

**identifying and Preventing the attacker in SDN Controller Ryu**

**Thesis**

**Submitted By**

|  |  |
| --- | --- |
| ID | Name |
|  |  |
| 16-32939-3 | Shovik,Abdullah Al |
|  |  |
| 16-32861-3 | Plaban Imtesar aziz |
|  |  |
| 16-32194-2 | Islam,Sumaiya |
|  |  |
| 16-32104-2 | Farzana Akter Eila |
|  |  |

**Department Of Computer Science Faculty of Science & IT America International University Bangladesh Fall, 2019-2020**

**Declaration**



We declare that this thesis is our original work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Shovik,Abdullah Al**

**Plaban imtesar aziz**

16-32939-3

16-32861-3

CSE

CSE

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Islam,Sumaiya**

**Frazana Akter Eila**

16-32194-2

16-32104-2

CSE

CSE

**Approval**

The thesis titled “**Indentifying and preventing the attacker in SDN Controller Ryu”** has been submitted to the following respected members of the board of examiners of the department of computer science in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science on (19-09-2019) and has been accepted as satisfactory.

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **Wardah Saleh** |
| Lecturer & Supervisor |
| Department of Computer Science |
| American International University-Bangladesh |

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **DR. M. M. MAHBUBUL SYEED** | **Professor Dr. Tafazzal Hossain** |
| Associate Professor & Head (Undergraduate) | Dean |
| Department of Computer Science | Faculty of Science & Information Technology |
| American International University-Bangladesh | American International University-Bangladesh |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Dr. Carmen Z. Lamagna**

Vice Chancellor

American International University-Bangladesh

**Acknowledgement**

We have taken great effort in creating this Thesis- "Indentify and Preventing the attacker in SDN". However, the successful completion of this project could not have been possible without the great help and support of many honorable individuals. First of all, we would like to be grateful to Almighty Allah who gives us the effort to work on this project. Thanks to our respected parents for their outstanding support at every point of our lives. We would like to thank our honorable supervisor Wardah Saleh ma’am for his enormous support, inspiration and helpful criticism. We are very grateful to him for giving us the opportunity to work with him. Without his continuous support, it would be very difficult for us to complete this project. We would also like to thank all the faculty members for their guideline for making the proper documentation of our project. We wish to express our gratitude towards American International University-Bangladesh (AIUB) for providing an excellent environment for research.

**Abstract**

Software defined networking (SDN) intends to manage computer networks flexibly by using a central controller to orchestrate switches on the network planner. Moreover, operators using conventional protocols have three major problems when deployed SDN for fast and fault-tolerant routing. One is that the time scales of the control plane are too sluggish to ensure successful load balancing to use existing network power effectively. Furthermore, SDN commodity switches have limited memory to implement standardized policy requirements that compromise versatile control promises. Finally, it is a single failure point that is intolerable to operators that operate centralized fault tolerant network protocols.

Now there are many advanced quality controller, among them Ryu and pox being the most popular. Ryu controller detects and mitigates DDos attack so far there have been many algorithm implementations, but there are some lacks. So in this paper we have generated an updated algorithm for detection and mitigation of DDos ​​attack, where the DDos ​​attack will be prevented and Attacker will be identified.

Contents

[**Chapter 1**](#page8) **Introduction ………………………………………………………………………………….. 7**

[**1.1 Introduction to SDN:**](#page8) **7**

[**1.2 SDN Controller:**](#page8) **8**

[**1.2.1 Different SDN Controllers:**](#page9) **8**

[**1.3 DDos Attack:**](#page8)**……………………………………………………………………………………………………………..…….. 12**

**1.3.1 DDos Attack in SDN:………………………………………………………………………………………………..12**

[**1.4 Motivation:**](#page10)**………………………………………………………………………………………………………………………..12**

[**1.5 Research Objectives:**](#page10)**…………………………………………………...……………………………………………………..14**

[**Chapter 2**](#page11)[**15**](#page11)

**Literature Review 15**

[**4.1 DDos**](#page19) **Mitigation by Ryu Controller 18**

[**4.2 Problems**](#page19)  **19**

[**4.3 Experimental**](#page19) **Setup 20**

[**Chapter 3**](#page16)[**22**](#page16)

[**Specification of SND**](#page11)  **22**

[**2.1 Software Defined Networks:**](#page11) **22**

[**2.2** **Open-flow protocols:**](#page11) **23**

[**2.3 Open-flow Specifications:**](#page12) **24**

[**2.3.1 Open-flow Switch:**](#page13) **25**

[**2.3.2 Secure channel:**](#page13) **28**

[**Chapter 4**](#page19) **29**

[**Research**](#page16) **Methodology 29**

[**3.1 Distributed Denial of Service Attack and its Mitigation:**](#page16) **29**

[**3.2 Types of DDos Attacks:**](#page16) **29**

[**3.3 Anomaly Detection for DDos Mitigation:**](#page17) **32**

[**Chapter 5**](#page21): **Conclusion & Future work** [**33**](#page21)

[**5.1**](#page22) **Conclusion & Future Work** [**33**](#page22)

**References** [**34**](#page32)

**Chapter 1**

**Introduction:**

In this chapter the topic of distributed denial of service (DDoS) in the SDN framework will be discussed for the first time. Then we will preventing and indentify the attacker in SDN Ryu controller by modify a thesis paper. We will discuss our research objectives and study priorities. In conclusion, the research's contributions are outlined, followed by the thesis organization. In this paper, we have developed a modified algorithm for the detection and prevention of ddos attacks, where the ddos attack is prevented and Attacker is detected at the host earlier.

**1.1 Introduction of SDN:**

Software Defined Network is another way to view networks. The main objective is to increase network asset control. Controlling and transmitting behavior in current production networks are designed in the equipment by manufacturers and are primarily proprietary. The SDN system distinguishes the aircraft power and communication and allows network management to manage the plane. Such isolation is carried out through network optimization so that the relay provides orders to forward the incoming packets instead of using its power. The switch includes tables of communication flows. Open flow is the SDN architecture-orchestrating protocol. The application layer will provide a single view of the network over the control layer and the whole system looks like a conceptual transfer. The control system consists of a control system and a protected channel between controller and switch. The controller separates the network infrastructure from the application layer by the control panel. All setup and improvements can be made in real time by using the control layer. Every system does not have to learn different protocols in the network layer and the only function remaining is to forward them. The Open Networking Foundation has a number of providers, different groups who work on Open flow specifications and promote the adoption of SDN.

