

Encountering distributed denial of service attack utilizing federated software defined network

Rima Abdelhadi, Moath H. Alsafasfeh, Bilal I. Alqudah

Department of Computer Engineering, Faculty of Engineering, Al-Hussein Bin Talal University, Ma'an, Jordan

Article Info

Article history:

Received Apr 23, 2023

Revised Jul 10, 2023

Accepted Jul 17, 2023

Keywords:

Detection and prevention
Distributed denial of service
attack

Distributed solution

Quality of service

Software defined network

ABSTRACT

This research defines the distributed denial of service (DDoS) problem in software-defined-networks (SDN) environments. The proposed solution uses Software defined networks capabilities to reduce risk, introduces a collaborative, distributed defense mechanism rather than server-side filtration. Our proposed network detection and prevention agent (NDPA) algorithm negotiates the maximum amount of traffic allowed to be passed to server by reconfiguring network switches and routers to reduce the ports' throughput of the network devices by the specified limit ratio. When the passed traffic is back to normal, NDPA starts network recovery to normal throughput levels, increasing ports' throughput by adding back the limit ratio gradually each time cycle. The simulation results showed that the proposed algorithms successfully detected and prevented a DDoS attack from overwhelming the targeted server. The server was able to coordinate its operations with the SDN controllers through a communication mechanism created specifically for this purpose. The system was also able to determine when the attack was over and utilize traffic engineering to improve the quality of service (QoS). The solution was designed with a sophisticated way and high level of separation of duties between components so it would not be affected by the design aspect of the network architecture.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Rima Abdelhadi

Department of Computer Engineering, Faculty of Engineering, Al-Hussein Bin Talal University

King Hussien Bin Talal University Str, Ma'an, Jordan

Email: Alqudah@ahu.edu.jo

1. INTRODUCTION

The use of online services is increasing, making it easier for attackers to find targets to achieve their goals. A very effective attack on online services is the denial-of-service attack (DoS) or the distributed form of it (DDoS). This attack can last for days, weeks, or even months, and can cost companies in average about \$1.6 Million USD and others about \$20,000 USD per day. DoS and DDoS attacks aim to prevent employees and customers from reaching services provided by the servers and can have catastrophic consequences if the attack takes place in servers providing healthcare services or emergency services [1]. On October 21, 2016, a stream of distributed denial of service (DDoS) attacks involving tens of millions of internet protocol (IP) addresses had been noted and attacked dynamic domain name system (DNS). The magnitude of the attack was claimed to be 1.2 Tbps and it involved internet of things (IoT) devices. To restore trust, researchers should investigate different methodologies to limit the consequences of the attack on victims and providers [1], [2]. The following is a summary of some techniques used to mitigate the danger of DDoS. Some studies categorized the attacks by the nature of the targeted part of the system, such as cisco, where it classified the attacks into: i) volume-based DDoS attacks: the attack targets server resources and facility bandwidth, networking equipment; ii) application DDoS attacks: this type of attack overwhelms services (*i.e.* voice over

IP (VoIP), hypertext transfer protocol (HTTP), domain name servers (DNS)); and iii) low-rate DoS (LDoS) attacks: this type attacks do not flood the network with traffic; however, it targets applications weaknesses such as (Slowloris) attack. However, Kaspersky security company classified DDoS attacks based on the method attack follow rather than what does the attack target on the victim system as a weakness. Kaspersky specified the following steps to protect against the attack: i) tolerate a web-server configuration against DDoS attacks, ii) alter an internet service providers (ISP) firewall to allow only the traffic complimenting to the services on the company side, iii) tweak a firewall to fight synchronization connection (SYN) flood attacks, iv) migrate public resources to another IP address; and v) relocate all business-critical applications to the cloud or move to the separate public subnet.

2. LITERATUR REVIEW

A denial of service (DoS) attack attempts to prohibit a client from accessing a certain service or to reduce the quality of the services provided. The severity of the attack is determined by the volume of traffic directed towards the victim. It is a one-to-one attack in the case of a DoS attack.

Hatagundi and Kumaraswamy [4] provided a survey for possible attacks in software-defined-networks (SDN) networks and the possible methods of mitigation. The study specified four goals to achieve:

- Analyze various attack on the SDN environment, including DoS and DDoS.
- To determine the state of the system by measuring the unpredictability or entropy in switches and controllers using statistical methods.
- To determine whether the system is under attack based on a predefined entropy threshold.
- To compute bandwidth utilization by each switch, assign a priority, and arrange packet flow using time slice allocation algorithms depending on the priority.

The methodology for mitigating DDoS attack is based on first identifying DDoS using an entropy detection attack and then mitigating it using a bandwidth prediction method. When a time slice allocation approach is implemented, the findings reveal that the switches with higher priority receive better time allocation. However, the authors encountered certain difficulties, such as the fact that OpenFlow security features typically incorporate more sophisticated logics rather than simplification and packet pausing or forwarding. The security of the OpenFlow feature necessitates the adoption of a controller that is platform dependent, followed by the creation of a security application that is platform dependent.

Dong *et al.* [5] provided a survey about the distributed nature of the DoS attack and the effect of the advances in computational ability in empowering DDoS attacks on the SDN itself. They also studied the cloud architecture of SDN. From the design point of view, the design of SDN represented in three plans, application plan: where the user applications run. The control plan: where the controllers reside and manage connected devices. The last plan is the data plan that represents the managed routers by the second plan. SDN architecture and design aspects described in detail in many literature [6]–[10], When compared to traditional networks, SDN offers the following advantages:

- Global authority: The controller has a thorough understanding of the network topology, network state, and application requirements.
- Adaptability and programmability: The data plane can be customized flexibly to enhance network resource allocation.
- Availability: The controller's communications with forwarding devices are not reliant on the device providers.

In the case of a DDoS attack, many networks or bots will flood the victim with traffic [11]. Canadian Institute for Cybersecurity at University of Brunswick came up with a new taxonomy for DDoS attacks that can be done using user datagram protocol/transmission control protocol (UDP/TCP) at the application layer. The classification came up to cover new possible attacks [12], [13].

The problem of DDoS attack in SDN covered from many points of views, Liu *et al.* [14] provided a survey covering the security issues in SDN from north to south, the authors introduced the background, architecture and working process of SDN, and summarized and analyzed the typical security issues from north to south. They also review and analyzed existing solutions and research progress of each layer, such as authorized authentication module, application isolation, DoS/DDoS defense, multi-controller deployment and flow rule consistency detection.

Chen *et al.* [15] focused on his research on increasing the efficiency and decreasing the latency when preventing a DDoS attack targeting SDN infrastructure by utilizing decision trees (DT), they called it DETPro. The intention is to make the framework lightweight through deciding what to do based on DT. They addressed the issue by proposing a DDoS attack detection module based on a modified decision tree algorithm that is both accurate and quick to process. The proposed DDoS attack mitigation module includes a dynamic whitelist system that can block attack traffic while forwarding benign traffic in real time. Then they

combined them in a "time-efficient, lightweight system" (DETPro) that incorporates both methodologies. DETPro can detect DDoS attacks accurately in the early stages, defend against DDoS attacks effectively, and safeguard the network, according to the testing results from the research they presented.

Sanjeetha *et al.* [16] focused in their research on DDoS attacks on primary victim servers flooded by high traffic. DDoS attack can be instigated on a primary server present in SDN during high traffic scenarios. The purpose of their research is to show how attackers can use a mechanism designed to deal with flow table space exhaustion in high-traffic conditions to conduct a DDoS attack on a primary server via the controller. They next attempted to detect and separate the assault flow from normal traffic.

Finally, they safeguard the core server against controller-initiated attacks and resource risks. Their work is mostly focused on improving DDoS categorization methods, with the following contributions: First, after being trained by a self-organizing map (SOM). They offer two approaches with different trade-off priorities for classifying normal and DDoS attack traffic a SOM distributed-center classification approach that is employed when SOM training is complete. This heuristic technique achieves a quick processing time while maintaining an acceptable level of accuracy. Second, they offer an SDN-based DDoS attack detection platform that has been built in a testbed utilizing SDN switches. This testbed allows the system to estimate the accuracy and processing time of several algorithms, as well as compare them.

