, and protocols for computer communications* (pp. 100-110). Karlsruhe, Germany: ACM.

INCAPSULA. (2017, September). Retrieved from INCAPSULA: https://www.incapsula.com/ddos/ddos-attacks/

inMon. (2018, 11 11). Retrieved from inMon: https://inmon.com/products/sFlow-RT.php

J. , S.-T., & Cristianini, N. (2000). *Support Vector Machines and other kernel-based learning methods.* Cambridge University Press.

Kesavan, A. (2016, November 15). *ThousandEyes*. Retrieved from ThousandEyes: https://blog.thousandeyes.com/three-types-ddos-attacks/

Kohonen, T. (1997). *Self-organizing Maps.* Secaucus, NJ, USA: Springer-Verlag.

Linux Foundation. (2017). Retrieved from OPENDAYLIGHT: http://www.opendaylight.org

Luo, S., Wu, J., Li, J., & Pei, B. (2015). A Defense Mechanism for Distributed Denial of Service Attack in Software-Defined Networks. *2015 Ninth International Conference on Frontier of Computer Science and Technology.* Dalian, China: IEEE.

Mininet. (2017). Retrieved from Mininet: http://mininet.org

Mirkovic, J., Martin, J., & Reiher, P. (2010). A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms. *Data Privacy Management and Autonomous Spontaneous Security* (pp. 220-236). Springer.

Mousavi, S. M. (2014). *Early Detection of DDoS Attacks in Software Defined Networks Controller.*

netem. (2018). Retrieved from netem: https://wiki.linuxfoundation.org/networking/netem

NOX. (2018). Retrieved from NOX: https://github.com/noxrepo/nox

ns-3. (2018). Retrieved from ns-3: https://www.nsnam.org/

ONF. (2017). *Open Networking Foundation*. Retrieved November 10, 2017, from https://www. opennetworking.org

ONOS. (2018). Retrieved from ONOS: https://onosproject.org/

Open Networking Foundation (ONF). (2017, November 10). Retrieved from Open Networking Foundation (ONF): https://www. opennetworking.org

OpenDaylight. (2018). Retrieved from OpenDaylight: https://www.opendaylight.org/

Pescatore, J. (2014, February). *SANS.* Retrieved from SANS: https://www.sans.org/reading-room/whitepapers/analyst/ddos-attacks-advancing-enduring-survey-34700

Phan, T. V., Bao, K. N., & Park, M. (2016). A Novel Hybrid Flow-based Handler with DDoS Attacks in Software-Defined Networking. *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress.* Seoul: IEEE.

Python Standard Library. (2017, September). Retrieved from Python Standard Library: https://docs.python.org/2/library/random.html

Rana, D. S., Garg, N., & Chamoli, S. K. (2012). A Study and Detection of TCP SYN Flood Attacks with IP spoofing and its Mitigations. *International Journal of Computer Technology and Applications*.

Rodrigo, B., Edjard, M., & Alexandre, P. (2010). Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow. *35th Annual IEEE Conference on Local Computer Networks*, (pp. 405-416). Denver, Colorado.

Rodrigo, B., Edjard, M., & Alexandre, P. (2010). Light-weight Distributed DoS Flood Attack Detection Using NOX & OpenFlow. Denver, Colorado: 35th Annual IEEE Conference on Local Computer Networks.

Scapy. (2017, September). Retrieved from Scapy: http://www.secdev.org/projects/scapy

sflow. (2017, September). Retrieved from sflow: http://www.sflow.org

sflow. (2017, September). Retrieved from sflow: http://blog.sflow.com/2011/01/presentation.html

Srivastava, A., Gupta, B. B., Tyagi, A., Sharma, A., & Mishra, A. (2011). A Recent Survey on DDoS Attacks and Defense Mechanisms. *Advances in Parallel Distributed Computing* (pp. 570-580). Springer.

Sutter, J. (2009, August 6). *CNN.* Retrieved from CNN: http://www.cnn.com/2009/TECH/08/06/twitter.attack/index.html

Virtualbox. (2018). Retrieved from Virtualbox: https://www.virtualbox.org/

Wesley M. Eddy, V. F. (2006, December). *Cisco.* Retrieved from Cisco: https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-34/syn-flooding-attacks.html