The main objective of SDN is to enable and customize the networking. SDN varies from conventional network architecture where each network system takes the decision on traffic on the basis of its defined routing table. The SDN model unites the network management and communication features, which allow for easy programming on the network operation. The main focus was on distinguishing the network control plane from the data plane.

**1.2 SDN Controller:**

SDN controllers are the network's "brains" in SDN. The app serves as the strategic control point within the SDN network, manages flow control over' below' switches / routers and above' smart networks ' application & business logic. A SDN Controller platform usually includes a collection of "pluggable" modules for various network tasks.

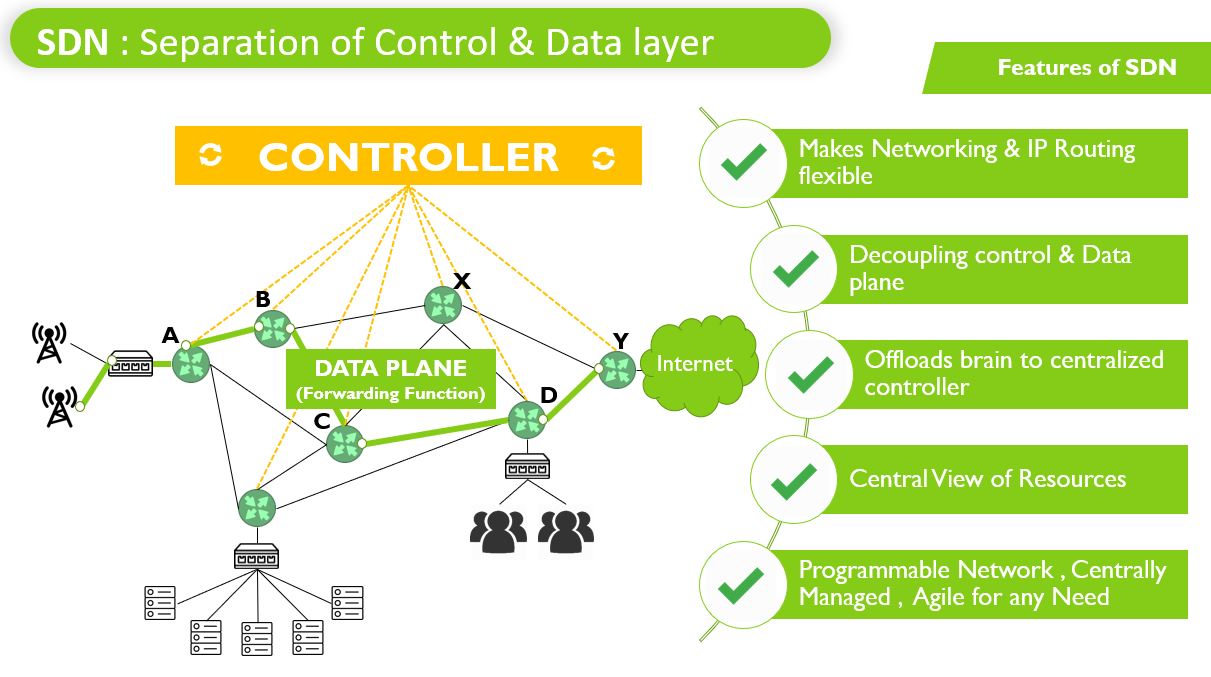


Fig: 1.2 Controller

**1.2.1 Different SDN Controllers:**

There are some SDN controller names:

1. NOX/POX

2. Open-Day-light

3. Ryu

4. Floodlight

5. ONOS

6. PYRETIC

7. TREMA

1. FRENETIC

**NOX/POX:** NOX is the first Open-Flow program. It serves as the basis for network management for the administration and development of high-level programmatic programming network control systems. The system-wide abstractions make networking an application challenge.

**🡪**Support C**++**

**🡪** Supports Open-Flow 1.0

🡪Slow programming

POX is an open source platform for SDN control systems based on Python software, such as OpenFlow SDN controllers. POX is becoming more popular than NOX, a sister project that makes rapid development and prototyping possible.

🡪 Python Code used in it

🡪 Supports Open-Flow 1.0

🡪Program are easy

**Open-Day-light:** Open-Day-light Project provides the Open-Day-light Controller, called the OpenDaylight Platform, with a community-led and industry-supported framework for improving application networking for the SDN Open Source. It is open to anyone interested with end users and consumers who can work together on a common platform to find new solutions.

**Ryu:** Ryu Controller is an open SDN controller that makes it easy to monitor and change how traffic is managed to improve network resilience. It's a tool that makes networking simpler. The SDN controller is basically the brain for the SDN network, talks with southbound APIs to switches and routers, and makes good sense with Northbound APIs. The Ryu Controller assisted by the NTT is as

🡪An open source python controller

🡪 Open-Flow 1.0,1.2,1.3 Nicira extensions Supports in it

🡪 Ryu is implemented in python as well as other SDN controllers like NOX do not perform.

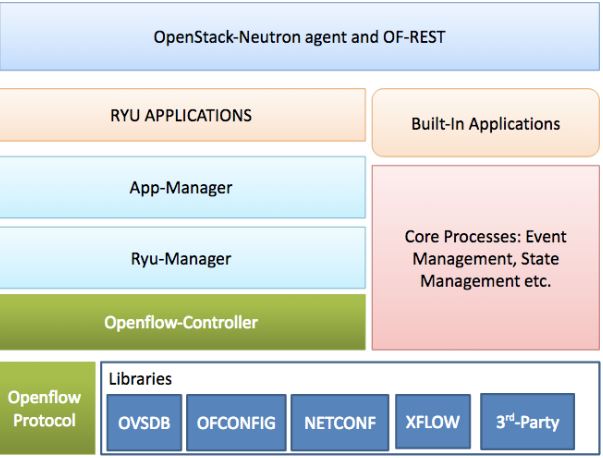


Fig: Ryu controller Architecture

**Floodlight:** Floodlight Open SDN Controller is a Java-based enterprise-class free flow controller licensed by Apache. A developer community that includes a variety of Big Switch Network engineers is sponsoring it. Open Flow is the Open Framework of a Universal Networking Foundation. It specifies a protocol that can alter network device behavior by flipping a remote controller with a well-specified "reach order." Floodlight is designed to work with the the number of common Open Flow switches, routers, virtual switches, and access points.

