CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter presents the background to common DDoS attacks and SDN components. The operation of DDoS attacks and the operation of OpenFlow in SDN are presented. This chapter also presents the SDN controllers and the benefits of SDN. The DDoS attacks in SDN are then discussed. The existing SDN based solutions for detecting DDoS attacks are categorized based on the detection technique used. Each category is discussed in the subsequent subsection according to the performance of the detection technique and parameters used for detection. A comparison between all SDN based DDoS detection solutions discussed in the chapter is provided. The SDN based DDoS attacks mitigation methods are discussed. A chapter conclusion is finally provided.

2.2 Distributed Denial of Service (DDoS) Attacks

The DDoS attack objective is to make the services inaccessible by legitimate users like in flooding attacks scenarios, where the victim is overwhelmed with the massive amount of traffic sent to it. The DDoS attack idea revolved around the fact that the victim is targeted with a large number of sources distributed across multiple locations. DDoS attacks are typically launched via botnets as a large collection of zombies (also known as compromised computers). The major focus of the research

community has been the prevention of DDoS attacks (Hoque et al., 2015; Wang et al., 2015).

2.2.1 Operation of DDoS Attacks

When it was difficult for attackers to overload the target's resource from a single computer, many recent DoS attacks were launched via a large number of distributed attacking hosts in the Internet. These attacks are called Distributed Denial of Service attacks. DDoS attack is an internet attack that is implemented as a coordinated attack and is launched indirectly through many compromised computers at a large scale. Source attacker uses client-server technology to multiply effectiveness of the Denial of Service significantly by exploiting the resources of multiple ignorant assistant computers. A DDoS attack is carried out by a group of machines (agents) sends packets to a victim host on receiving commands from a machine (master) controlled by the attacker. In more details, the host of masters and slaves is compromised machines that have identified during the scanning process and are tainted by malicious code. The attacker handles master agents and they coordinate and orders slave agents. More specifically, the attacker sends an attack command to master agents and activates all attack processes on those machines, which are in hibernation, waiting for the appropriate command to wake up and start attacking. Then, master agents, through those processes, send attack commands to slave agents, ordering them to mount a DDoS attack against the target. In that way, the agent machines (slaves) begin to send a large volume of packets to the target, flooding its system with useless load and exhausting its resources. Figure 2.1 shows this kind of DDoS attack (Dillon & Berkelaar, 2014).

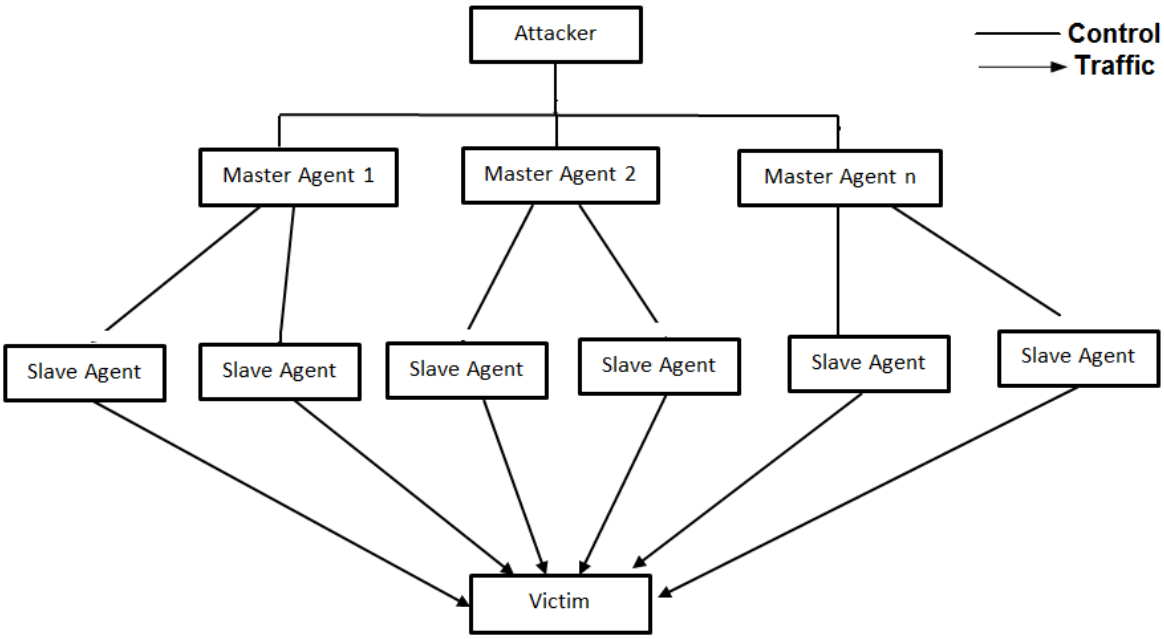

Figure 2.1: DDoS Attack structure

2.2.2 Types of DDoS Attacks

Highest DDoS attacks incidents that have been reported in the first quarter of 2019 (Securelist, 2019) showed that there was a sharp increase in the number and proportion of flooding SYN DDoS attacks as well as the percentage of UDP attacks also grew significantly. These attacks will be looked at next paragraphs.

In SYN attack, the attack exploits the three-way handshake between the sender and receiver by sending large amount of SYN requests with spoofed source address. If those half-open connection binds resources on the server or the server software is licensed per-connection, all these resources might be taken up.

Healthy TCP handshake is a synchronization message sent by a client to the server and that means a request for opening or establishing a connection was sent. The server responds with acknowledgement to the client by sending SYN-ACK message and that means server accepts the establishing of the connection. The client

completes this establishment by responding with an ACK message. The connection between the client and the server is then opened and the data can be exchanged between them. But in spoofed SYN handshake the server waits for the client's ACK message after sending SYN-ACK message to it. The client or (victim) will never receive the message because of the spoofing source IPs in SYN messages and with this process the connection will not be completed but created what is called half-open connection. The half-open connection will be allocated in limited space in the victim's process table and with too many half-open connections this space will soon filled up. As result, the victim will not be able to accept any new connections and therefore this will lead to stop providing services (Meena & Jadon, 2014).

Unlike flooding DDoS attacks that exhaust the bandwidth, CPU power, or memory of the victim host by flooding an overwhelming number of packets from thousands of compromised computers (zombies) to deny legitimate flows, low rate DDoS attacks exhaust the victim's resources by launching low number of attack packets continuously to be similar to legitimate traffic. A low-rate DDoS attacker exploits the vulnerability of TCP's congestion control mechanism by continuously launching SYN attack packets at a constant low rate (constant attack) where these attacks reduce the average number of attack packets to avoid being detected by existing detection solutions at a time of traffic generation (Zhou et al., 2017).

In User Datagram Protocol (UDP) attack a large amount of UDP packets may be sent simply by attackers towards to a victim. Since an intermediate network can deliver higher traffic volume than the victim network, the flooding traffic can exhaust the victim's connection resources (Meena & Jadon, 2014).

Juyal and Prabhakar (2012) identified Internet Control Message Protocol (ICMP) Attack as Smurf attack and send forged ICMP echo request packets to IP broadcast addresses. A large amount of ICMP echo reply packets are sent via an intermediary site to a victim. This large flow of echo reply causes network congestion and disruption of resources from victim. ICMP datagram use the ping command to construct oversized ICMP datagram to launch ping attack.