Nam *et al.* [17] focused in their study on identifying DDoS attacks, which have been a long-standing issue in network security. They used two approaches with distinct tradeoff priorities to categorize regular and DDoS attack traffic. To have a faster processing time and a more precise detecting method, as well as to assign different priority than existing detecting algorithms, they included four new SDN controller modules: monitor, algorithm, alert, and DDoS mitigation, as well as a lightweight SDN-based DDoS detection and mitigation architecture. The entropy of the source IP address (etpSrcIP), the entropy of the source port (etpSrcP), the entropy of the destination port (etpDstP), the entropy of the packet protocol (etpProtocol), and the total number of packets were then examined and categorized to detect distinct types of DDoS assaults (totalPacket). When given different priorities, traditional detection algorithms have been demonstrated to perform worse than the proposed techniques. They addressed some of the issues that researchers investigating DDoS attacks encounter, such as the similarities between DDoS attack and regular traffic patterns, the spread of big DDoS attacks powered by IoT bots, and the attributes that are manually picked for each type of host and each type of attack.

Ubale and Jain [18] investigated the SDN centralized functionality issue, which makes it vulnerable to TCP SYN flood attacks, which degrade server resources and transform the controller into a single point of failure by exhausting controller resources. Additionally, it causes a data plane saturation attack. The researchers used two functional modules: the hashing module and the flow aggregator module, to avoid TCP SYN flood DDoS attacks in SDN settings, address security issues such as ternary content addressable memory (TCAM) exhaustion attacks, and overcome the disadvantages of standard SDN, and the (OPERETTA and SLICOTS) modules that is implemented in the controller to investigate SYN flood attack. They addressed the problem by first proposing their (SRL) hashing module, a competent and streamlined framework for mitigating TCP SYN flood attacks that is implemented in the controller, then installing permanent forwarding for requested connections and after handshake completion SRL moves user to Whitelist and using hashing module to replace flow rules from flow table according to priority of hash value. A flow aggregator was used to stop the malicious connection requests. They were able to detect a slow rate DDoS attack by limiting TCP connection requests. Finally, SRL was implemented in the Floodlight controller under various attacking and usual traffic scenarios. According to their research, SRL has only a minor impact on SDN controller operations. If there was a difficulty, it permitted real users to connect, as well as recognizing malicious users who conducted a complete TCP handshake before beginning an attack. Using a prototype implementation of SLR, SRL performance and efficiency are compared to SLICOTS, OPERETTA, and standard SDN, revealing that SLR has a lower service delivery time for requests under attack, lower CPU utilization caused by regular traffic with many users, a lower number of real entries getting replaced before the attack, and the lowest DDoS attack detection rate. The main challenge is determining the threshold value; if it is too high, attack detection may be slowed, and if it is too low, erroneous findings may occur.

Aryal *et al.* [19] proposed a model for detecting DDoS attacks in 5G networks using SDN. A hybrid method is used, with five separate machine learning classifiers used to calculate a dynamic threshold. The proposed strategy is a hybrid approach that combines statistical analysis and machine learning. In an iterative process, data sets are evaluated and compared to a dynamic threshold. Sixteen features are retrieved, and correlation values between the features are examined using machine learning. With the deployment of SDN, the system provided dynamic configuration with software defined security (SDS). The traffic is constantly evaluated to improve the controller's efficacy in terms of task handling capacity. In terms of accuracy, precision, and efficiency, the suggested method outperforms current conventional approaches. In terms of

outcomes, machine learning is also being used to improve detection precision. This improves accuracy from nearly 87% to 99.86% while lowering the false positive rate (FPR). The findings achieved using experimental data sets outperformed previous methods. Different machine learning techniques were employed for poor outcomes, and the findings showed that their model surpassed others in terms of accuracy and precision.

Sharma [20] research discussed that configuring detection and prevention strategies were the current remedies for the DDoS attack. All solutions require pre-configuration and cannot be modified or accepted in response to changes in the network or the nature of the attack. This study attempts to propose a DDoS solution based on SDN architecture. Their goals were to allow networks to regulate their behavior and take appropriate measures when they were attacked, to measure efficiency and resilience against DDoS attacks and false policy-based attacks by comparing the architecture to filter-based intrusion prevention architectures, and to provide real-time monitoring and reactive service by combining management applications at the control plane in SDN with multi-agent system (MAS) and statistical modeling. Then it aimed to use the programmatic interface of SDN coupled with MAS to represent and reason knowledge about network topology and state, then to identify and avoid potential threats by using data flows rather than individual packets, and finally to control and manage large-scale networks with the least amount of downtime. The authors outlined the difficulties encountered during the research as follows: it is hard to recognize a real-world example in which an attack was discovered in SDN networks due to a lack of availability. Comparative research employing a common benchmark that has yet to be developed or established. The suggested agent-based self-mitigating architecture takes use of trade-offs between the timeliness and volume of load data exposed, as well as the efficiency and granularity of the control input required for quick reconfiguration.

Hyun *et al.* [21] maintain server stability by limiting the number of packets entering the server per second, without discriminating between legitimate users and botnets. Per the findings, network administrators and web administrators can use the SDN controller to adjust network packet flow based on the company's overall environment and specific network environment. Kalliola *et al.* [22] studied the DDoS flooding attack and the method to mitigate it with SDN by managing traffic. They established an approach based on learning traffic values and allocating appropriate resources to withstand the attack. The proposed approach is designed to protect both end hosts and network links from packet and bandwidth flooding assaults. The technique in this study allows optional input from external signature-based blacklist sources, as well as traffic management. The report stated that even when confronted with an overwhelming attack, the efficacy in preserving service was in the range of (50% to 80%). Blacklist integration and level of protection are two of the research's challenges: the method permits the integration of fixed or dynamic blacklists consisting of particular IP addresses. Because the defensive mechanism functions at the IP layer, attacks targeting higher levels are not covered.

Manso *et al.* [23] develop and deploy a software-defined intrusion detection system (IDS) that identifies and mitigates attacks at their source, assuring network infrastructure availability and sustaining the QoS even while an online service is already under attack. Their IDS detects DDoS-based cyber-attack scenarios automatically and limits them at the client level, reducing the negative repercussions of the attack's wider effect for possible victims, and then notifies SDN controller as soon as an attack is discovered. Their method additionally sends from the SDN controller to network devices some essential traffic forwarding decisions. An event is created whenever a substantial change in the traffic trend is noticed; if an event linked with a DDoS attack is triggered, an SDN controller produces flow rules to restrict the malicious traffic. According to the findings, their method right away detects a variety of DDoS-based cyber-attacks, mitigates their negative impacts on network performance, and ensures that normal traffic is delivered correctly while neutral traffic is protected. Their defensive technique would be most successful if it was installed as near to the attack source that creates rogue traffic as feasible, which would need coordination among many service providers to authenticate packet source addresses and apply other flow-based filtering capabilities. This analysis might be carried out by a federation of SDN controllers, one for each service provider.

Mousavi and St-Hilaire [24] focused on the hypothesis that SDN could be a single point of failure that could be attacked by DDoS. Such controller attacks will deplete the resources of the intended victim. The proposed solution is a potential tool for early detection (when more than 500 packets are assumed as attack traffic threshold). The researchers relied on incoming traffic's randomness, where entropy refers to the probability of an event occurring given the total number of events, which in this case is incoming packets.

The proposed method is based on assessing the network's randomness; if one machine receives more traffic (regardless of the source of the traffic), the entropy will fall due to a lack of randomness on the network. The following two factors influenced their methodology; the Windows definition: where either the number of packets or the time interval were described as Windows, and the Threshold: within this window, entropy is calculated to determine the degree of uncertainty in the incoming packets. According to the study, the proposed method can detect DDoS attacks within the first 500 packets of attack traffic and can recognize

both the controller and the targeted host. In a dynamic environment, human involvement is essential. Each time the network configuration changes, a new threshold and entropy must be calculated. The threshold is chosen experimentally, which implies that if the threshold is incorrectly set, several attacks may succeed, or legitimate traffic may be blocked.

Sahay *et al.* [25] proposed an automatic DDoS attack mitigation utilizing SDN. Customers can request DDoS mitigation services from ISPs using a distributed collaborative framework. ISPs can modify the label of aberrant traffic and divert it to security middle boxes upon request, while attack detection and analysis modules are deployed at the customer's end, avoiding privacy breaches and other legal issues.

3. PROBLEM FORMULATION

The network is illustrated using conceptual representation of SDN box (controller and switches). Connection between controllers does not mean a direct hard wire between them, but it represents a communication ability. Network representation or topology did not follow one common network architecture or design such as star or tree or any single architecture.