🡪Written in JAVA

🡪 OpenFlow 1.0 Supports

🡪It has a fairly step learning, By using Floodlight have to known about JAVA.

**ONOS:** The primary open source SDN framework for the development of SDN / NFV new generation technologies is Open Network Operating System (ONOS™). There was a mistake. The ONOS platform includes: a platform and a collection of features to serve as an extensible SDN controller.

**1.3 DDOS Attack**

Distributed denial-of-service attacks on internet and network targets . The goal is to increase traffic across them than can be accommodated in the server or network. The aim is to make the platform or device inoperative. The traffic may consist of incoming messages, link requests or bogus packets. In some circumstances a DDoS assault or a low level attack on the intended users is challenged. This can be paired with the risk of an abduction of a devastating attack if the organization does not offer a cryptocurrency ransom.

Recently, the rapid identification of the attack and the appropriate reaction mechanisms have been imperative with that the serious damage done by attacks by DDoS. There is no effective defense against these threats under existing security frameworks. Such attacks will kill a target easily and cause enormous loss of income. The port / protocol or process form used can easily differ in DDoS attacks. we are providing an exhaustive list of current DDoS attack prevention and mitigation technologies focused on SDN. We can use Ryu Minnet to reduce DDos attacks.

**1.3.1 DDos Attack in SDN**

Software Define Network (SDN) has many benefits over a traditional network. The major advantage of SDN is the physical separation of network controllers from delivery equipment. SDN is able to solve other existing network security challenges. Nonetheless, other bugs have been found with SDN. Distributed Denial of Service (DDoS) assaults are the main problem for SDN weaknesses. A major problem is the assault on SDN by DDoS and different methods for identification and prevention purposes have been implemented. The purpose of this paper is to suggest a method of detecting DDoS attacks and preventing the attacker earlier, using SDN-based techniques that can mitigate valid user behavior and to introduce the Advanced Support Vector Machine methodology (ASVM) as an enhancement to the current Support Vector Machine (SVM) algorithm for recognizing DDoS attacks. The ASVM methodology is a three-class multi-class classification system. In this article, two types of flood-based DDoS attacks can be observed successfully. Our detection technique can reduce both training time and testing time by using two key features, asymmetric features and namely volumetric.

**1.4 Motivation**

The Software Defined network design principle is a new and new way to manage the network. Switches do not manage incoming packets in SDN. You are searching for a fit of the input packet in your forwarding tables and if there are none, they will be sent for processing to the dispatcher. SDN operating system is the master. This reads the packets and determines whether or not the message is sent on the switch. SDN distinguishes delivery and distribution arrangements when implementing this protocol. Now there are many advanced quality controller, among them ryu and pox being the most popular. Ryu Controller is a free, software-defined network controller (SDN) designed to expand the network's mobility by making it easy to monitor and change how traffic is managed. The SDN Controller is usually the brains belonging to the SDN system, speaking information down to switches and routers with southbound APIs, and up to functions and business good judgment with northbound APIs. Using NTT, the Ryu Controller is supported and also implemented in NTT cloud statistics centers. Ryu is an open source python controller, Supports OpenFlow 1.0,1.2,1.3 Nicira extensions Ryu is implemented in python as well as other SDN controllers like NOX.

The SDN architecture can be a multi-controller network with links to the network of switches. Each of these networks can be seen as a part of the network with its master. In order to protect against DdoS we focus on each of those slices. The network sacrifices its operating speed if the link between the switches and the controller is broken. It ensures that packets are no longer stored in the network, and the SDN design becomes destroyed when the interface is disabled.

A DDoS assault is one of those choices that can make the controller unattainable. A large number of packets are sent to a server or a network host party in DDoS attacks. If the incoming packet's source addresses, the normal theory, are spoofed, the switch does not fit and has to forward the packet to the controller. The set of legitimate packages and the DDoS spoofing will connect the controller resources through continuous processing that complicates them. This will make 2 the controller unattainable for the newly arrived valid packets and may bring down the controller triggering the software loss to the SDN. Even if there is a controller for the backup, the same obstacle must be met.

The main objective of this research is to detect a DDoS attack. Preventing and indentify the attackers in its early stages. The early term is based on the network itself. Because the controller program can run on a laptop or a powerful desktop, the early term will rely on system tolerance and traffic properties. If the detection takes place in the first few hundred packets, however, the mitigation will be implemented before the controller is completely swamped with the large number of malicious packets. A simple controller DDoS attack where the usual incoming packet rate is about 100 packets per second. The rate rises rapidly to about 250 packets per second when the attack occurs.

Ryu controller detects and mitigates Ddos attack so far there have been many algorithm implementations, but there are some lacks. So in this paper we have generated an updated algorithm for detection and mitigation of DDos ​​attack, where the Ddos ​​attack will be prevented and Attacker will be identified.

**1.5 Research Objectives**

In this research, we studied Ddos attack detection and mitigation in Ryu Controller and find possible weak points of their work to DDoS attacks. This study is about

Preventing and identifies the attacker earlier at hosts in SDN. We consider the controller to be the weak link in a scenario of DDoS assault. With a view to protecting the server, we examined various methods that could be used in the controller in DDoS detection. SDN's framework, however, raised its constraint on the solution form and how it was applied.

These limitations were:

1. Limited resources of the controller.
2. Due to the large number of malicious packets, the need to detect an intrusion before the controller is out of reach.

The main objectives are:

a) If the attacker has already attacked the host?

b) If the attacker uses the USB port to pass data to the host or receive data?