Table 2.1: Classification of DDoS attacks

Classification	Attack	Description
Reflection-based flooding attacks	Smurf Attack	In Smurf Attack a large number of Internet Control Message Protocol (ICMP) packets with the intended victim's forged source IP are broadcasted to a computer network using an IP Broadcast address. This generates huge amount of illegitimate traffic on the network causing network congestion.
	Fraggle Attack	Fraggle attack is similar to Smurf attack, but it uses illegitimate UDP traffic instead of ICMP traffic to achieve the same goal.
Protocol exploitation flooding attacks	SYN Flooding attack	In TCP SYN flood attack, an attacker sends the packet with the SYN bit setoff TCP three-way handshake. The victim responds with the packet back to the source address with SYN-ACK bit set. The attacker never responds to the reply packet, either on purpose or because the source address of the packet is forged. Therefore, the victim's TCP receive

	UDP Fragmentation attack	queues would be filled up, denying new TCP connections to legitimate Clients. In a UDP Fragmentation attack, attackers send large UDP packets (1500+ bytes) to consume more bandwidth with fewer packets. The victim's resources are consumed in reassembling these forged packets which no ability to be re-assembled

2.3 Software Defined Networking (SDN)

Software Defined Networking (SDN) is a new network structure designed to facilitate the way of managing, measuring, debugging and controlling the network dynamically, and to make it suitable for the modern applications. SDN reliability and flexibility are shown from the separation between the control plane and the data plane. Data plane or the infrastructure layer is composed of switches. The major work of these switches is forwarding the incoming packets according to the flow tables. Forwarding decisions can be decided and configured by the control plane through the southbound protocol. The First standard of the southbound protocol is the OpenFlow protocol (Myint et al., 2019). Control plane or control layer, logically contains the intelligence of the network which is known as a controller, while data plane contains the packets forwarding devices such as switches and routers (Cui et al., 2016). OpenFlow network (OF) is the core of the SDN and the OpenFlow devices (OF controllers and OF switches) are located in it. OpenFlow can do more than just sending routing information and receiving packet information. It can receive variety of information from the switch like port status, flow status and look

up table's status as examples. Therefore, the adopting of a various network application is feasible in OpenFlow (Fundation, O. N, 2012).

Considering architecture in Figure 2.2, SDN can be separated into three layers the top, middle and the bottom layers. These layers are known as Application layer, Control layer and Infrastructure layer. Generally, control plane is the logic that controls the forwarding traffic behavior in the network and the data plane is responsible for forwarding that traffic to the selected destination according to the control plane's logic. Moreover, control plane provides performance and fault management via SDN standard protocols. It typically handles configuration management of the SDN compliant devices and understands the network topology. On the other hand, the routing protocol as one of the control plane functions, it might computes the shortest path over a topology but ultimately the result of such computations must be installed in switches that actually do the forwarding. The forwarding table entries themselves and specifically the actions associated with forwarding traffic; according to the control plane logic is what constitutes the data plane. Switches can either be reliant on the controller to make forwarding decisions or make the decisions on their own (Doria et al., 2010). One can think of it in some ways as, the control plane is the brain of this operation and the data plane is the muscle.

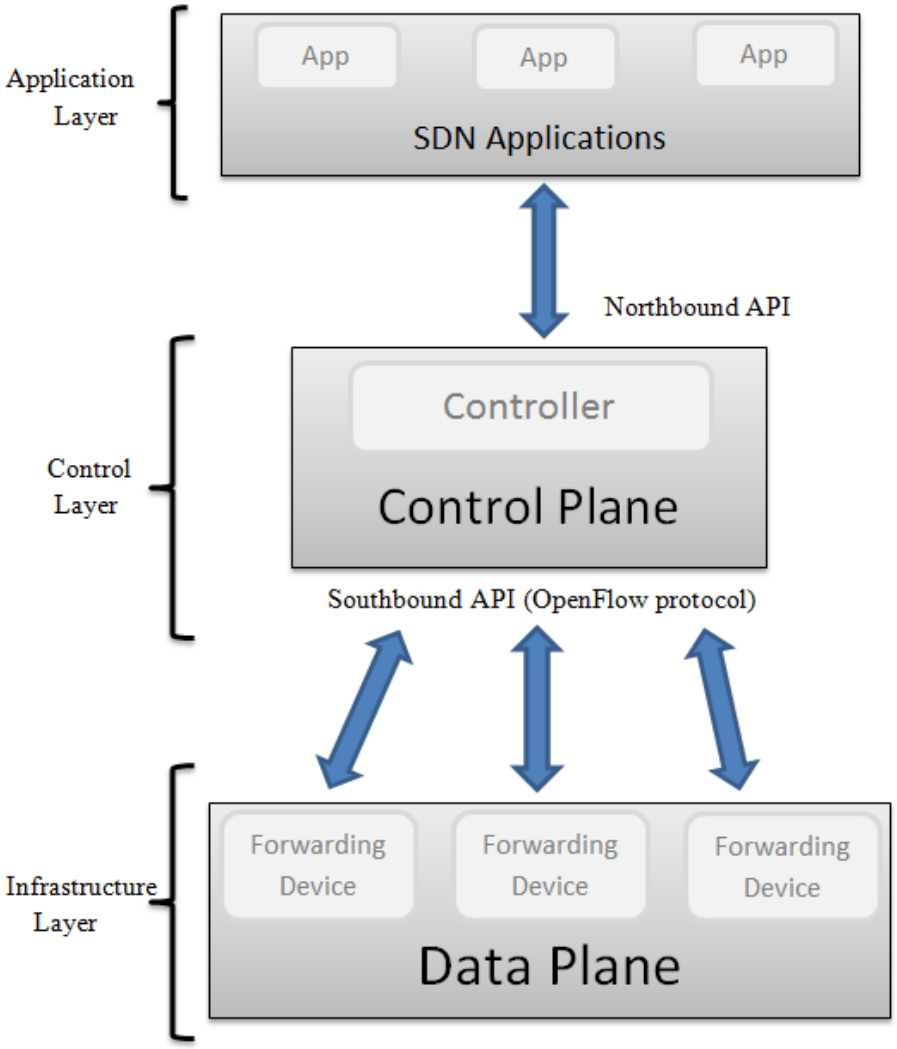

Figure 2.2: SDN architecture

Loaded with these details, the controller (control layer) can process connection requests based on desired requirements such as QoS levels from the application layer via the northbound API and it can also perform link management between devices in the infrastructure layer using the southbound API the OpenFlow protocol.

OpenFlow controllers are allowing control and watch the SDN network globally from the packet flow perspective. Meanwhile, OF switches are meant to keep flow tables with statistics about all active flows. Among other devices, the

controller will observe any variation from the normal behaviour in network traffic flows (Ahmed & Kim, 2016).

Ahmed and Kim (2016) found that "all the features information required is easily accessible by means of SDN controllers and then processed by an intelligent algorithm to decide about the maliciousness of traffic flows. Therefore, those attacks which are being launched from or converge at the SDN network domain can easily be detected and mitigated by exploiting the features offered by the SDN" (p. 5).

2.4 Openflow Protocol

OpenFlow protocol is the standard protocol used for communicating between control layer and infrastructure layer as defined by Open Networking Foundation (ONF) (Fundation, O. N, 2012). OpenFlow gives the direct access to and manipulation of the forwarding devices such as switches and routers over the network.