The representation will be a mix between them to show the problem with many possible complications. As shown in Figure 1, a victimized server located in network (v) is being attacked by bots located in networks (A) and (B). There are many possible routes between attacking bots and victimized server. Traffic oriented to victimized server will go through many SDN nodes that have enough capabilities to process traffic and communicate.

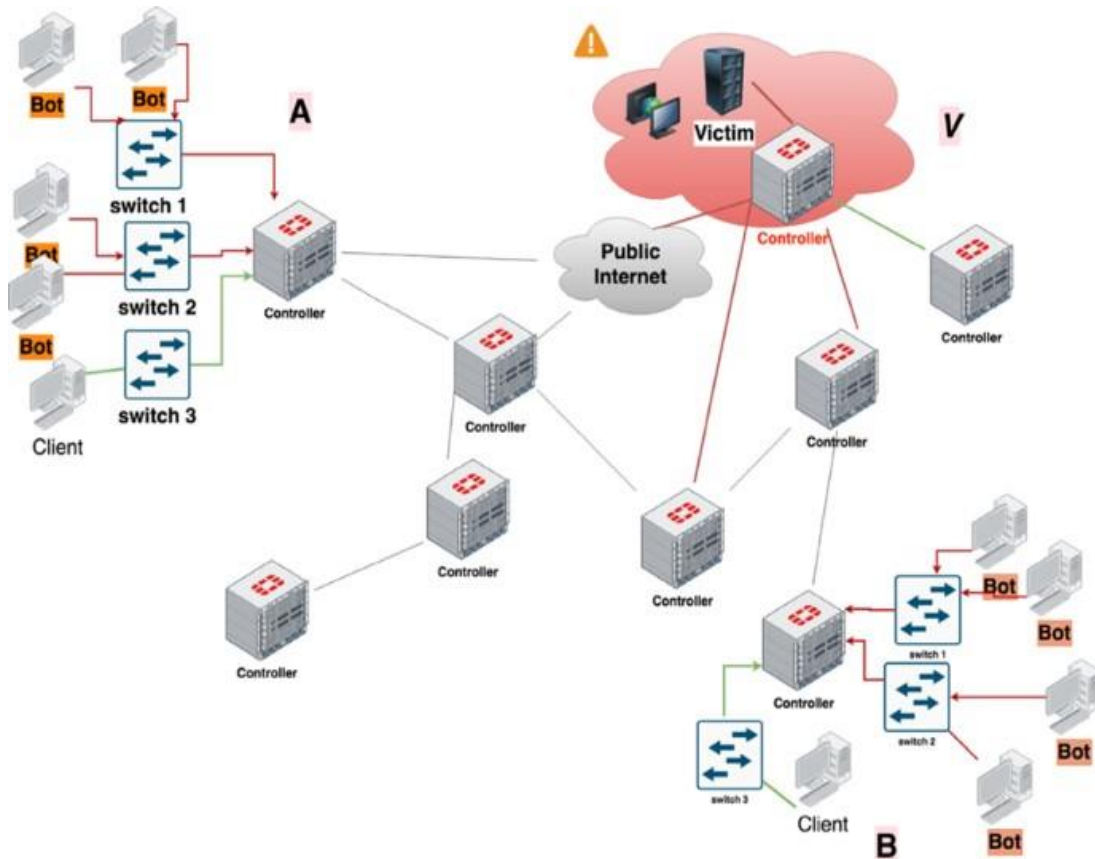


Figure 1. Scenario of DDoS attack in SDN managed network

Assuming a victim (v) connected to internet with many routes available towards it from outside. Some attacker (E) located somewhere in a remote network. The attacker managed to prepare many bots (bn) where (n) is the network ID the bot (b) belongs to. Each bot (b) is generating oriented traffic (tn) towards (v). On the path to (v), generated packets/traffic will be routed by many switches or routers, both are denoted as (sRd) where (R) is the router ID connecting (s) the source network ID and (d) the destination network ID.

Despite that all routers are transferring malicious traffic from bots to victim (v), routers can do nothing but passing the traffic if not classified as malicious traffic (traffic to drop).

In one hand, the installed firewall on the victimized side must deal with all incoming traffic. On the other hand, network must be busy with malicious traffic (t_x) that should not be transmitted through the network. The traffic directed to the victim machine (t_v) is explained in (1), where the (t_x) is the attack traffic from network x and (t_l) is the legitimate traffic from any link l .

$$t_v = (\sum_{x=1}^n(t_x)) + (t_l) : t_l \text{ is the Legitimate traffic towards } v \text{ from any link "l"} \quad (1)$$

The machine will be under attack when the sum of illegal traffic (t_x) is much more than the legitimate traffic (t_l) as shown in (2). although, it worth noticing that the legitimate traffic effect might be high in some applications, such as video broadcasting. However, attackers still send a very high volume of traffic to overwhelm victim servers.

$$(\sum_{x=1}^n(t_x)) \gg (t_l) : (t_x) \text{ is the attack traffic volume from network "x"} \quad (2)$$

Our goal will be minimizing the total (t_x) to maximize the chance of (t_l), which is the legal traffic, to reach its destination (v). If that goal achieved, then the risk of the attack is minimized, and the victim (v) will have enough time to heal and recover from the attack while continue to provide service. The following section will explain the problem when SDN components used rather than regular switches and routers. If the distance between any group of bots attacking the victim (v) is given by (d_b) measured by number of hubs or controllers, traffic (t) will be much higher when getting closer to the victim (v), as explained in (3).

$$(t_{x1} : d_b) < (t_{x2} : d_b - 1) < \dots < (t_{xn} : d_b - n_b - 1)$$

$$\text{These yields } \rightarrow (t_{x1} : d_b) \ll (t_{xn} : d_b - n_b - 1) \quad (3)$$

where (t_x) is the malicious traffic toward (v) from bots (b), d_b-n is the distance getting less when (n) increase. The reason for this accumulation of traffic is the fact that many nodes will start to join traffic in the nodes close to the victim. We will be using this phenomenon to trace back the malicious traffic to the source.

4. PROPOSED SOLUTION

In our proposed solution, detection and prevention for DDoS attacks can be initiated either from the server side or from the controller side. Figure 2 shows the flow chart for both cases of detection and prevention. To coordinate and manage the attack mitigation process, we provided software called "agent". The agent job is to take discussions based on statistics provided by controllers. In the server-side agent, depends on frequent readings for metrics (CPU utilization, memory consumption and network traffic). If measurements showed that utilization levels are near to the maximum available resources' utilization with a very high network use, it will be an indication of DDoS attack, then the victimized server agent will notify the edge SDN controller by a flag representing possible positive diagnosis. Figure 2(a) shows the flowchart for detection and the stage where the server asks for the prevention process to be started by edge SDN controller. Servers normally suffer from the DDoS attack before getting detected by the network since the attack is coming from several sources.

The edge SDN controller is capable of doing diagnosis by measuring external traffic and bandwidth consumption against maximum known history. If there was a high traffic rate much higher than the usual known behavior or close to the maximum bandwidth available, then the controller is suspecting a DDoS attack and asking the server to initiate self-evaluation process and waits for its response, this detection flow can work effectively when victimized servers are powerful enough and detect the attack late. Figure 2(b) shows the flow chart for the designed detection process. However, the edge SDN controller will ask the victimized server to confirm the result the controllers found regarding the potential attack.