**Chapter 2**

**Literature Review**

Ryu controller detects and mitigates Ddos attack so far there have been many algorithm implementations, but there are some lacks. So in this paper we have generated an updated algorithm for detection and mitigation of ddos ​​attack, where the ddos ​​attack will be prevented and Attacker will be identified earlier at host. First they detect the Ddos attack, if the same packet size, protocol and port numbers are used in massive number of packets there have DDoS attack. Real switches with open-folw support, they have used Ryu to mitigate DDos attack. There is a python code set in Ryu controller which can count the traffic rate of coming packets. When traffic rate of coming packets are increased, the server will get down and those packet will drop.

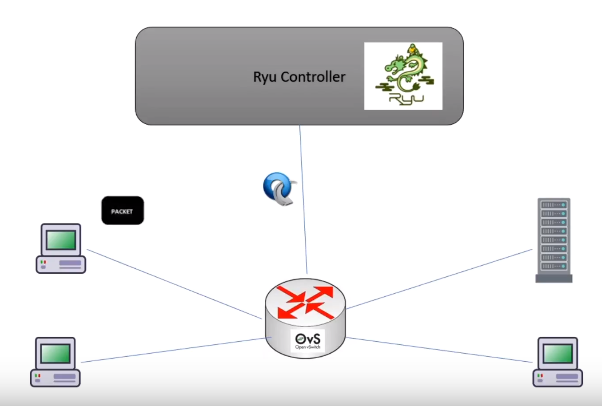


Figure 4.1: Network Diagram

**Step 1**: whenever a host sends to package to the source, the switch check the flow table and if there is a table miss entry. The package is forwarding to the controller.

**Step2**: whenever is there package place event taking place, our Ddos application passes the package and looks up for the header information. Here, we will write a code such a way, it will check for the source and destination IP address.

**Step 3**: After processing the IP header information of the package, the controller will send a flow ruled to the switch. Now the controller will send the package out message to the switch. And then switch forward the package as per the flow entry.

Here, their Ddos attack Detection Algorithm Flowchart:

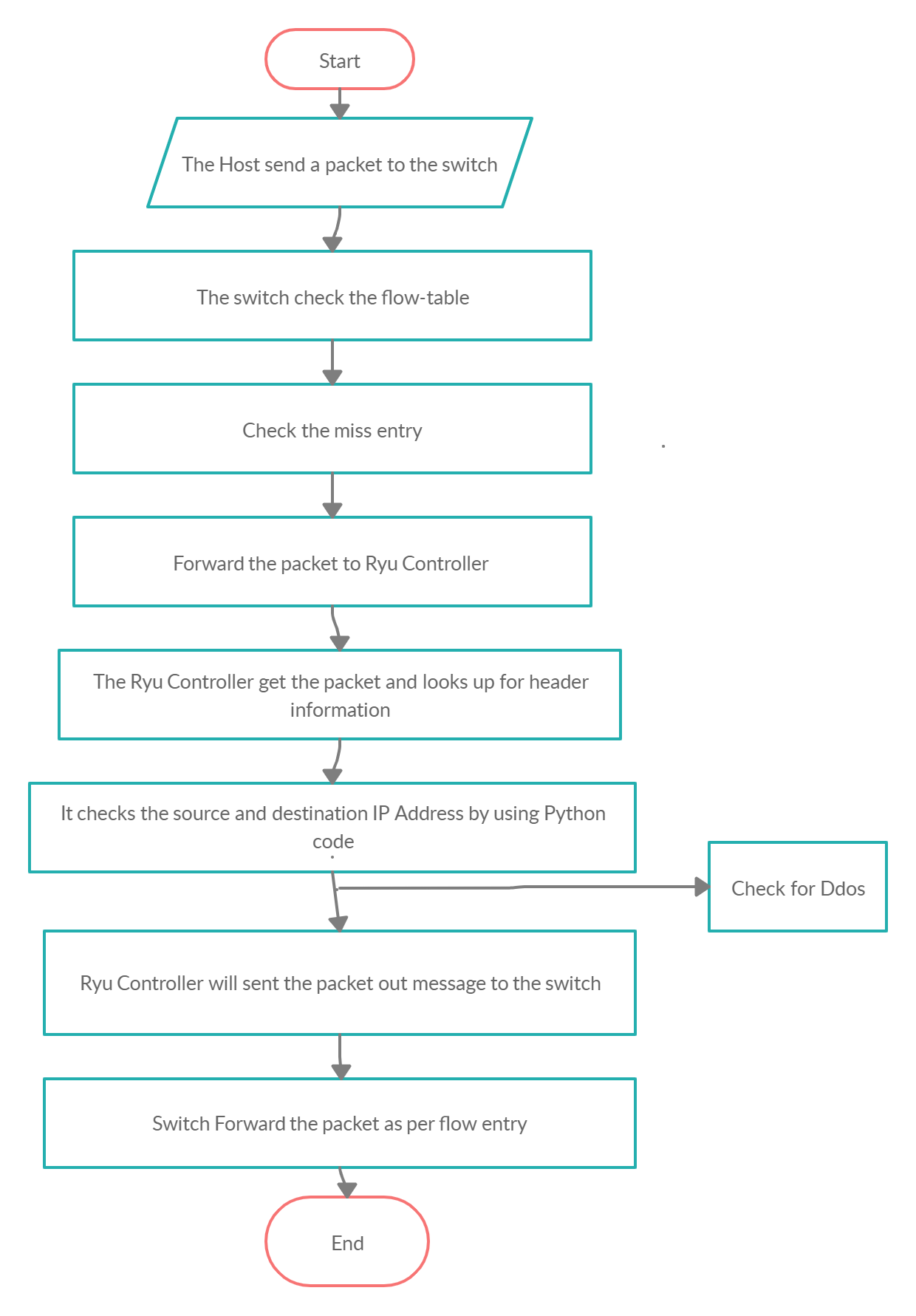
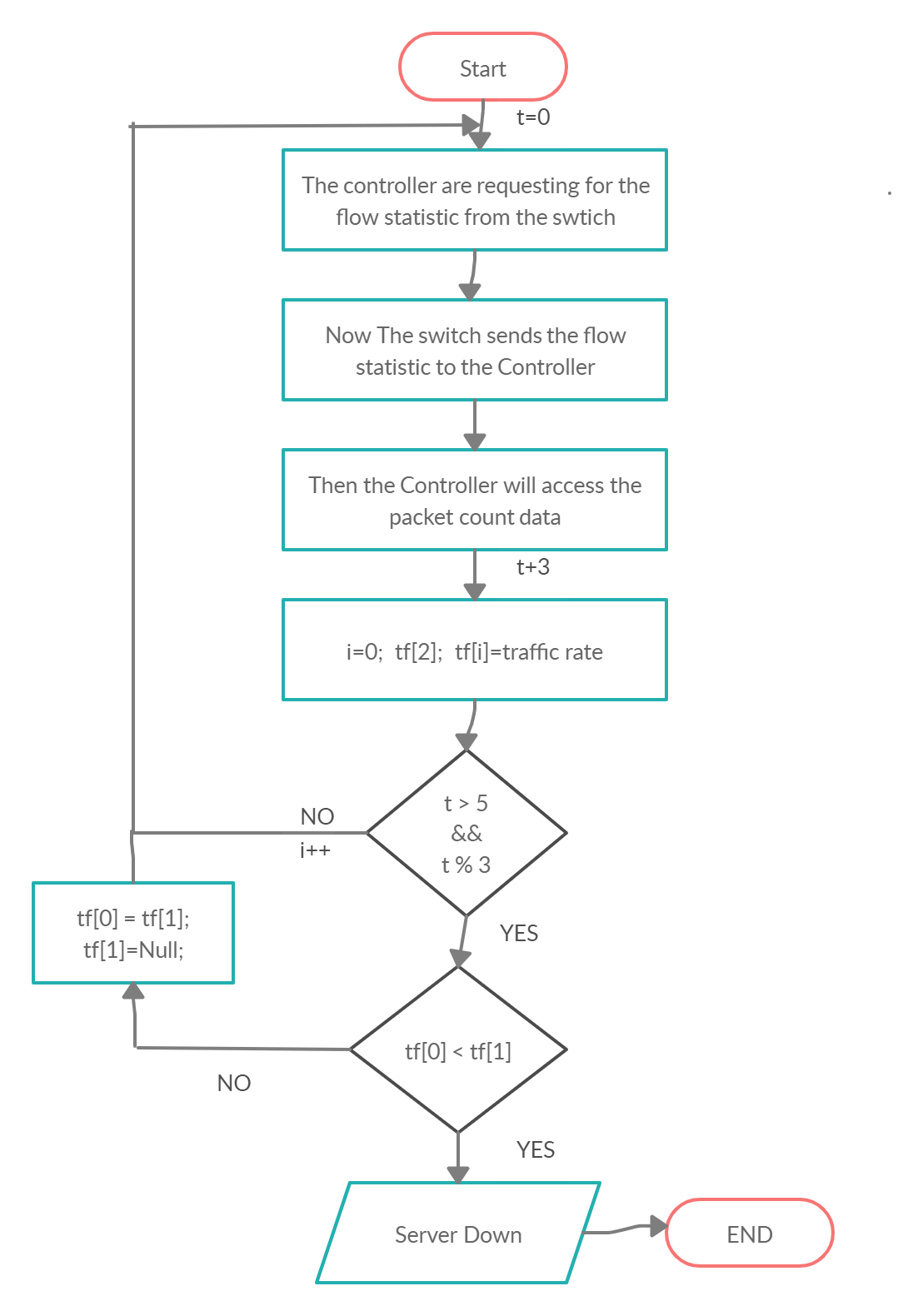