2.4.1 Openflow Table Entry

OpenFlow table is a data structure that resides in the high-speed data plane of an OpenFlow switch. Its contents determine the forwarding behavior and packet handling behavior of that OpenFlow switch. An OpenFlow table has one or more flow entries and each flow entry has a set of components. As illustrated in Figure 2.3, the flow entry in the OpenFlow switch's flow table includes Match Fields, Priority, Counter, Instructions, Timeout, Cookie, Flags (Version, O. S. S. 1.5. 1, 2014). Match fields that are used to match against packet headers in each individual flow. Colored flow entries in Figure 2.3 are the basic flow table entry components Specification, O. S. (2009). Once the incoming packet is matched with any of the match fields the packet and byte counters are automatically updated. The cumulated

numbers that stored into these two counters could be used as network statistics and the controller could enquiry the switch to forward these statistics to it. The packets counter is used in our research as an important statistic element. There are some instructions must be applied on these incoming packets once the matching process is done which are known as actions. These actions are installed on the switch by the SDN controller. These actions will be updated or add to it based on the status of the packet that sent by the OpenFlow message from the switch to the controller (Dillon & Berkelaar, 2014).

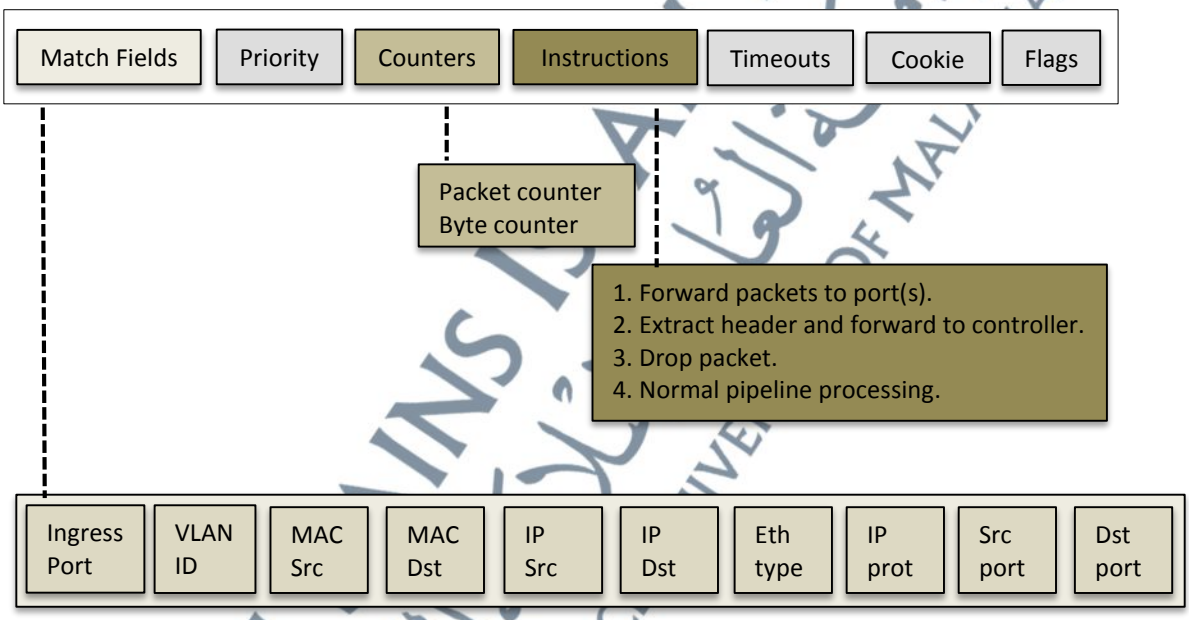

Figure 2.3: Flow table entry (Adopted from Version, O. S. S. 1.5. 1, 2014)

2.4.2 OpenFlow Message Types

The three message types that supported by the OpenFlow switch protocol are controller-to-switch, asynchronous, and symmetric messages. The message that initiated by the controller and used to inspect the state of the switch or to directly managing it named as controller-to-switch message. The asynchronous message is a message initiated by the switch and used to denote the controller with details about

the switch state changes, and to update it about the network events. A symmetric message is a message initiated either by the controller or by the switch same as the scenario of the Hello and echo messages.

2.4.3 OpenFlow Operation

When a host attempts to communicate with another host over an SDN as appeared in Figure 2.4, the first packets from the client involved with the new flow are used to determine whether or not a forwarding decision can be made locally by the switch or the switch needs to ask the controller what to do as in step 1. If the switch determines that it must ask the controller as in step 2, it will do so via a secure channel using the control protocol. The controller decides based on policies if the flow should be granted. If allowed, details about the flow could be entered into the controller's connection table. In the step 3 the controller instructs the switch(s) with actions that should be taken and creates a flow entry accordingly. The controller sends this flow entry to the switch to let the switch knows how these packets will traverse. Once the packets that belong to the same flow come to the switch again the switch can make the decision locally without asking the controller what is the best path along the data plane that the flow should be directed through as in the steps 1, 4 and 5. The switches may also tell the controller when a flow is no longer active. This removes it from the table.

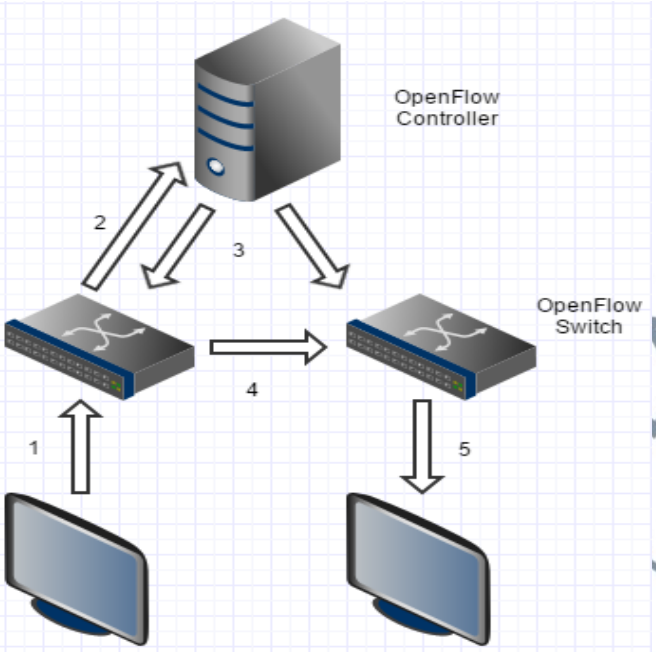

Figure 2.4: The flow processing procedure in OpenFlow

2.5 SDN Controllers

Typically in SDN, the intelligence of the network is logically centralized in controllers which are (software-based). The control logic in the controllers is designed and operated as a centralized application, rather than a distributed system on a global network view. There are various types of controllers in the market open-source and commercial controllers. Normally the controllers that used in the academic research are open-source controllers due to ease of re-programming and the availability of use without costs. There are a number of open-source controllers that are currently in use. The most used controllers are POX, Beacon, Ryu and Floodlight. These controllers are open-source controllers but are written in different programming languages such as Python and Java. Table 2 provides a brief overview of the controller characteristics. Among others, the most popular controller in the research and academia is POX controller which is a Python-based controller (Prete et

al., 2014). The proposed scheme in this research is implemented over POX controller.