The evaluation process on the victim server (VSe) is expressed by (4), since the DOS attack goal is to prevent the server from providing service, we specified a threshold for each studied metric considered as the upper boundary for resource utilization before considered over-utilized.

$$VSe = (CPUu > maxCPUThreshold \ || \ MEMu > maxMemThreshold) \ \& \ NWu > maxNetworkThreshold \quad (4)$$

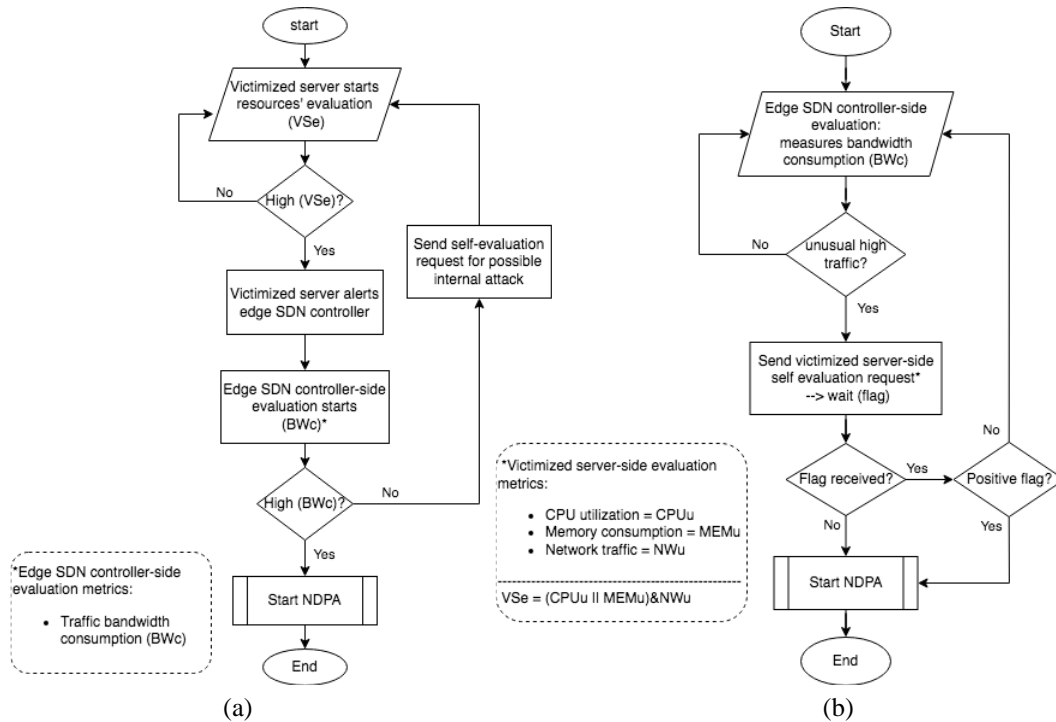


Figure 2. Prediction and detection process starting, (a) from the victim side, (b) from edge SDN controllers

5. ALGORITHM

Algorithm 1 shows the victimized server-side agent. The server agent job is to evaluate the victimized server status and communicate with the edge SDN controllers (the controllers whose job is to manage the routers and switches connected to the server or servers subject to attack “victimized”). Lines (1-6) sets the acceptable thresholds on the simulated network. Each network has its own settings based on the nature of services running on the network; for example, gaming network has higher CPU usage than a network with database servers. However, a network with video streaming services will consume memory and network more than CPU. The variable “maxRound” in line 5 sets the maximum number of evaluations allowed for the server before sounding alters regarding a possible DoS attack.

Algorithm 1. Victimized server-side agent

```

ServerAgent():
1. Set alertForPossibleAttack=FALSE
2. Set CPUThreshold=0.95
3. Set MemThreshold=0.95
4. Set NetThreshold=0.80
5. Set maxRounds=100
6. Set recoveryTime=5000ms
7. Set round=0.
8. While (true)
9. networkUtilize= NetworkThroughput/Networkbandwidth
10. CPUutilize=getCPUUtilization()
11. MemUtilize= getMemoryUtilization()
12. If ((CPUutilize>CPUThreshold) || (MemUtilize>MemThreshold) && (networkUtilize>NetThreshold))
12.1. round++.
12.2. if (round > maxRounds)
12.2.1. round=0
12.2.2. alertForPossibleAttack=TRUE
12.2.3. If (EdgeControllerAgent.HighTraffic==FALSE) raise(internalAttack)
12.2.4. If (EdgeControllerAgent.HighTraffic==TRUE) && alertForPossibleAttack
Then
ControllerResponse= EdgeControllerAgent.C2C()
12.2.5. Wait(recoveryTime)
13. Else
13.1. alertForPossibleAttack=FALSE
14. wait(10ms)
15. end while
  
```

In line 12, the algorithm for the server agent tests the current CPU utilization, memory utilization, and network utilization against the maximum values expected. If either CPU or memory are highly consumed with a high network usage, the algorithm will count that as a positive sign (line 12.1). in line (12.2) if the total positive checks exceed the maximum allowed threshold, the line (12.2.2) will set the flag for possible attack to (TRUE) to indicate a possible attack. The algorithm will stop the evaluation for a while (line 12.2.6) after alerting the controller managing the local network routers and switches. This “wait time” assumes that the controller will take an action and allow that action enough time to reflect on the server status. This time is adjustable as well.

In line (12.2.3) the server will get a feedback form the controller. The received feedback represents the network status of the outer network (outside the facility network). If the outer network has no high traffic and the server still getting high traffic from somewhere, then the server will raise a flag for a possible Internal DoS attack. In such case, the server must check for data theft (or very high information transfer rate) and investigate its legality.

Algorithm 2 describes the agent that runs on the SDN controllers participating in the network detection and prevention process. The edge SDN controller agent’s algorithm checks the network traffic status by calling the predict high traffic algorithm (Algorithm 3) in line 2. If there was high traffic, the edge SDN controller agent’s algorithm asks the victimized server if there was a possible attack alert due to a continuous increase in traffic detected and waits for its response. This notification will trigger the process of attack detection by calling the C2C algorithm (Algorithm 4) in line 3.1 which calls NDPA algorithm (Algorithm 5).

Algorithm 2. SDN controller-side agent

```
SDNControllerAgent
1. while (TRUE)
2. HighTraffic ← PredictHighTraffic()
   2.1. if (HighTraffic)
       2.1.1. serverResponse=getServer(alertForPossibleAttack)
3. if (serverResponse=TRUE or NULL)
   3.1. Call C2C( )
4. Wait(delay)
5. End while
```

Algorithm 3. Predicting high traffic

```
1. set MaxHistory [NoOfManagedNetworkingDevices]= {0}
2. set HighTraffic=FALSE. ← public
3. set MaxTotal=0, maxSmampleCount=100, delay=1000ms
4. set trafficToleranceFactor =w
5. set counter=0
6. set static oldHistory=0
7. while (true)
8. for each managed networking device
   8.1. MaxHistory[device(i)] = device(i).getmaxtraffic
   8.2. MaxTotal=MaxTotal+ MaxHistory[device(i)]
9. If (OldHistory + trafficToleranceFactor <MaxTotal)
   9.1. counter=counter+1
   9.2. if (counter >maxSmampleCount)
       9.1.1. set HighTraffic=TRUE
       9.1.2. counter=0
10. Wait(delay)
11. oldHistory=MaxTotal
12. End while
```

Algorithm 3 predicts if there was unusual high traffic referring to network device traffic history and pre-set traffic threshold. Lines 1-6 of the algorithm is setting the initial values; each network can be configured based on its nature. For example, if the network is designed to carry backups at a certain time, then traffic tolerance factor (line 4) can be adjusted dynamically to avoid wrong detections. The *maxSampleCount* parameter controls the sampling lifecycle. The *MaxTotal* parameter keeps track of the maximum traffic known and detected in the history of the detection cycle for all devices the controller manages. The algorithm in line 8 iterates over each managed networking device (*i.e.* routers and switches) the controller manages and get the maximum traffic passed through the device and store it regardless the source or the destination.

Line 9 checks the *maxTotal* for the current round of sampling in addition to a pre-defined tolerance factor with the maximum known history. If the new readings are bigger than the history, then the algorithm starts counting that towards a cycle of detection (lines 9.1 and 9.2). Algorithm 4 calls the predict high traffic algorithm (Algorithm 3), if the return value was true which means high traffic detected, C2C algorithm will

call the network detection and prevention agent's algorithm NDPA (Algorithm 5) and it will return a report of the network device's ID that has the high traffic to the calling algorithm.

Algorithm 4. Controller to controller communication algorithm

```

C2C( ):
1. highTrafficSources[]=null
2. while (TRUE)
3. HighTraffic ← PredictHighTraffic()
   3.1. if (HighTraffic)
       3.1.1. highTrafficSources.add[] ← NDPA(MyID)
       3.1.2. ReportBack(callerID, HighTraffic)
4. Wait(delay)
End while

```

The two previous algorithms (the server agent algorithm and the controller agent algorithm) collectively build a common opinion and negotiate it to predict the possible attack based on the traffic nature and the server status. If they predict an attack is about to happen or happening then the victimized server can call the C2C algorithm which starts the NDPA algorithm, and the edge controller can call C2C algorithm in line (3) which starts NDPA algorithm in line (2.1.1). The NDPA works by negotiating the maximum traffic allowed to pass reconfiguring switches and routers.