Figure 4.2: Ddos attack Detection Algorithm

**4.1 Ddos Mitigation by Ryu Controller:**



**Step 1**: Here, we are requesting the flow statistic from the switch. Now the switch sends the flow stats .from the flow stats, we will access the package count data. After 3 secs, we again request the flow statistic from the switch.

**Step 2**: Now using the package count data of consecutive request, we will find a traffic rate of a particular flow. IF the traffic rate is greater than desired rate, we will add a flow to drop the packets using that flow. When another package comes from the attacker, it will get drop.

**4.2 Problems**

* If the attacker has already attacked the host?
* If the attacker uses the USB port to pass data to the host or receive data?
* According to this algorithm, when the packet passes,the Ryu controller will just check the host’s IP address and destination address.

Then what will be the solution now?

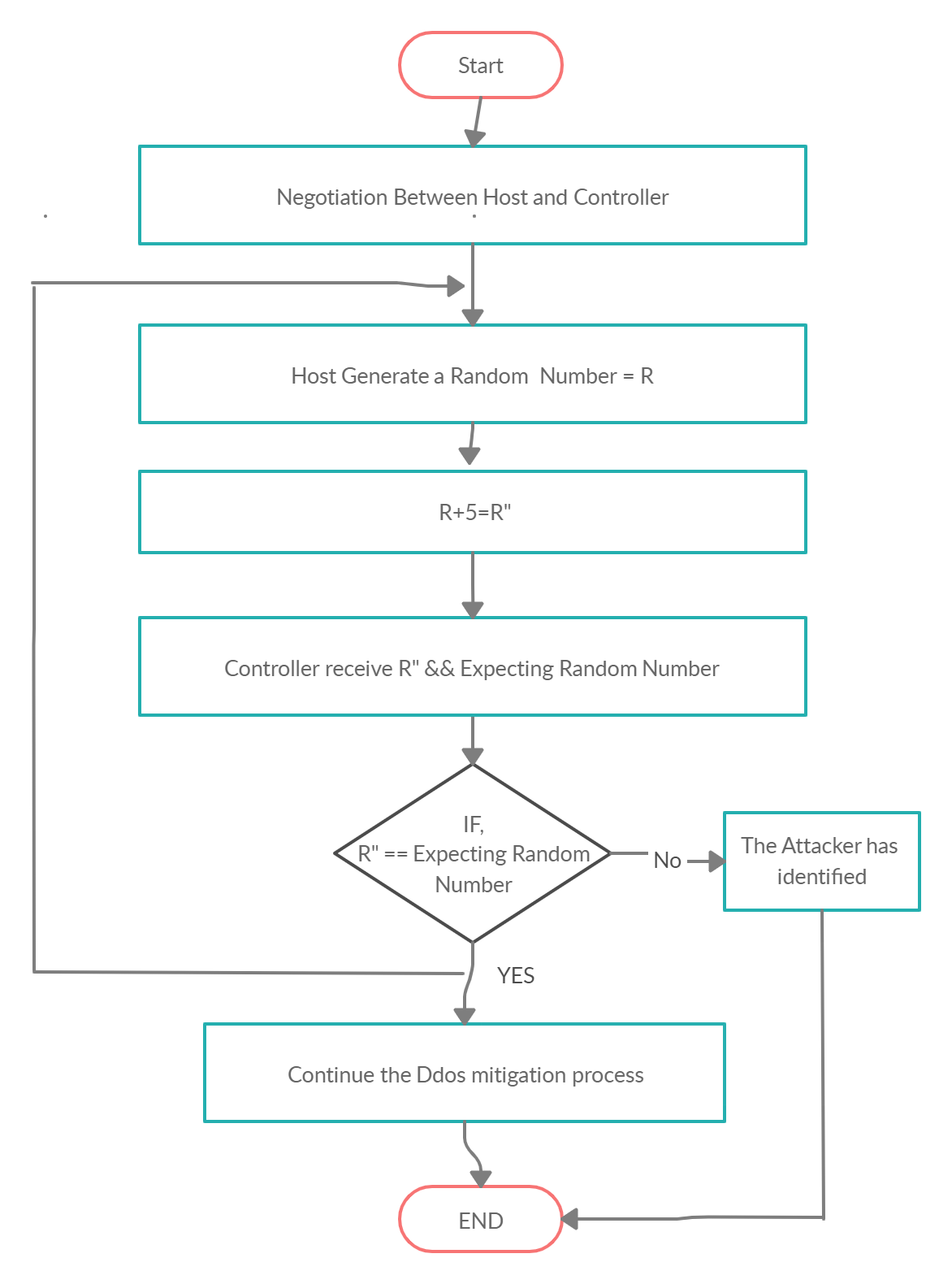
**4.3 Experimental Setup** Here, we can use the flag concept to solve this Problem