Table 2.2: Controller characteristics

Controller	Programming language	Developer	Overview
POX	Python	Nicira	General, open-source Python-based controller, lightweight OpenFlow controller, suitable platform for SDN research, academic work, education, and experimentation.
Beacon	Java	Stanford	A cross platform, open-source Java-based controller, supports event-based and threaded operations.
Ryu	Python	Nippon Telegraph and Telephone Corporation (NTT)	Ryu is commonly referred to as component-based, open source software defined by a networking framework. It is implemented entirely in Python, and supported by NTT's labs.
Floodlight	Java	BigSwitch	Based on Beacon implementation, Java-based controller, support (OpenFlow v1.3).

2.6 SDN benefits

Decoupling of control from the data plane brings a lot of benefits in SDN. SDN is a strategy in which a controller software application manages network activity as opposed to the hardware supporting the platform. The separation of the control plane from the data plane enables SDN to provide a high simplification of the network management. By decoupling the data plane and control plane and thus enables to

establish easily large scale attack and defense experiments. The logical centralization of control helps the controller in SDN to have a global view of the network to build consistent security policy and to monitor or analyse traffic patterns for potential security threats (Nishtha et al., 2014).

Table 2.3: SDN features in defending DDoS attacks

SDN features	Help for defeating DDoS attacks
Global view	Establish consistent/regular security police
Flexibility and programmability	Change and support new set of instructions, supports also unforseen security measures
Dynamic network updates	Respond rapidly and directly
Separation of control plane from forwarding planes	Establish large scale attack and defence

The programmability of the SDN supports a process of harvesting intelligence from existing Intrusion Detection system (IDs) and Intrusion Prevention system (IPs) (Scott-Hayward et al., 2013). As presents in Table 3, the centralized management maintains a global view of the network; the flexibility in the SDN controller helps organizations rapidly to deploy new application services, and infrastructure to meet changing business goals and objectives.

2.7 DDoS Attacks in SDN

One of the clear indications of the limitations of existing technologies is the current DDoS trends. In the first quarter of 2019, SYN attack made up the biggest share of junk traffic where its share was 84% and the UDP attack holding on to the second spot by a share of 8.90% (Securelist, 2019).

Amid raising the number of DDoS attacks and the attackers' ability to develop attack types to penetrate traditional protection methods, SDN has emerged as an alternative environment to minimize the damage of this attack. Recently, software defined networking or SDN has appeared as a new paradigm of networking where it has attracted the attention of the research community widely. The detection of DDoS attacks becomes much easier if we are able to take advantage of the SDN distinct characteristics such as the centralization of control over the infrastructure, decoupling of the control plane from the data plane and the flow-based traffic concept (Kim et al., 2015). In SDN, the logically centralized controller has knowledge about the whole network. This global knowledge of network is useful in design of the appropriate security policies for the network, which further helps in detection and mitigation of DDoS attacks (Bhushan & Gupta, 2018).

As in Figure 2.5, DDoS attack targets the SDN controller by sending a huge number of different spoofed source addresses to the OF switch. When a new packet is received at the switch the packet waits until a new flow entry is installed in the flow table, while the headers of these packets must be sent to the controller. Receiving a large number of new incoming packets in this situation will cause the overflow in both the packet queue and the flow table within the switch because of the size restriction of both of them.

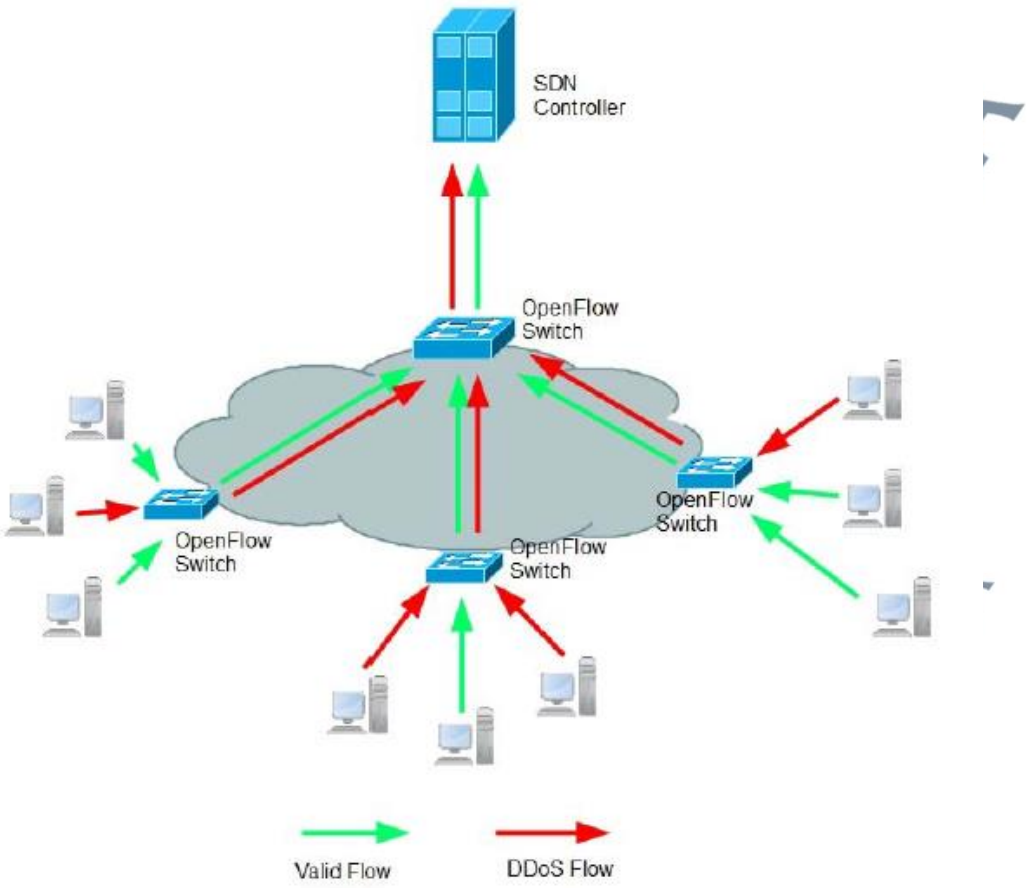

Figure 2.5: DDoS attack scenario in SDN (Adapted from Dharma et al., 2015)

As a result, the switch will breakdown very quickly. When the switch is down all packets from all over the network will be directly forwarded to the controller for processing. Keeping the controller in this position for a while will eventually lead the controller to be out of resources and unable to handle any new incoming requests (Shin & Gu, 2013). Breakdown the controller means the crash of the entire network. This failure point in SDN networks is the vulnerability used by the DDoS attackers to cause maximum damages.

Using the SDN characteristics, the attacker can launch DDoS attacks to overflow the flow tables to overload the switch which will cause of higher bandwidth

consumption in the links between the switch and the controller. This will eventually lead to saturate the controller resources such as CPU, bandwidth, and memory.

2.7.1 Flow Table overflow

The DDoS attacker can send a large number of table-miss packets to the victim switch. The victim switch should buffer them and generate flow requests sending to the controller since it cannot find matching rules for them. Because of limited resources (CPU and memory), the switch can only generate a limited number of flow requests. (Wang et al., 2014) show that a hardware switch can only generate less than 1000 requests per second. Thus, the switch may be overloaded, and as a result, flows from benign users may be delayed or dropped. Furthermore, if the controller processed flow requests successfully, a huge number of flow rules should be distributed to the victim switch. Since TCAM is a scarce resource, it only supports a small number of flow rules. For example, the Pronto-Pica8 3290 switch can only hold 2000 rules (Katta et al., 2016). Thus, the flow table of victim switch will be filled up quickly and eventually overflow. These two threats in the switch have a local impact as they reduce the throughput of the victim switch. In addition, the low-rate DDoS attacker can send a large number of packets to the victim switch to increase the number of packet-in messages for controller's resources depletion.