Algorithm 5. Network detection and prevention agent (NDPA)

```

1. StatusList[] ← NDPA(callerID)
2. Set initial=0.5%
3. Set static limitRatio= initial
4. Local:
5. If(PredictHighTraffic() == TRUE )
   5.1. For each Network Device (ND):
       5.1.1. ND → Reduce_BW(limitRatio)
       5.1.2. StatusList.add(HighTraffic, MyID)
   5.2. limitRatio= limitRatio+initial
   5.3. For each neighbouring controller (NCi) :
       5.3.1. NCi → C2C ( )
6. Elseif (limitRatio>initial )
   6.1. limitRatio= limitRatio - initial
   6.2. For each Network Device (ND):
       6.2.1. ND → Increase_BW(initial)
7. ReportBack(StatusList,callerID)

```

The NDPA algorithm calls the predict high traffic algorithm (Algorithm 3). A “TRUE” returned value means high traffic detected. Thus, NDPA will start reducing the throughput of the ports by the limit ratio of 25% initially in line 4.1.1 or any other ratio specified by the engineers. As long as the high traffic is “TRUE” the throughput reduction will be increased by 25% each round as shown in line 4.2. NDPA will contact its neighboring controllers to check their network devices for high traffic by calling C2C algorithm line 4.3.1.

If the predict high traffic algorithm return value was “FALSE”, which means traffic is not high or high traffic reduced; NDPA starts the network recovery to normal throughput levels as shown in lines 5-5.2.1 by increasing ports throughput gradually adding 25% back to the port each time. The NDPA algorithm will update the history list to new values.

It is worth mentioning that the NDPA algorithm, which is a network throughput reduction policy, applies only to a network connection that produces the highest traffic only in abnormal rates compared to its history. Other links and nodes in the network will not suffer from any reduction in throughput and will be served normally. This guarantees that only abnormal sub-networks or nodes causing undesired behavior will be affected by the reduction.

6. RESULTS

Algorithm simulation results showed the ability of the proposed algorithm to detect possible DDoS attacks in a tree topology network. As shown in Table 1, after observing the incremental nature of traffic volume causing the CPU to suffer high utilization, the server started negotiating traffic with the edge controller. At time 20 through 24 the utilization became very high, and the traffic throughput reached the maximum allowed amount. At that point the server started a positive detection process in collaboration with the edge controller. The detection process lasted for 4 cycles. In time 25 the controller started reducing the maximum amount of traffic by 25% to become 75% of the original throughput.

However, the problem is not solved because the server reported back a high utilization. The controller takes further steps to mitigate the high traffic by reducing another 25% of throughput and so on-going from 100% throughput to 75%, then 56.25%, 42.1875%, 31.640625%, 23.730469%, till 17.797852%, and finally 13.348389% of the original throughput which allows 11,676 kilobytes per second to be passed to the server.

Table 1. Very high normal traffic followed by attack traffic

Time	Utilization %			Throughput %	Total Traffic	Passed Traffic	attack detected
	CPU	Memory	Network				
1	80	65	100	100	11,900	11,900	FALSE
2	81.904	66.904	100	100	11,900	11,900	FALSE
3	83.808	68.808	100	100	11,900	11,900	FALSE
....
23	97.3630	83.82553	100	100	12,495	12,495	FALSE
24	99.3622	85.82473	100	100	12,460	12,460	FALSE
25	97.3686	83.83113	100	75	12,600	12,500	TRUE
26	99.3686	85.83113	100	56.25	12,600	12,500	TRUE
27	100	87.83113	100	42.1875	12,600	12,500	TRUE
28	100	89.83113	100	31.640625	13,300	12,500	TRUE
29	100	91.83113	100	23.730469	13,300	12,500	TRUE
30	100	93.83113	100	17.797852	13,300	12,500	TRUE
31	100	95.83113	100	13.348389	13,300	12,500	TRUE
32	100	97.83113	100	16.463013	11,676	11,676	FALSE
33	98.1318	95.96297	100	12.34726	12,600	12,500	TRUE
34	95	93.06483	100	15.228287	10,801	10,801	FALSE
35	93.2718	91.33667	100	11.421215	13,300	12,500	TRUE
36	90.5082	88.66984	100	14.086164	9,989	9,989	FALSE
37	92.1064	90.26808	100	17.372936	12,320	12,320	FALSE
38	94.0776	92.23928	100	13.029702	13,300	12,500	TRUE
39	96.0776	94.23928	100	16.069967	11,396	11,396	FALSE
40	97.9010	96.06264	100	12.052475	12,600	12,500	TRUE
41	94.9059	93.15951	100	14.864719	10,542	10,542	FALSE
42	96.5927	94.84623	100	11.14854	12,600	12,500	TRUE
43	93.6630	92.00391	100	13.749865	9,751	9,751	FALSE
44	92.1029	90.44375	100	16.958166	12,026	12,026	FALSE
45	94.0270	92.36792	100	12.718625	13,300	12,500	TRUE

Figure 3 shows the growing traffic flow toward the victim server, and the area between dashed lines A and B in Figure 3 shows approximately where the proposed algorithm started detecting a possible attack. Figure 4 shows the throughput of traffic in the local victim server network, where the dashed line A approximates the time when the NDPA algorithm decided to run a throughput reduction policy to reduce the possible malicious traffic. The area after dashed line B in both figures represents the drop in resource consumption due to throughput reduction enforced by the prevention algorithm (NDPA).

The zigzag behavior in Figure 4 indicates the areas where the algorithm assumes a possible recovery in server resources and tries to restore normal traffic gradually, however, when continuity in attack is detected, the algorithm enforces the throughput reduction again to insure providing service and server not to crash. The detection process lasted for 4 cycles. In time 25 the controller started reducing the maximum amount of traffic by 25% to become 75% of the original throughput. However, the problem is not solved because the server reported back a high utilization. The controller takes further steps to mitigate the high traffic by reducing another 25% of throughput and so on-going from 100% throughput to 75%, then 56.25%, 42.19%, 31.64%, 23.73%, to 17.79%, and finally 13.35% of the original throughput which allows 11,676 kilobytes per second to be passed to the server.

In the second experiment, Table 2, it is assumed that the victim server and the edge controller have no previous experience with attack, no traffic history available for server or edge controller. However, the server will start working using the pre-set metrics or default and thresholds. The edge controller will assume that the server will be able to detect any abnormal levels of utilization and report it back through the controller agent. The controller then starts with the NDPA algorithm to limit the traffic after making sure that the reported metrics from the server are not due to internal attack from the server side. As shown in Table 2, the server reported back to the controller with attack detected in time slice 7. The controller started the process of limiting the traffic throughput to acceptable levels. In time 20, it is noticeable from the traffic amount that the attack stopped by the attacker. The detection and prevention method stopped and most importantly it restored the network ability to transfer more information by gradually increase the upper limit allowed throughput.