Figure 4.2: Flag concept flowchart

At first, Host and controller set a negotiation frame with expecting Random number for their communication. When host wants to send a data, it will send a Random generate number R for every packets and before the host & controller network is added, tell them in the negotiation frame how many numbers the host will increase by its random number. Then, for each packet, a random number will be generated and it will be incremented with a specific number which will be the controller's expecting number. Now if for some reason the controller does not get its expecting number, then the controller will immediately lower the server and the attacker will be identified. And once the controller gets its expecting number, the previous process will continue.

**Chapter 3**

**Specification of SND**

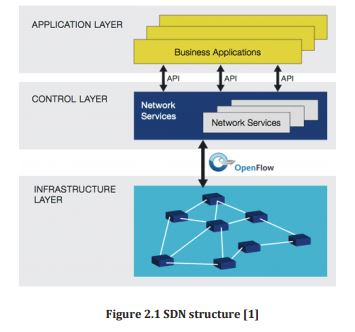
This chapter will first cover the background of Software Defined Networks, the Open flow protocol and its specification. And it will cover DDoS and its detection techniques in SDN networks Ryu Controller. Finally, the literature related to SDN security will be reviewed.

**2.1 Software Defined Networks**

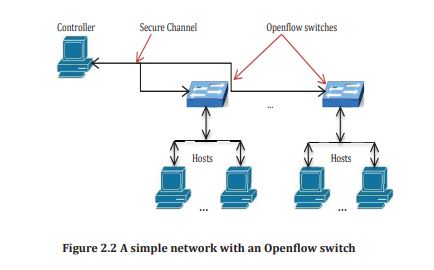
Software Defined Network is another way to look at networks. The main purpose is to improve ownership over network infrastructure. All control and forwarding behavior are programmed by suppliers in the equipment in current production networks and are, often, proprietary applications. The SDN architecture divides control plane and forwarding plane and allows the control plane to be operated over by network administrators [1]. This distinction is achieved by changing the network so that the firewall provides routing commands instead of using its power to handle the incoming packets. The turn should include tables instructing forwarding with flows. Open flow is the specification for the architecture of the SDN. It system is made up of switches activated by Open flow, a controller and a stable channel between controller and switches. Figure 2.1 displays different SDN layout levels. Through the control layer, the application layer will have a single view of the network and the entire system would appear like a logical transfer. The control layer is where the device abstracts from the application layer the network architecture. All adjustments and changes can be rendered in real time by using the control layer. There is no need for every system to learn different protocols in the network framework, and the only function remaining is to forward. The Open Networking Foundation is the largest agency supporting SDN implementation, partnering with multiple vendors, and having various groups working on Open flow specification.

**2.2 Open-flow Protocol**

It is possible to consider the Open flow protocol as SDN's workhorse. This controls the network switches and enables an external person such as the dispatcher to monitor the network movement of packets. Open flow has been developed as a network research-focused tool [2].



However, several vendors have started offering their Open-flow switches in recent years [3]. All switches have tables that show a packet's paths of entry and egress for that switch. Open-flow uses this property and makes it possible for the controller to access these tables. An Open-flow transfer can obtain a secure channel with its flow table entries and exclusion from the device. A simple network is shown in Figure 2.2



When a new packet arrives at an Open-flow switch, a match can be found in the flow table. If no match is found in the table, the packet is sent to the controller.

The controller handles the packet and marks the packet with an action such as:

* Add a new incoming packet flow
* Drop similar packets
* Tag with a queue ID

**2.3 Open-flow Specifications**

Lookup, match, forward and request controller operation are all done on the basis of the Open flow specification provided by the Open Networking Foundation. Version 1.0[1] was used in this work. Version 1.0 was the first package suitable for networks of output. Version 1.3 specifications for the new features will be referenced from time to time. This section will cover, without dwelling in too many specifics, the main areas of the requirements relevant to this research topic.

**2.3.1 Open-flow Switch**

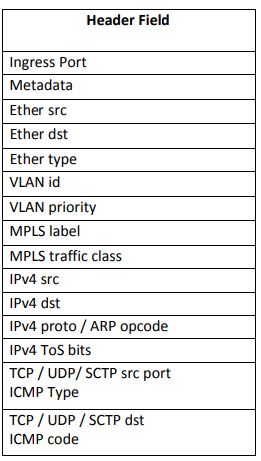
An Open-flow switch is made up of a flow table or a set of them and a stable controller interface. For each entry, each table has a match field, counters, and instructions set. The corresponding mechanism in the transfer occupies various fields in the header of the packets. The fields a switch can use to find a match in its tables are shown in Table 2.1. There is a metadata field in the table that is known as a mask able register (second row in Table 2.1) to bring information from one table to the other when there is more than one table. It is a means of transferring header information from table to table. Switches often have several pipeline tables.