2.7.2 Bandwidth Consumption

The controller receives the flow requests flooded by the victim switch through the data-to control channel, with a lot of bandwidth requirements. The switch sends the entire packet instead of just a packet header to the controller if the buffer of the victim switch fills up. This results in higher bandwidth consumption. This links will overwhelm and the normal flow requests will experience congestion.

2.7.3 Controller Resource Saturation

Finally, the controller's resource (i.e. CPU, memory, and bandwidth) will be consumed for flow rule computation and installation if the flooded flow requests arrived at the controller. Legitimate requests may be dropped as results of the controller's resource saturation.

2.8 Flooding DDoS Attack Detection Techniques in SDN

The techniques proposed to tackle the DDoS attack problems in SDN are varied in terms of the techniques they have used to detect the attack. The related surveys of the previous researches had focused generally on two main categories the machine learning-based techniques and entropy-based techniques. In this section, the pros and cons of the proposed machine learning based solutions and the entropy-based solutions will be described. In terms of used parameters, the accuracy achieved, false alarms produced and the overhead measured, the mentioned solutions are described. A deep comparison of the different solutions discussed in the literature review is given in table 4.

2.8.1 Machine Learning Techniques

A lightweight method proposed in (Braga et al., 2010) to detect DDoS attacks based on Self-Organizing Maps (SOM). It is built into the first SDN controller called NOX controller. This method is a machine learning based method meant to learn the behaviour of the network to decide whether an attack is in progress or not. SOM is an unsupervised artificial neural network trained with the features of the network flow that is periodically collected from the switches. The traffic is classified as either

normal or abnormal based on the SOM pattern. In three modules, this detection method runs periodically within a loop in the NOX controller.

The switches are queried periodically by the flow collector module for their flow tables. The main features that are studied for DDoS attack detection are extracted by the feature extractor module and gather them in 6-tuples. Average of packets per flow, average of bytes per flow, average of duration per flow, percentage of pair flows, growth of single-flows and growth of different ports are the main parameters that are calculated based on the collected features and will be studied in the next module for the traffic classification. The given 6-tuple must be analysed and decided by the classifier module whether it corresponds to a DDoS attack.

In Braga's method, the elapsed time parameter is not among the parameters used to detect the attack. Rather than that, this method querying the switches periodically to extract the features average of packets per flow, average of bytes per flow, average of duration per flow, percentage of pair flows, growth of single-flows and growth of different ports which puts an extreme overhead on the controller. Also, the time taken to extract these features from the traffic is another reason for increasing the system overhead. Furthermore, processing this high volume of flows in the flow tables is another issue that must also be well-thought-out. The accuracy achieved in this work is very high but it costs very high overhead.

Finding the large spikes in traffic that could be signs of an attack have been proposed to monitor the flow statistics sent from the open flow switch to the controller (Dillon et al., 2014). After the OpenFlow controller finds the sources of the attack, flows are installed on the switches to drop the traffic from the suspected sources for mitigation. The detection techniques proposed are using packet symmetry and temporary

blocking of the traffic. A symmetrical behaviour exists between the two sides of a communication in routine traffic state while the symmetry ratio is analysed in the network and sources with high asymmetric ratio are suspected of an attack in the learning phase. In temporary blocking the flows are blocked for a short period and the traffic behaviour to this blocking is used to analyse if the traffic is legitimate or not. Sampling, blocking and analysis are the three phases of this strategy.

In this work, the detection method is based on packet symmetry neither packet arrival time nor elapsed time. Sampling the packets for the active flows in the switch is used in the detection face but using sampling to decrease the overhead is affecting the accuracy. In addition, blocking the flows temporarily to check the traffic behaviour will increase the switch overhead. Although having sample traffic from and to the victim helps to analyse the ratio between requests and replies, create flows for incoming and outgoing sample traffic from the victim of a DDoS attack and mirror this traffic to the controller will increase the controller's overhead.

Dotcenko et al. (2014) proposed a security mechanism that uses the combination of rate limiting (Williamson, 2002) and Threshold Random Walk with Credit Based Rate Limiting (TRW-CB) (Schechter et al., 2004) algorithms with fuzzy logic inference. This is considered one of the most pragmatic approaches in solving fuzzy modelling problems as it is based on fuzzy logic. Results show that decision making based on fuzzy rules is better than using the security algorithms, separately.

Results of this work show accuracy in detecting the attack but they did not show at what cost of resources usage this accuracy has been achieved. Another observation is that this mechanism does not use or even consider the packet arrival

time or elapsed time as a parameter can enhance the performance among the different parameters used by the two algorithms that combined in this work.

Tang et al. (2016) implemented a deep learning (DL) algorithm to detect the network intrusion and to evaluate the proposed network intrusion detection system model. However, it provided an insight in the integration of the model within an SDN environment; results were not good enough to be adopted in any alternative solution for signature-based intrusion detection system.

In terms of accuracy, the results of this work were compared to the results of other machine learning algorithms such as the work presented in (Braga et al., 2010). While the detection rate in Braga et al. (2010) was 99.11%, the accuracy achieved by this algorithm is only 75.75%. The main reason may be the feature selection for training and testing. In Braga's work, the six features were mainly specified for detecting DDoS attacks while features of this DL algorithm were used to detect DoS and other types of attacks. In addition, the resources usage of the DL algorithm used in this work was not evaluated. However, both of the solutions increase the CPU usage due to the time taken for training and testing. Moreover, this work has concentrated on six features, number of data bytes from source to destination, number of data bytes from destination to source and number of seconds of the connection length are three of them. Using the number of packets and the number of seconds between successive packets instead of the number of data bytes from source to destination and vice versa would be the appropriate choice to detect flooding attacks especially DDoS attacks. Also, using the number of flows instead of the number of seconds of the connection length would be the proper choice to detect attacks that create extra flows by the half-open connection process such as SYN

DDoS attacks. Using the new statistical information will reflect positively on accuracy and CPU usage.

Niyaz et al. (2017) proposed a detection system that can identify individual DDoS attack class and classify the traffic in normal and attack classes. The deep learning detection system presented in this work can identify the hosts with normal traffic and the ones with attack traffic by separately extracting features for each host which has incoming traffic for an interval. Accordingly, the controller can install flow rules inside the switches to block the traffic for a particular host if it undergoes an attack.

The proposed system has a number of limitations in terms of processing capabilities. The traffic collector and flow installer (TCFI) and feature extractor (FE) modules collect every packet to extract features and are implemented in the controller for low false positive in detection. Also, time taken for features extraction is high due to a large number of features to be extracted and the extraction process conducted for every host. In addition, the time of training and testing of the extracted features is always considered high in these techniques. These limitations limit the controller's performance by increasing the controller's bottlenecks. The performance of the controller in this work can be enhanced by choosing the appropriate parameters for DDoS detection without the need for extracting features for each host. Among the large number of features extracted in this work, packet arrival time does not appear.