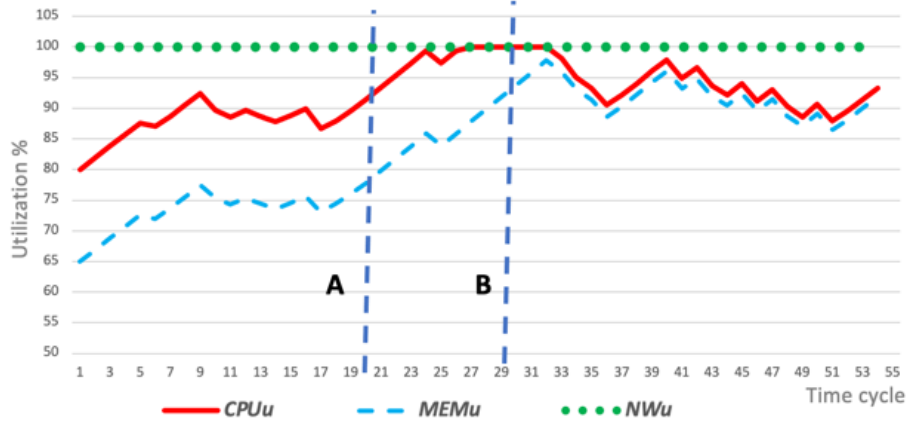


Figure 3. Resources utilization before and during continuous attack

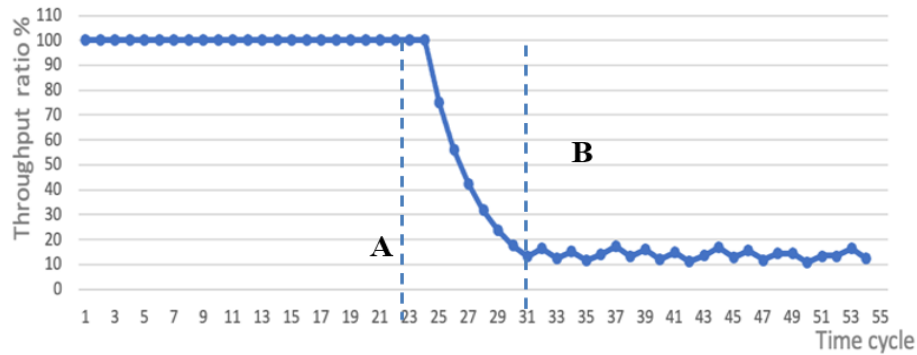


Figure 4. Throughput before and after attack detection

Table 2. Sudden attack with no previous history records of traffic

Time	Utilization %			Throughput Ratio	Total Traffic	Passed Traffic	Attack detected
	CPU	Memory	Network				
1	80	65	100	100	15,600	12,500	FALSE
2	82	67	100	100	12,600	12,500	FALSE
3	84	69	100	100	12,600	12,500	FALSE
4	86	71	100	100	12,600	12,500	FALSE
5	88	73	100	100	13,300	12,500	FALSE
6	90	75	100	100	14,000	12,500	FALSE
7	92	77	100	75	12,600	12,500	TRUE
8	94	79	100	56.25	12,600	12,500	TRUE
9	91.2	76.95	100	75	5,600	5,600	FALSE
10	90.30399	76.05399	100	56.25	12,600	12,500	TRUE
11	92.30399	78.05399	100	42.1875	12,600	12,500	TRUE
12	94.30399	80.05399	100	31.640625	12,600	12,500	TRUE
13	96.30399	82.05399	100	23.730469	13,300	12,500	TRUE
14	98.30399	84.05399	100	17.797852	14,000	12,500	TRUE
15	100	86.05399	100	13.348389	12,600	12,500	TRUE
16	100	88.05399	100	17.797852	11,676	11,676	FALSE
17	100	89.92215	100	13.348389	12,600	12,500	TRUE
18	100	91.92215	100	17.797852	11,676	11,676	FALSE
19	98.13184	90.05399	100	13.348389	12,600	12,500	TRUE
20	100	92.05399	100	17.797852	11,676	11,676	FALSE
21	93.22525	85.676544	100	23.730469	11,200	11,200	FALSE
22	90.26639	83.095116	100	31.640625	11,200	11,200	FALSE
23	87.45547	80.64276	100	42.1875	11,200	11,200	FALSE
24	89.24747	82.43476	100	56.25	11,200	11,200	FALSE
25	86.487495	80.01542	100	75	10,500	10,500	FALSE
26	84.807495	78.33542	100	100	10,500	10,500	FALSE
27	86.487495	80.01542	100	100	10,500	10,500	FALSE
28	88.167496	81.69542	100	100	10,850	10,850	FALSE

Figures 5 and 6 show the analysis of the victim server and NDPA algorithm response to a temporary attack during normal high demand on server resources. This detection process in this scenario started from the controller agent side. The detection and prevention algorithm detected that before that and started reducing the network traffic to a level that the server can continue providing service as shown in Figure 5 in area A. in area B, the system was stable until the attack stopped around time 23. NDPA restored network throughput since the server is not suffering from over-utilization to normal levels as shown in Figure 5 area C. As shown in Figure 6, in time 13 the server was suffering from a potential DDoS attack, and server resources are over-utilized. In time 23 the NDPA algorithm was able to lower the throughput lowering the CPU utilization to acceptable ratio. Simulation results demonstrated that the proposed algorithms were able to detect and prevent the server from crashing and continued providing services despite the DDoS attack because the throughput reduction applied to all network routes that generate high traffic only.

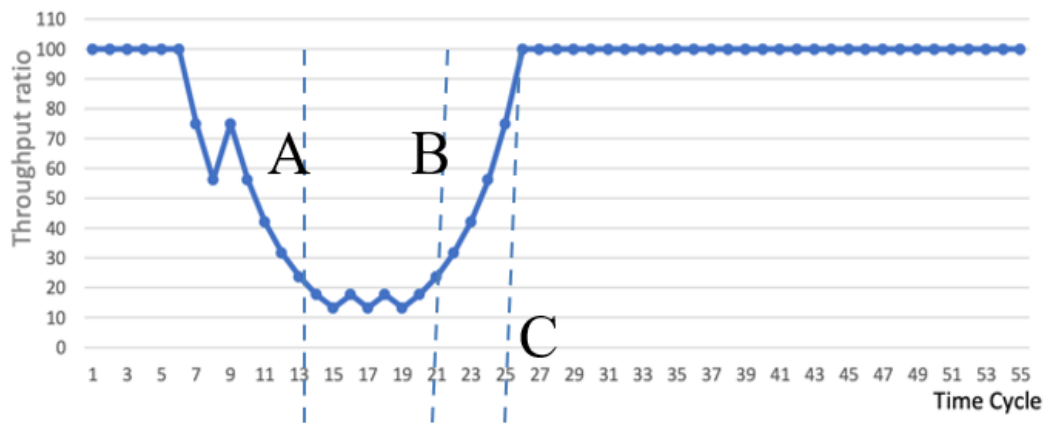


Figure 5. Throughput recovery to normal after temporary DDoS attack

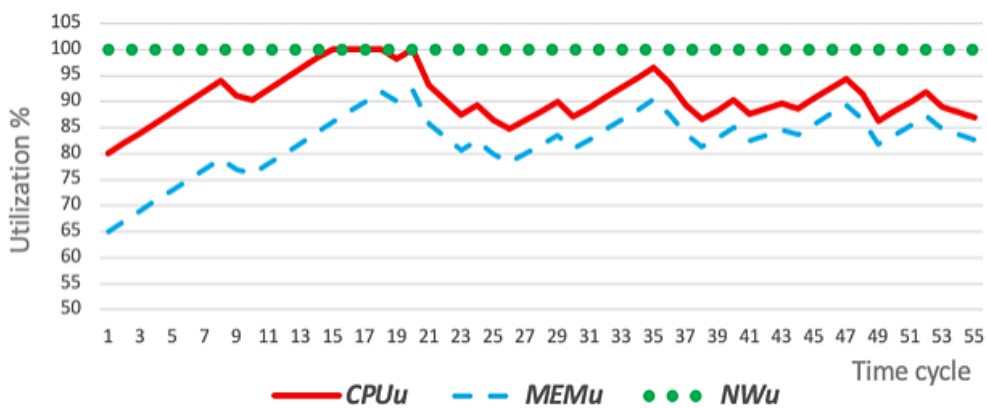


Figure 6. Resources consumption for temporary attack and during a normal high traffic

In the 3rd experiment; we assume an attack traffic initiated from the neighboring network, then the attacker stops the attack. What is assumed that the NDPA should be able to detect the attack, limit the throughput, then when the attack is over, NDPA must restore the network throughput to its original values. This behavior must protect the sever form the bad traffic. As shown in Table 3, first router (Router0) managed by (Controller 2) is passing safe traffic. However, (Router1) is passing very large traffic and causing a DDoS attack symptom. However, traffic is coming from different sources and other networks.

At time 4, (Router1) experienced limitations imposed by the managing controller based on the commination and negotiation with the edge controller. From the table, it is observable that (Router0) did not suffer any reduction in the throughput due to NDPA control of traffic on the network. The edge-router throughput was reduced to be able to prevent any local high traffic on the incoming uplinks, though, the good

about that is the local traffic will not be affected by the reduction since the new throughput will be more sufficient to pass the safe traffic through towards the server. Users might experience some delay, but the server will not crash due to high traffic.