For a match the packet moves from one table to another and bears the metadata. If a match is found, it will change the metadata attribute appropriately.

Any packets that reach the switch will be reviewed for all current table’s flows. If a match is found, it will perform the action allocated to that entry and refresh the entry counter. Counters cover a number of components per flow entry, table, port, queue and other areas on a switch like counters. Duration for example, refers to the time spent on a flow in the table. All the counters are wrapped without an excess. For different reasons, the controller can and will measure those counters. Not all counters are used because controllers are designed or modified to match the vendor's needs. The counters can be disabled in version 1.3.

If there may be no match, the packet will be despatched to the controller if the field isn't always matched, the packet may be dropped in the specifications model 1.3. Because the Header has no fields laid out in Table 2.1, it considers an invalid or illegal package. Their solution works with the table's IP address.

**Table 2.1 Packet header match fields**

****

The controller can send new packets as a whole or the switch can buffer the payload and send the header only. The default mode is the latter.

It will be encapsulated and labeled as an OFPT PACKET IN message when a packet is sent to the controller. We're going to call it Packet In. Considering the number of switches, time of day, packet length, priority and other factors, the controller has to process these packets and send a response to handle the packet and the packets that follow from the same source. This is where the controller will manage the loading entirely and only the forwarding will be done by the switch.

The controller sends a set of actions to the switch; forward, drop, push the queue, service quality and modify a field, that is, modify the VLAN tag, MAC address or IP address. The main actions are forward and down, and fields are available to queue and modify. The action will be configured for the packet in the controller and returned via Packet Out message to the switch.

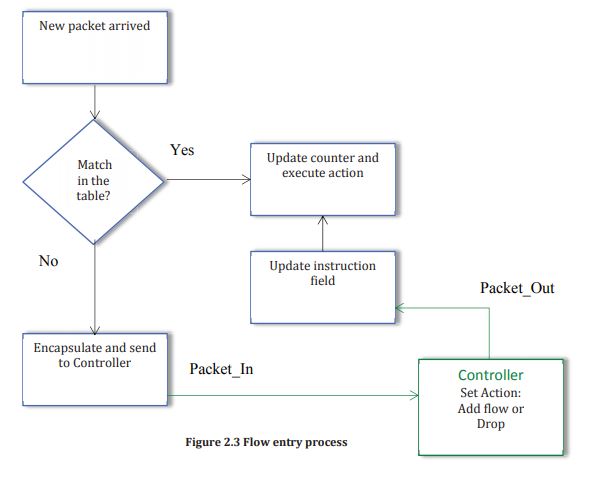


Figure 2.3 shows the flow entry process. If a packet is flagged with a drop action, a flow entry will be added. Any packet that matches the flow will be dropped. If the flow does not receive packets, the flow input will time out and be removed from the table after some time.

**2.3.2 Secure Channel**

A brief look at the specification for Open-flow reveals the single point of contact between the forwarding plane and the control plane. It's a secure channel between the controller and the switch. If the connection to the controller is lost, a pure open-flow switch will not manage the unknown incoming packets.

The safe channel for all Open flow switches in SDN is the life line. The relation between the controller and the switch is TLS or TCP. The switch attempts to connect with a backup controller if the connection is lost. When the connection is lost this is the "fail safe mode," which means that all packets are dropped from the controller.

On an Open-flow switch, the controller is not sent to all new packets processed by the switch. The Hybrid Switch is named if the switch will operate in SDN and in none SDN. The switch will not follow the Open-flow specification in this situation, and the network would lose its SDN design.

The Open-flow design shows that we have no central control or isolation of forwarding and control plane without a network device. The importance of every hazard, which can make the controller unattainable, is highlighted in this section.

**Chapter 4**

[**Research**](#page16) **Methodology**

**3.1 Distributed Denial of Service Attack and its Mitigation**

The DDoS attack is a known attempt by sending them heavy traffic to maliciously exhaust the resources of a computer or a network of computers [4 ]. The attacker's two main objectives are:

i) Resource exhaustion [5 ].

ii)Depletion of bandwidth.

The DDoS attack starts with an attacker planting malware on infected PCs called Botnet. Such codes are controlled at the time of the attack and traffic flows are directed at the victim. A more sophisticated assault uses a specialized layer of infected PCs known as handlers to monitor a greater number of PCs called zombies. Zum producing attack traffic is in charge of the zombie hosts [5]. The use of botnets concentrates the attack and hides the attacker behind the scene.

DDoS is one of the most common methods of network congestion and destruction. More than 7.000 these attacks are conducted each day by hackers, and data show that the total bandwidth in the first quarter of 2013 exceeded 48.25 Gbps, a 718% higher bandwidth than in 2012[6]. Significant daily attacks have taken place from May 2013 to September 2013. The Virtual Threat Map from Google picks up such attacks on its database[ 7]. It cannot be identified or reported all threats, but by analyzing the figures, each network is at imminent danger. The DDos Attack path is shown in Figure 3.1.

**3.2 Types of DDoS Attacks**

The computers in the victim's subnet must be reached as zombies to order to launch an attack.  Hackers are using scanning to identify compromised machines in the network. The testing may be random, based on a hit list, a local subnet search, or a hacker algorithm [8].

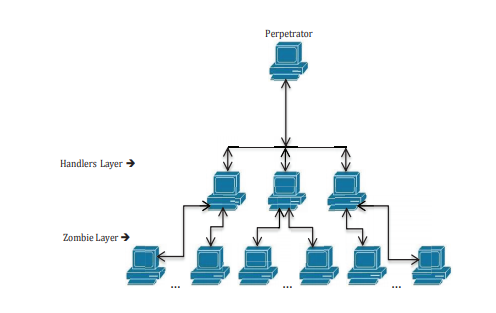


Figure 3.1: DDsos Attack Components

These attacks could be categorized into attacks on the client, host, server attacks, resource attacks, network attacks [8 ].