2.8.2 Entropy Techniques

Mehdi et al. (2011) argued that tasks of network security should be delegated to the networks of home and office ISPs instead. A security policy implementation is

delegated to the downstream networks in the presented work. Four prominent traffic anomaly detection algorithms, threshold random walk with credit based rate limiting (TRW-CB) (Schechter et al., 2004); rate limiting (Williamson, 2002), maximum entropy detector (Gu et al., 2005) and Network Traffic Anomaly Detector (NETAD) (Mahoney, 2003) are implemented in NOX controller.

While this work observed that the anomaly detection can function well at line rates without any performance degradation, implementing separate pairs of working sets and delay queues for every internal host will increase the switch processing overhead. Furthermore, Maximum Entropy algorithm requires the inspection of every packet for building class distributions which will increase the controller CPU usage. However, using this algorithm will negate OpenFlow's efficiency benets. Also, keeping the suspicions connections that have yet to receive a response (i.e. a SYNACK) wait until they get a reply or times out to only install two healthy flows will degrade the performance. Moreover, the authors only used the low rate network traffic to do the experiments as they focused on the home environment and did not discussed the overhead to the control plane when use these approaches in high rate traffic situation. In addition, the four algorithms are implemented in NOX controller which is not in use.

Giotis et al. (2014) implemented a widely used entropy based approach (Lakhina et al., 2005) to detect DDoS, portscan attacks and worm propagation effectively. The source and destination IP addresses and source and destination ports are the flow-related traffic features that used to detect anomalies. To identify the presence of anomalies, predefined thresholds on changes in the entropy values have been used. In

this work, a mechanism combining OpenFlow and sFlow is proposed to decrease the controller overhead problem.

As the implementation was based on flow sampling using sFlow, false-positive was quite high in attack detection. The sFlow-based mechanism is reduced the CPU load compared to the native OpenFlow approach used but this CPU performance was measured in NOX controller as the controller used in the experiments. However, NOX is no longer in development (McCauley, 2014). Further, starting the detection function every 30 seconds will give the attack packets the chance to be cumulated and eventually fill up the flow tables in the switch.

Mousavi (2014) proposed a method uses the entropy variation of destination IP address as an early detection of the pox controller. Entropy is defined as a measure of randomness. The maximum entropy occurs when every incoming packet is destined for a different host whereas the minimum entropy is seen when all the packets are destined to the same destination address. The method proposed in the work uses the destination IP for entropy computation. The entropy is calculated for destination IP addresses of a studied window of packets. An attack will be reported if the calculated entropy is less than the threshold for a continuous number of times.

In this work, the proposed entropy detection function will result in false positive attack detections when the load of the traffic increases in the network with legitimate traffic in the peak times. CPU usage increases with window size. The low CPU usage is only measured when the entropy computed to the smallest window size. The proposed entropy may not be reliable since threshold value will vary in different scenarios.

Wang et al. (2015) proposed a distributed algorithm for entropy-based anomaly detection scheme running on OpenFlow edge switches. Calculation of the entropy is using probability distribution of destination IP address at each switch. Alert information is sent to the controller once the DDoS flooding attack is detected. In this algorithm, the authors extend the flow entry in the OF table by adding a copy counter (RP Local) of *Received Packets* counter to the traditional OpenFlow edge switch due to it does not know its local topology and whether a IP address belongs to its local network or not. Also, due to the traditional OpenFlow edge switch does not know the packet number of a specific flow handled by it during an interval, they add the copy counter (RP Local) of *Received Packets* counter.

In this research, the additional storage that contains the copies of the counters and the previous entropy values will result in increasing the switch memory usage that will increase the switch processing overhead. Therefore, the communication overload between OpenFlow switches and the controller will be increased. Moreover, the algorithm attempts to bring back the intelligence in network forwarding devices by the extension made to the flow entries in the OpenFlow switch tables.

Kia (2015) proposed a DDoS detection algorithm designed based on three main concepts. These concepts are entropy variation of destination IP address, flow initiation rate and study of flow specification. The phases of this detection algorithm are: Data Collection, Entropy calculation and comparison, Flow initiation rate computation and comparison, Finding the possible attack path(s) if an attack is suspected in the second or third phases, otherwise returning to the first phase, Polling the switches in the attack path for flow statistics and studying the returned results

from the switches to confirm or cancel an attack state and updating the thresholds according to the detection result in the previous phase. After an attack is detected a mitigation method will be applied to prevent the network switches from breaking down and to give enough time to network administrators to take the required actions. The mitigation approach set the flow idle_timer to a small value for the new flows.

Although entropy has proven to be a successful detection method, using entropy cannot detect many attack scenarios. For example at peak times with the sudden rise of legitimate traffic, the demand to a certain network destination such as the web server or email server grows and the entropy based detection method may continuously report false positive alarms. On the other hand, when the attacker distributes the attack among many victims, the entropy may not show a significant decrease and so it will result in a false negative report. In addition, the CPU usage still considered high when entropy based DDoS detection solutions are used. However, entropy-based solutions have low overhead compared to machine learning based solutions.

Typically, entropy-based detection methods detect unexpected changes in the time series of the entropy of certain traffic features. When entropy is calculated, the probability distribution of a feature is represented by a single value. This is very effective for analysis but however, relevant information about the distribution of the analysed feature is lost though. Similarly, the different distributions with the same amount of uncertainty cannot be distinguished by values of entropy. Hence, the anomalies which do not disturb randomness remain undetected (Javed et al., 2009). However, the anomaly detection schemes accuracy may be affected by flow rate sampling (Wang et al., 2015).

2.9 Low-Rate DDoS Attack Detection in SDN

Although authors have proposed many DDoS attacks detection solutions, we found that few works have considered low rate DDoS attacks in SDN.

Dong et al. (2016) designed a detection method to locate the compromised interfaces where the attackers are connected. In this work, a detection method proposed to detect DDoS attacks against SDN controllers by vast new low-traffic flows. The detection method proposed in this work is comprised of a flow classification function and an attack detection function. The proposed detection method has advantages of promptness, versatility, and accuracy. The goal of this work is to detect the attack and locate the potential interfaces that are compromised.

Although the promptness and accuracy are advantages of the proposed detection method in this work, accuracy was not presented in percentage and at what cost of CPU usage it is achieved. In terms of parameters used to detect low-rate attack packets, the classification function in this work counting the number of packets in each flow to classify the normal flow from the attack flow. In contrast, to classify the normal flow from the attack flow, the detection function in our scheme is counting the total number of packets in all flows. While counting the number of packets in each flow can increase the switch processing overhead especially if there are lots of switches in the network, counting all single flows that created by single packets will decrease the overhead and detect low-rate attack packets fast and accurate. Moreover, the proposed detection method has been evaluated based on DARPA DDoS attack dataset as one of the publically and extensively used available datasets. According to Behal and Kumar (2016), the publically available datasets,

including DARPA dataset, suffered from a number of limitations which limits their usage for validating DDoS research.

The detection mechanism proposed in (Sahoo et al., 2018) is an extension to the idea in (Xiang et al., 2011), and it also explore the Generalized Entropy (GE) and Generalized Information Divergence (GID) metrics to detect low rate DDoS attacks. This detection mechanism uses the information distance metric as no work has adopted the information distance metric for the attack traffic detection purpose. This work uses entropy as an essential concept in information theory. This mechanism uses the information distance metric to detect control plane DDoS attack. The detection mechanism extracts the traffic statistics from the flow table and imposes the GE metric to make an early alert for the attack.