Table 3. Two-level architecture, experiment 3 results

Time	Edge-Controller Traffic	Edge-Router Traffic	isHigh Traffic	Edge-Router Throughput Ratio	Managed by Controller2, Router0 and Router1 from network2 connected to the Edge-Router from the previous network						
					Controller2 Traffic	Router0 Traffic	isHigh Traffic	Router0 Throughput Ratio	Router1 Traffic	isHigh Traffic	Router1 Throughput Ratio
1	14,900	14,900	TRUE	100	16,800	1,400	FALSE	100	15,400	TRUE	100
2	13,700	13,700	TRUE	100	21,700	700	FALSE	100	21,000	TRUE	100
3	13,700	13,700	TRUE	100	18,200	700	FALSE	100	17,500	TRUE	100
4	14,900	14,900	TRUE	75	22,400	1,400	FALSE	100	21,000	TRUE	75
5	12,975	12,975	TRUE	56.25	23,100	2,100	FALSE	100	21,000	TRUE	56.25
6	10,631	10,631	FALSE	75	23,100	2,100	FALSE	100	21,000	TRUE	42.1875
7	9,735	9,735	FALSE	100	21,210	210	FALSE	100	21,000	TRUE	31.640625
8	15,860	15,860	TRUE	100	21,560	1,960	FALSE	100	19,600	TRUE	31.640625
9	14,900	14,900	TRUE	100	21,000	1,400	FALSE	100	19,600	TRUE	31.640625
10	14,900	14,900	TRUE	100	21,000	1,400	FALSE	100	19,600	TRUE	31.640625
11	13,700	13,700	TRUE	75	13,300	700	FALSE	100	12,600	TRUE	23.730469
12	10,575	10,575	FALSE	100	13,300	700	FALSE	100	12,600	TRUE	17.797852
13	13,700	13,700	TRUE	100	14,000	700	FALSE	100	13,300	TRUE	17.797852
14	14,900	14,900	TRUE	100	15,400	1,400	FALSE	100	14,000	TRUE	17.797852
15	22,100	22,100	TRUE	75	18,200	5,600	FALSE	100	12,600	TRUE	13.348389
16	10,575	10,575	FALSE	100	12,376	700	FALSE	100	11,676	FALSE	17.797852
17	14,660	14,660	TRUE	75	13,860	1,260	FALSE	100	12,600	TRUE	13.348389
18	11,535	11,535	FALSE	100	12,936	1,260	FALSE	100	11,676	FALSE	17.797852
19	13,700	13,700	TRUE	75	13,300	700	FALSE	100	12,600	TRUE	13.348389
20	10,575	10,575	FALSE	100	12,376	700	FALSE	100	11,676	FALSE	17.797852
21	18,400	18,400	TRUE	100	15,400	4,200	FALSE	100	11,200	FALSE	17.797852
22	12,400	12,400	FALSE	100	11,900	700	FALSE	100	11,200	FALSE	23.730469
23	12,400	12,400	FALSE	100	11,900	700	FALSE	100	11,200	FALSE	23.730469
24	12,400	12,400	FALSE	100	11,900	700	FALSE	100	11,200	FALSE	23.730469
25	12,300	12,300	FALSE	100	11,550	1,050	FALSE	100	10,500	FALSE	23.730469
26	11,700	11,700	FALSE	100	11,200	700	FALSE	100	10,500	FALSE	23.730469
27	11,700	11,700	FALSE	100	11,200	700	FALSE	100	10,500	FALSE	23.730469
28	11,510	11,510	FALSE	100	11,235	385	FALSE	100	10,850	FALSE	23.730469
29	10,970	10,970	FALSE	100	10,920	70	FALSE	100	10,850	FALSE	31.640625
30	12,400	12,400	FALSE	100	11,900	700	FALSE	100	11,200	FALSE	31.640625
31	20,300	20,300	TRUE	100	16,800	4,900	FALSE	100	11,900	FALSE	31.640625
32	20,300	20,300	TRUE	100	16,800	4,900	FALSE	100	11,900	FALSE	31.640625
33	13,100	13,100	TRUE	100	12,600	700	FALSE	100	11,900	FALSE	31.640625
34	13,100	13,100	TRUE	75	12,600	700	FALSE	100	11,900	FALSE	42.1875
35	6,473	6,473	FALSE	100	12,600	700	FALSE	100	11,900	FALSE	56.25
36	4,450	4,450	FALSE	100	4,200	350	FALSE	100	3,850	FALSE	75
37	11,450	11,450	FALSE	100	11,200	350	FALSE	100	10,850	FALSE	75
38	13,120	13,120	TRUE	75	12,320	1,120	FALSE	100	11,200	FALSE	100
39	13,940	13,940	TRUE	56.25	13,090	1,190	FALSE	100	11,900	FALSE	100
40	9,071	9,071	FALSE	75	13,090	1,190	FALSE	100	11,900	FALSE	100
41	7,380	7,380	FALSE	100	6,930	630	FALSE	100	6,300	FALSE	100
42	7,380	7,380	FALSE	100	6,930	630	FALSE	100	6,300	FALSE	100
43	6,560	6,560	FALSE	100	6,160	560	FALSE	100	5,600	FALSE	100
44	13,940	13,940	TRUE	100	13,090	1,190	FALSE	100	11,900	FALSE	100
45	13,940	13,940	TRUE	75	13,090	1,190	FALSE	100	11,900	FALSE	100
46	11,415	11,415	FALSE	100	13,090	1,190	FALSE	100	11,900	FALSE	100
47	13,100	13,100	TRUE	75	12,600	700	FALSE	100	11,900	FALSE	100
48	4,450	4,450	FALSE	100	4,200	350	FALSE	100	3,850	FALSE	100
49	11,450	11,450	FALSE	100	11,200	350	FALSE	100	10,850	FALSE	100
50	13,600	13,600	TRUE	75	12,600	1,400	FALSE	100	11,200	FALSE	100
51	10,575	10,575	FALSE	100	12,600	700	FALSE	100	11,900	FALSE	100

As shown in Figure 7, controller 2 monitoring and managing network 2 concluded that the routers in the network are passing very high traffic through. Meanwhile server is suffering the high traffic and reported it back to the edge-controller agent. In its turn, the edge-controller initiated a controller-to-controller communication (C2C) to track down traffic. it is noticeable Figure 7, in cycle 11, that controller 2 and just after implementing NDPA started observing lower traffic than before. This new traffic is a direct result of controllers implementing throughput reduction policy in routers they manage.

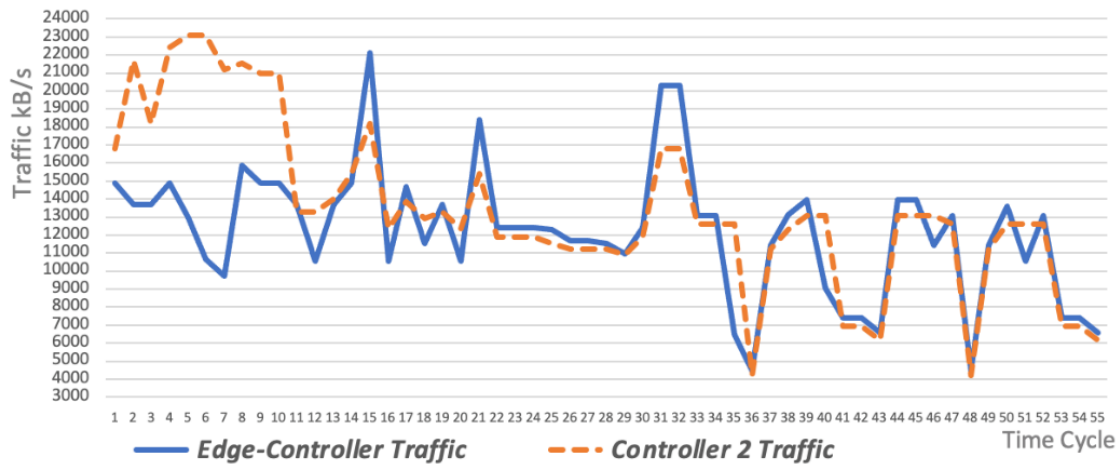


Figure 7. Two controllers traffic detection

7. CONCLUSION

Software defined networks can be used to provide a variety of services, such as security, with a sophisticated design and separation of duties between components to prevent DDoS attacks. Basic assumptions made for this solution to work as intended are: The ability for controllers to communicate efficiently, controllers have some computational power, routers and switches throughput can be reconfigured, and controllers can get statistics about routers' traffic. As shown by experiments designed, the proposed solution and its algorithms managed to detect and the proposed solution was able to prevent a DDoS attack from overwhelming the server, orchestrate its work with controllers, detect if the attack was over, and restore throughput to its original values to elevate the QoS provided to customers and networks.