1. Host: making an inaccessible host. .
2. Client: aiming to deny legitimate use of an application on a host.
3. Application: attempting to refuse legal use of an application on a server.
4. Network: To maximize the bandwidth, send a large volume of traffic to a network
5. Infrastructure: attacking a domain name server at various locations at the same time.
6. Resource: to overload a server to keep it connected to the continuous stream of false requests.

In recent years, some types of DDoS were very popular. The sequence of attacks reveals that hackers used certain methods to launch attacks [6][7 ]. Many of these methods are next to being investigated.

A Web site was created by Google and Arbor networks that tracks attacks and gathers statistics from around the globe[7]. Statistics indicate the type of attack, length of attack, use of bandwidth, the source and destination mostly used ports, source of attack and the country of the victim.

1. UDP Flood is a type of attack that sends a number of packets to the victim's computer for random ports to make the system check for applications. The system will send unreachable packets for each incoming packet. The computer is finally unavailable when incoming packages are rising and the time interval increases.
2. SYN Flood is an assault that attacks the computer of the user using a TCP communication initialization. The victim is sent several SYN packets but no ACK is sent back to the victim which causes the resources to be used waiting for an attacker to confirm. With 38.7% of the attacks perpetrated, TCP Flood was the highest attack detected until mid2013 [9].
3. DNS Reflection Attack sends spooofed IPs and calls for a much larger response than the request and delivers it to the target. The attacker alters the source's IP address to the target's IP, which contributes to extreme traffic. Spoofing packets are common practice for DDoS attacks.
4. ICMP Flood is another type of attack which uses a large number of ICMP pings (echo requests) to exhaust victims ' resources and which keeps the server bound to send answers (echo responses).
5. HTTP Flood consists of sending a massive number of requests to a server and crippling it to the degree that valid requests cannot be answered.

High traffic in the victims and the loss of their resources is the common factor in all these attacks. Detection and mitigation of these heavy streams should help to better understand their effects on the SDN architecture within existing traditional networks.

**3.3 Anomaly Detection for DDoS Mitigation**

The common factor in various types of attacks by DDoS is the abnormal traffic sent to the target. There is a pattern and accepted consumption of bandwidth in ordinary circumstances in network activity. If traffic increases unexpectedly, delays, CPU use or the output of any network asset suddenly decreases, these are often considered to be anomalous. Any DDoS detection would scan the network for these anomalies. The existence of the data in the network[10] is generally related to anomalies. A bottleneck or application layer type causing the explosion of CPU resources can be guided to the network layer. DDoS's abnormal traffic to the target is the most common factor in different types of attacks. In normal circumstances there is a trend and agreed bandwidth usage in network operations. When there is an unexpected increase in traffic, delays, CPU usage or sudden reduction in the outcome of network assets, these are often deemed to be anomalous. Some DDoS monitoring will check for such irregularities in the network. Data are generally associated with irregularities in the network[10]. An excess of CPU power can be directed to the network layer through a bottleneck or the application layer.

**Chapter 5**

* 1. **Conclusion & Future Work:**

We've seen this before in a paper where a Ddos attack with the Ryu controller has been detected and mitigation algorithm implemented, but there are some lacks that we have tried to solve into this paper. This paper has discussed some of the SDN design security flaws and the various  possible threats in SDN. We provided a system that is easy to implement centered on a set intersection. we have generated an updated algorithm for detection and mitigation of Ddos ​​attack, where Before the host & controller network is added, tell them in the negotiation frame how many numbers the host will increase by its random number. Then, for each packet, a random number will be generated and it will be incremented with a specific number which will be the controller's expecting number. Now if for some reason the controller does not get its expecting number, then the controller will immediately lower the server and the attacker will be identified.

As a result the controller to identify the attacker at any time and host send the packet at any time and mitigate the Ddos ​​attacks. Only because of the negotiations between the host and the controller will there be a short time delay, which we will work on in the future with a better quality controller so that there is no time delay.

**REFERENCES**

|  |  |
| --- | --- |
| [1] | S. S. V. Y. M. F. M. T. G. G. P. Porras, "A Security Enforcement Kernel for OpenFlow Networks," *Proc. HotSDN’12,* August 13, 2012, Helsinki, Finland. |
| [2] | M. Dabbagh, B. Hamdaoui, M. Guizani, A. Rayes, "Software-defined networking security: pros and cons," *IEEE Communications Magazine Communications Standards Supplement ,* June 2015. |
| [3] | S. N. S. S. S. Scott-Hayward, "A Survey of Security in Software Defined Networks," *IEEE communication surveys & tutorials,* Vols. vol. 18, no. 1, no. first quarter 2016, DOI: 10.1109/COMST.2015.2453114 . |
| [4] | C. Qi, W. Jiangxing, H. Hongchao,C. Guozhen, "Dynamic-scheduling mechanism of controllers based on security policy in software defined network," *Electronics letters ,* vol. Vol. 52 No. 23, p. pp. 1918–1920 , 10th November 2016. |
| [5] | Q. Yan, FR. Yu, Q. Gong, 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 & tutorials,* vol. vol. 18, no. first quarter 2016, doi: 10.1109/COMST.2015.2487361, p. no. 1. |
| [6] | Danda B. Rawat, Swetha R. Reddy, "Software Defined Networking Architecture, Security and Energy Efficiency: A Survey," *IEEE Communications Surveys & Tutorials ,* vol. Vol 9 , no. Issue 1, DOI: 10.1109/COMST.2016.2618874 . |
| [7] | "SDN Central. (2013, Oct.) sdncentral. [Online]," [Online]. Available: http://www.sdncentral.com/announced-sdn-products/. |
| [8] | Ramachandra et.al., "Security Analysis of OpenDaylight, ONOS, Rosemary and Ryu SDN Controllers," no. TNSPS’16, DOI: 10.1109/NETWKS.2016.7751150 . |

[9] Open Networking Foundation. (2014, Jan.) ONF. [Online]. <https://www.opennetworking.org/>

[10] <https://www.youtube.com/watch?v=PFShXmWJn1E&t=2s>