Using entropy calculation to detect DDoS attack has a limitation which is depending on selecting a threshold value after performing several experiments. Selecting the optimal threshold value is made by specifying a window size for every rate, which makes threshold value vary in different scenarios. Consequently, the accuracy rate may not be reliable due to increasing either false positives or false negatives rates. Entropy used lots of math calculations to find the optimal value of the threshold. However, the accuracy rate and CPU usage to detect the low rate DDoS attacks have not mentioned in percentage in this work.

Table 2.4: Comparison among various solutions of machine learning and entropy

	Solution/developer	Attack traffic	Parameters	Accuracy	Overhead
Machine Learning	Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow/(Braga et al., 2010)	TCP, UDP and ICMP	Average packets per flow, average bytes per flow, average duration per flow, percentage of pair flows, growth of single flow, and growth of single ports.	99.11% of accuracy and a very low False Alarms rate of 0.46%	Very high due to the time taken to extract lots of features and to train the features set.
	Detection and blocking of attacks method. /(Dillon et al., 2014)	TCP and HTTP	Number of packets, number of bytes, packet symmetry.	No accuracy percentage provided	Overhead increased due to creating flows for incoming and outgoing sample traffic from the victim of a DDoS attack and mirror this traffic to the controller.
	A Fuzzy Logic-Based Information Security Management for SDN/ (Dotcenko et al., 2014)	Either TCP or UDP	Connection initiations packets and response packets. Additionally input parameters for the system may contain statistical data from switches are required such as data speed of selected flows and ports and minimum and maximum number of	95% of accuracy at 1.2% false positives	The two algorithms used in this work gave a remarkable usage when implemented on the NOX controller but the impact of using them with a fuzzy logic interface on the Beacon controller resource usage is not

			packets per one IP.		measured.
	Deep Learning Approach for Network Intrusion Detection in SDN/(Tang et al., 2016)	TCP, UDP and ICMP	duration, protocol_type, src_bytes, dst_bytes, count and srv_count.	The best accuracy achieved among the different learning rates is 75.75% at loss rate 19.51	While the training time for the features of the machine learning techniques always high, the resources usage is getting high accordingly.
	A Deep Learning Based DDoS Detection System in SDN/ (Niyaz et al., 2017)	TCP, UDP and ICMP	Headers extracted from TCP (Src IP, Window, Dst IP, SYN, Src Port, ACK, Dst Port, URG, Protocol, FIN, Data Size RST, TTL, PUSH) From UDP (Src IP, Dst IP, Src Port, Dst Port, Protocol, Data Size , TTL) And from ICMP (Src IP, Dst IP, ICMP Type, ICMP Code, Protocol, Data Size, and TTL) and a large number of features extracted from these packets headers.	99.82% of accuracy at 0.3% False positive Rate for the 2-class model and 95.65% of accuracy at 0.5% False positive Rate for the 8-class model	Overhead is high due to collecting every packet to extract features which limits the controller's performance. The time taken to extract a very large number of features

Entropy	Revisiting Traffic Anomaly Detection using SDN/ (Mehdi et al., 2011)	TCP SYN, TCP SYN flood and fraggle (UDP flood)	Parameters used in the four algorithms are connection initiations packets, response packets, protocol type, destination port number, all non-IP packets, all incoming traffic, TCP packets starting after the first 100 bytes and packets to any address/port/protocol combination if more than 16 are received in a minute.	Both of TRW-CB and Rate Limiting algorithms on the home network dataset achieved 90% of accuracy. On the ISP dataset, they both suffer accuracy degradation where the best accuracy of TRW-CB was 85% for false positive 70%.	The CPU usage measured for different data rates for the four algorithms is a remarkable usage but it is measured on NOX controller. NOX is no longer in development. NOX controller has upgraded to POX controller which is the controller currently in use.
	Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments/ (Giotis et al., 2014)	TCP	Src IP, Dst IP, Src Port and Dst Port	The other two algorithms Maximum Entropy and NETAD have achieved accuracy of 90%. 100% accuracy with a False Positive rate ranging from 23% to 39.3%. The sFlow-based implementation performed slightly worse, achieving 100% accuracy rate with a False Positive rate around 50%.	The OpenFlow Virtual Switch CPU load of the sFlow implementation was significantly less, compared to the native OpenFlow implementation Also, the CPU performance was measured in NOX controller.

				False-positive was quite high in attack detection due to the fact that sFlow method is based on flow sampling.	
	Early Detection of DDoS Attacks against SDN Controllers/(Mousavi , 2014)	UDP	Number of packets and Dst IP	96% of accuracy for a window size of 50 packets. Increasing the window size will increase the false positive.	CPU usage increases with window size. The CPU usage reaches 78%. The low CPU usage is only measured when
	An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined Networking/(Wang et al., 2015)	UDP	Dst IP, number of packets that have same Dst IP and number of times Dst IP is repeated	The algorithm can achieve 100% detection rate while has approximately 25% false positive ratio for both 100 and 500 Mbps background traffic.	The CPU usage increases with window size. The low CPU usage is only measured when the entropy computed to the smallest window size.

	Early Detection and Mitigation of DDoS Attacks In Software Defined Networks/ (Kia, M., 2015)	UDP	number of packets, Dst IP, total number of Dst IP and number of times Dst IP is repeated	When the attack and legitimate traffic have distinct characteristics as it was in traffic pattern "A" and the window size is as small as suggested in this work which is fifty packets, the algorithm is able to perform with a 100% detection rate. But when the traffic pattern has the assumption is that legitimate traffic is a mix of short and long flows, the algorithm starts to show an increase in false positive detections. This shows decreasing in the accuracy rate to 90% and a false positive rate at 10%.	The additional storage that contains the copies of the counters and the previous entropy values will result in increasing the switch memory usage that will increase the switch processing overhead. Therefore, the communicati on overload between OpenFlow switches and the controller will be increased. A lot of computations are performed to generate the required data which consumes a lot of network and controller resources, thus, the controller's overhead will be increased. The complexity is appeared by adding 300 lines of coding to the controller.

2.10 Popular SDN-Based DDoS Attack Mitigation Methods

SDN has been focused to improve network flexibility and agility. It delegates networks for quickly responding to the changes in the network requirements via a centralized controller. Therefore, a global view of the network is provided by the SDN controller. Furthermore, the concept of the centralized controller leads to a consistent and robust configuration throughout the network. Since all network policies are defined at the centralized controller, it not only simplifies anomaly detection but also paves the way for mitigation mechanisms (Ali et al., 2015). For instance, when a DDoS attack is detected, a threat mitigation application may effectively reprogram switches to mitigate the suspicious traffic.

Since the effective detection mechanisms alone are not enough to end the dangers of a threat, so many SDN-based DDoS defence methods incorporate mitigation strategies. The commonly used mitigation strategies that are extensively deployed in SDN are dropping packets, blocking ports, and redirecting traffic (Dillon et al., 2014; Giotis et al., 2014; Chin et al., 2015; Xing et al., 2013; Chung et al., 2013; Lim et al., 2014). Nevertheless, the deep packet inspection method, changing MAC and IP addresses technique, and isolating traffic which is mitigation strategies are also implemented in SDN-based DDoS defence mechanisms (Xing et al., 2013; Chung et al., 2013). A summary of prevalent DDoS mitigation methods is briefly described in table 5.