REFERENCES

- [1] Kaspersky Lab, "Distributed denial of service: How DDoS attacks work," *Kaspersky*, <https://me-en.kaspersky.com/resource-center/preemptive-safety/how-does-ddos-attack-work> (accessed May 22, 2022).
- [2] T. Mahjabin, Y. Xiao, G. Sun, and W. Jiang, "A survey of distributed denial-of-service attack, prevention, and mitigation techniques," *International Journal of Distributed Sensor Networks*, vol. 13, no. 12, Art. no. 155014771774146, Dec. 2017, doi: 10.1177/1550147717741463.
- [3] W. Queiroz, M. A. M. Capretz, and M. Dantas, "An approach for SDN traffic monitoring based on big data techniques," *Journal of Network and Computer Applications*, vol. 131, pp. 28–39, Apr. 2019, doi: 10.1016/j.jnca.2019.01.016.
- [4] M. D. Hatagundi and H. V. Kumaraswamy, "A comprehensive survey on different attacks on SDN and approaches to mitigate," in *Proceedings of the 3rd International Conference on Computing Methodologies and Communication, ICCMC 2019*, IEEE, Mar. 2019, pp. 624–627, doi: 10.1109/ICCMC.2019.8819717.
- [5] S. Dong, K. Abbas, and R. Jain, "A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments," *IEEE Access*, vol. 7, pp. 80813–80828, 2019, doi: 10.1109/ACCESS.2019.2922196.
- [6] D. Chouhan and A. Pal, "Detection and mitigation of mitigate denial of service (Dos) attacks using trust-based mechanism," *International Journal of Scientific Research & Engineering Trends (IJSRET)*, vol. 7, no. 4, 2021.
- [7] L. F. Eliyan and R. di Pietro, "DoS and DDoS attacks in software defined networks: A survey of existing solutions and research challenges," *Future Generation Computer Systems*, vol. 122, pp. 149–171, Sep. 2021, doi: 10.1016/j.future.2021.03.011.
- [8] S. B. Chinmay Dharmadhikari, Salil Kulkarni, Swarali Temkar, "A study of DDoS attacks in software defined networks," *A Study of DDoS Attacks in Software Defined Networks*, vol. 6, no. 12, 2019.
- [9] B. Kumar Joshi, N. Joshi, and M. Chandra Joshi, "Early detection of distributed denial of service attack in era of software-defined network," in *2018 11th International Conference on Contemporary Computing, IC3 2018*, IEEE, Aug. 2018, doi: 10.1109/IC3.2018.8530546.
- [10] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDOS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 602–622, 2016, doi: 10.1109/COMST.2015.2487361.
- [11] N. N. Tuan, P. H. Hung, N. D. Nghia, N. Van Tho, T. Van Phan, and N. H. Thanh, "A DDoS attack mitigation scheme in ISP networks using machine learning based on SDN," *Electronics (Switzerland)*, vol. 9, no. 3, p. 413, Feb. 2020, doi: 10.3390/electronics9030413.
- [12] UNB, "DDoS evaluation dataset (CIC-DDoS2019)," *Canadian Institute for Cybersecurity*, <https://www.unb.ca/cic/datasets/ddos-2019.html> (accessed Jul. 13, 2023).
- [13] H. A. Alamri and V. Thayananthan, "Bandwidth control mechanism and extreme gradient boosting algorithm for protecting software-defined networks against DDoS attacks," *IEEE Access*, vol. 8, pp. 194269–194288, 2020, doi: 10.1109/ACCESS.2020.3033942.
- [14] Y. Liu, B. Zhao, P. Zhao, P. Fan, and H. Liu, "A survey: Typical security issues of software-defined networking," *China Communications*, vol. 16, no. 7, pp. 13–31, Jul. 2020, doi: 10.23919/jcc.2019.07.002.
- [15] Y. Chen, J. Pei, and D. Li, "DETPro: A high-efficiency and low-latency system against DDoS attacks in SDN based on decision tree," in *IEEE International Conference on Communications*, IEEE, May 2019, doi: 10.1109/ICC.2019.8761580.




- [16] R. Sanjeetha, A. Prasanna, D. Pradeep Kumar, and A. Kanavalli, "Mitigation of controller induced DDoS attack on primary server in high traffic scenarios of software defined networks," in *International Symposium on Advanced Networks and Telecommunication Systems, ANTS*, IEEE, Dec. 2018, doi: 10.1109/ANTS.2018.8710066.
- [17] T. M. Nam *et al.*, "Self-organizing map-based approaches in DDoS flooding detection using SDN," in *International Conference on Information Networking*, IEEE, Jan. 2018, pp. 249–254, doi: 10.1109/ICOIN.2018.8343119.
- [18] T. Ubale and A. K. Jain, "SRL: An TCP SYNFLOOD DDoS mitigation approach in software-defined networks," in *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018*, IEEE, Mar. 2018, pp. 956–962, doi: 10.1109/ICECA.2018.8474561.
- [19] B. Aryal, R. Abbas, and I. B. Collings, "SDN enabled DDoS attack detection and mitigation for 5G networks," *Journal of Communications*, vol. 16, no. 7, pp. 267–275, 2021, doi: 10.12720/jcm.16.7.267-275.
- [20] V. Sharma, "Multi-agent based intrusion prevention and mitigation architecture for software defined networks," in *International Conference on Information and Communication Technology Convergence: ICT Convergence Technologies Leading the Fourth Industrial Revolution, ICTC 2017*, IEEE, Oct. 2017, pp. 686–692, doi: 10.1109/ICTC.2017.8191067.
- [21] D. Hyun, J. Kim, D. Hong, and J. Jeong, "SDN-based network security functions for effective DDoS attack mitigation," in *International Conference on Information and Communication Technology Convergence: ICT Convergence Technologies Leading the Fourth Industrial Revolution, ICTC 2017*, IEEE, Oct. 2017, pp. 834–839, doi: 10.1109/ICTC.2017.8190794.
- [22] A. Kalliola, K. Lee, H. Lee, and T. Aura, "Flooding DDoS mitigation and traffic management with software defined networking," in *2015 IEEE 4th International Conference on Cloud Networking, CloudNet 2015*, IEEE, Oct. 2015, pp. 248–254, doi: 10.1109/CloudNet.2015.7335317.
- [23] P. Manso, J. Moura, and C. Serrão, "SDN-based intrusion detection system for early detection and mitigation of DDoS attacks," *Information (Switzerland)*, vol. 10, no. 3, p. 106, Mar. 2019, doi: 10.3390/info10030106.
- [24] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *2015 International Conference on Computing, Networking and Communications, ICNC 2015*, IEEE, Feb. 2015, pp. 77–81, doi: 10.1109/ICCNC.2015.7069319.
- [25] R. Sahay, G. Blanc, Z. Zhang, and H. Debar, "Towards autonomic DDoS mitigation using software defined networking," in *Proceedings 2015 Workshop on Security of Emerging Networking Technologies*, Internet Society, 2015, doi: 10.14722/sent.2015.23004.

BIOGRAPHIES OF AUTHORS






Rima Abdelhadi    received a B.Sc. degree in science, majoring in computer science from Mutah University, Alkarak, Jordan, in 2002, and a Master of Engineering, specializing in computer and networks engineering in 2022, from Al-Hussein bin Talal University, Ma'an, Jordan. Alqudah focuses on her work in computer security and protection, Protecting infrastructure and resources. She can be contacted at email: rima.qudah@gmail.com



Moath H. Alsafasfeh    associate professor electrical and computer engineering 2017 from Western Michigan University (WMU), USA, an M.Eng. in Communication and Computer Engineering from Universiti Kebangsaan Malaysia (UKM) in June 2011, and B.Eng. Computer Engineering from Mutah University, Jordan. Dr. Alsafasfeh is an associate professor in the College of Engineering, Department of Computer Engineering, and also the Director of the Academic Development and Quality Assurance Center at Al-Hussein bin Talal University. Research interests are parallel processing, multiprocessor, and multicore systems; computer vision, image processing, and pattern recognition; AI and machine learning; non-destructive testing and evaluation; renewable energy systems (solar panels and gardens); drones and quadcopters. He can be contacted at email: moath.alsafasfeh@ahu.edu.jo



Bilal I. Alqudah    assistant professor of computer security and privacy protection at the College of Engineering at Al-Hussein Bin Talal University, Ma'an. He holds a Ph.D. in computer security and privacy protection, and a master's degree in computer science from the Bobby B. Lyle College of Engineering, Southern Methodist University, Dallas, Texas- USA, bachelor's degree in computer science from Mutah University in Karak, Jordan. Dr. Alqudah has held many local and international training seminars and conferences in their field of specialization. Dr. Alqudah focuses on computer security and privacy research, electronic medical record, and access controlling in addition to other areas of interest. He can be contacted at email: alqudah@ahu.edu.jo.