Table 2.5: Various DDoS mitigation methods

Attack Mitigation methods	
Drop packets	The network traffic conforming to defined rules is transmitted and remaining is dropped
Block port	The network traffic from attacking port is

Redirection	completely blocked The legitimate traffic is redirected to new IP address

Dropping packets and blocking ports are simple and fast mitigation techniques which are completely used to block the attack sources potential. These mitigation mechanisms may result in dropping the legitimate traffic if they were not based on accurate detection methods. For instance, a legitimate node or port is completely blocked if this node or port being compromised which resulting in dropping of any legitimate traffic it may generate (Dillon et al., 2014; Giotis et al., 2014; Chin et al., 2015).

From my standpoint of view, dropping packets technique is better than blocking ports technique. Blocking a port will cause of blocking all network traffic coming from this port while dropping packets will only drop the mentioned packets that signed as malicious packets. Even in the case of false detection alarms, a few packets will only be dropped and the other packets will pass through that port.

After the attack is detected, the traffic redirection technique forwards the traffic to a new IP address on the available server. In order to prevent bots from directly accessing the redirected address, all connections to the existing server are torn down. Nonetheless, this technique increases the network traffic processing time due to redirecting the address that analyses and forward only legitimate traffic using deep packet inspection (Xing et al., 2013; Chung et al., 2013).

2.11 Related Work

Due to the unique nature of this work involving elapsed-time based attack detection in an SDN, and unavailability of existing literature in this specific domain, it was difficult to find a related work other than this work.

YuHunag et al. (2011) from Chungwa Telecom Co. proposed an OpenFlow DDoS Defender that monitors flows on an OpenFlow switch. If the number of packets received in 5 seconds exceeds 3000 then the number of packets will be studied in per second duration. If the number of packets per second exceeds 800 for 5 continuous times then an attack is detected and the DDoS defender will start dropping the incoming packets until the flow entry times out. The method used is illustrated in Figure 2.6.

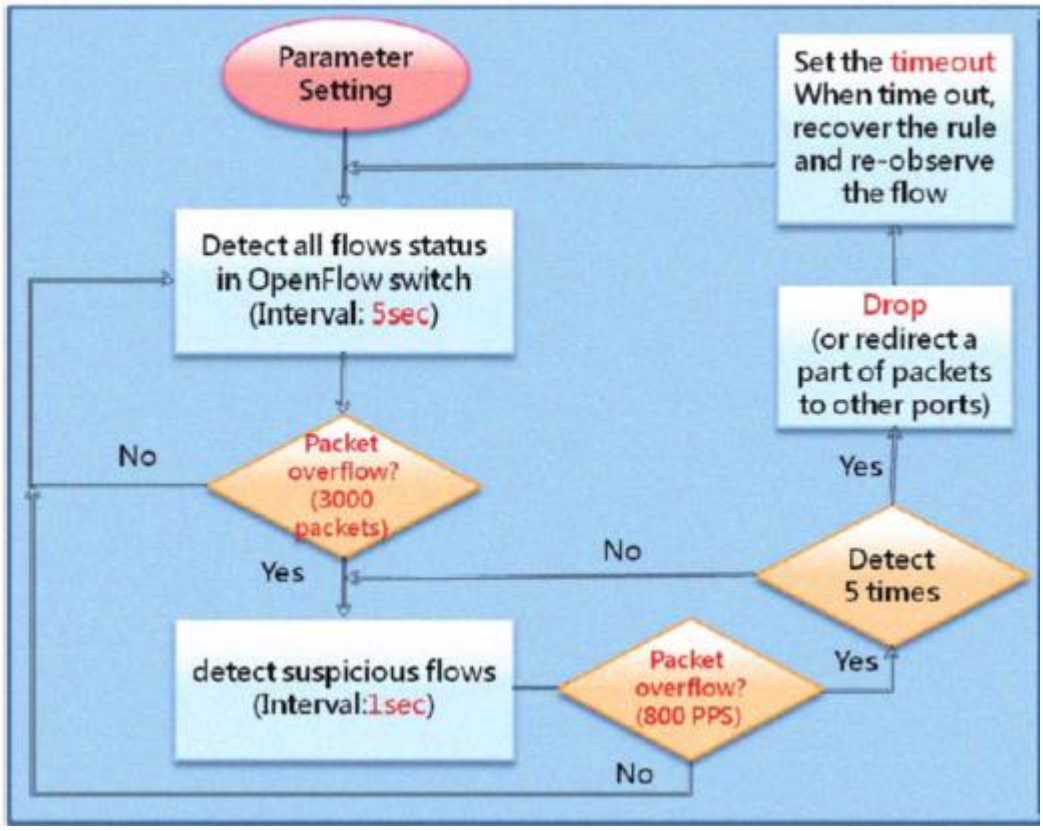

Figure 2.6: DDoS defender flowchart (Adopted from YuHunag et al., 2011)

This work mainly focuses on the speed of the packets which is studying the number of packets in per second duration. Also, this work concentrates in its advanced stage on a particular number of packets that repeated in a particular number of times. This work does not take into account the sudden increase in the number of the flows caused by packets that have been sent at low traffic rates. This will eventually lead to miss detecting the attack packets being sent by TCP flood DDoS attack as an example. Further, the method proposed in this work does not terminate the flows directly when the attack packets are detected but it waits until the flow is times out instead. As consequence, especially in the case of low rates packets, the number of the flows in the flow tables will increase dramatically to eventually make the switch overflow. However, no CPU usage is measured and no false alarms percentage provided in this work.

2.12 Summary

This chapter presents the DDoS attacks and the effects of these attacks on SDN components. Also, this chapter presents SDN features such as providing a unique opportunity for effective detection and containment of network security problems and allowing the integration of complex network security applications in large networks. SDN also offers flexible network management by decoupling forwarding and control planes, in its architecture; network management is logically centralized at the control plane, while the forwarding plane only needs to forward packets under the manipulation of the control plane. Enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services makes the network more flexible and agile. Due to its flexibility, programmability and maintainability, SDN has been widely studied for

its applications in different types of networks. In this chapter, an illustration of the SDN layers has been shown to demonstrate the features provided by SDN such as the separation between the control plane and data plane. Other illustrations have been shown in the chapter such as the flow table entry and the procedure of flow processing in OpenFlow.

SDN security, especially DDoS attack, has become a popular research field since software defined network was proposed. The characteristics of centralized control and programmability of SDN make it easier to detect and react to DDoS attack than traditional networks. In this chapter, an investigation has been done throughout section 2.8 to demonstrate the pros and cons of the SDN-based solutions proposed to detect DDoS attacks either uses machine learning technique or entropy technique. Table 4 presents a comparison among these solutions in terms of the following: attack type, parameters, accuracy, overhead and false alarms.

This research presents the effects on the performance of the DDoS defense solutions by analyzing the previously proposed solutions for detecting DDoS attacks in SDN in terms of the detection technique on one hand and the parameters used for detection on the other hand. Precisely, the effects of the detection technique and the parameters on the performance metrics: CPU usage, accuracy, and false alarms. While using the entropy techniques may affect the accuracy by producing high false alarms and using the machine learning techniques may increase the overhead, thus, the techniques discussed in this chapter show clearly that the problem with existing researches is the lack of solutions that have an efficient and less complex that produce low false alarms in the SDN environment.