



UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea

CYBER RANGE IN IOT

RAGHAV SINGHAL

Relatore: *Dr. Letterio Galletta*

Anno Accademico 2021-2022

INDICE

1. INTRODUCTION	11
1.1 Introduction.....	11
1.2 Structure	13
2. CYBER SECURITY TRAINING	14
2.1 Cyber Security Issues.....	14
2.2 Cyber Security Scenarios	15
3. CYBER RANGE.....	18
3.1 What is a Cyber Range?	18
3.2 Cyber Range Features.....	19
3.3 Types of Cyber Range Based on Virtual Technologies	19
3.3.1 Conventional Cyber Ranges	20
3.3.2 Cloud Based Cyber Ranges	20
3.3.3 Hybrid Cyber Ranges.....	20
3.3.4 Cost of Cyber Range	21
4. SOFTWARE DEFINED NETWORK (SDN)	22
4.1 What is Software-Defined networking?	22
4.2 Software Defined Network architecture.....	22
4.3 Three Main layers of SDN	24
4.3.1 Infrastructure layer or data plane.....	24
4.3.2 Control layer or Control Plane.....	25
4.3.3. Application Layer	25
4.3.4. Advantages of SDN [8]:	26
4.4 Open Flow Architecture	26
5. SECURITY ISSUES OF SDN.....	28
5.1 Attacks in SDN	28
5.2 DoS attack	30
5.2.1 Detection of DOS attack Wireshark Network Monitoring	30
5.2.2 How to prevent and Stop DoS attack:	30
6. MININET TOOL.....	32
6.1 Introduction to Mininet.....	32
6.2 Installation of Mininet.....	33
6.2.1 Installation of Pox controller.....	33
6.3 Characterstics of Mininet[10]	34
6.4 Basic commands with Mininet [9]	34

6.5 Creating a network with Mininet	35
6.6 Topology in Mininet.....	35
7. D-ITG DISTRIBUTED INTERNET TRAFFIC GENERATOR.....	38
7.1 Installation.....	38
7.2 D-ITG Software Architecture	39
7.3 D-ITG Components.....	39
8. EXPERIMENTAL RESULTS	42
8.1 D-ITG TRAFFIC GENERATOR IN MININET	42
8.2 Experimenting with SDN	47
8.2.1 Calculations for Single Topology	47
8.2.2 Calculations for Linear Topology.....	50
8.2.3 Calculations for Tree Topology	52
8.2.4 Comparing the Performance of Single, Linear and Tree Topology.....	54
8.3 Mobility in Mininet WIFI.....	56
8.3.1 Results and Discussions	57
8.4 Telemetry in Mininet-WIFI	59
8.4.1 Methodology.....	60
8.5 AR.Drone 2.0 Dos attack using CoppeliaSim	63
8.5.1 Theoretical background	64
8.5.2 Computer Security Principles	64
9. CONCLUSION	69
9.1 Future Work.....	69
10. REFERENCES.....	70
11. CODE USED IN THIS THESIS.....	72

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my thesis supervisor Dr. Letterio Galletta, Assistant Professor at IMT School for Advanced Studies Lucca. His patience, enthusiasm, cooperation, and suggestions help me complete this research work in the present form. His brilliant, skilful supervision enriched this study higher than my expectation. I am very obliged for his beneficial feedback and suggestions during the preparation of my research work

Further, yet importantly, sense of respect goes to my father Mr. Anupam Singhal, mother Mrs. Meenu Singhal and my family and friends who provided me continuous support and encouraged me throughout my years of study

Finally, I am thankful to the School of Mathematical, Physical and Natural Sciences of the University of Firenze, Italy for providing me the platform to conduct and execute this research work.

Raghav Singhal
Matriculation Number:7031789
raghav.singhal@stud.unifi.it
Master's Degree in Computer Science- Resilient and Secure Cyber
Physical Systems

ABSTRACT

The Internet Of Thing (IOT) is one of the pointer technologies in the digital transformation era. IOT is expected to jump forward in upcoming years, with nearly 50 billion devices/things connected to the Internet which was earlier 7 billion in 2008. IOT's capability in functioning as a network that is used to connect physical entities aka smart devices like machines, wearable devices, and autonomous cars. The security and the correctness of IOT, therefore, is paramount. Cyber ranges (CR) are interactive, simulated platforms and representations of networks, systems, tools, and applications. CR can provide performance-based assessment and learning, provide a simulated environment where different teams can work together to improve team capabilities and teamwork, they also provide a real time feedback and provide environment where latest ideas can be implemented and evaluated by teams so that they can solve problems.

The IOT cyber range allow designers and developers of such systems to specify and work on a modifiable IOT networks, both physical as well as a virtual, and supports the parallel execution of many scenarios in a scalable way.

This thesis considers some tools for simulating different kinds of network and use them to create and experiment with some attack scenarios in IOT. Our goal is to show that is relatively easy create scenarios and study in a synthetic way how attacks may carried out in IOT.

LIST OF FIGURES

Figure 1: The concept of Next Generation Cyber Range.....	19
Figure 2: Architecture Of SDN [7]	24
Figure 3: Architecture of Open Flow Switch.....	27
Figure 4: Creation Of Netwok In Mininet	35
Figure 5: simple topology	36
Figure 6: single topology of 4 host.....	36
Figure 7: Linear topology of 4 switch and 4 host.....	37
Figure 8: Tree Topology using Depth as 2	37
Figure 9: Results of receiver.log File.....	41
Figure 10: Starting of POX controller	42
Figure 11: - ./ITGRecv receiver side	43
Figure 12: - ./ITGSend	44
Figure 13: - sender.log file	45
Figure 14: receiver log file.	46
Figure 15: Maximum Delay for Single Topology	48
Figure 16: Average Jitter Of Single Topology	49
Figure 17: Average Bitrate of Single Topology.....	49
Figure 18: Maximum Delay of Linear Topology.....	50
Figure 19: Average Jitter of Linear Topology.....	51
Figure 20: Average Bitrate of linear Topology	52
Figure 21: Maximum Delay of Tree Topology	53
Figure 22: Average Jitter of Tree Topology.....	53
Figure 23: Average Bitrate of Tree Toplogy	54
Figure 24: Maximum Delay In POX controller	54
Figure 25: Average jitter in POX Controller	55
Figure 26: Initial state of Mobility.....	57
Figure 27: Final state of Mobility.....	58
Figure 28: Mobility after DDoS attack.....	59
Figure 29: Testing the bandwidth with tx_paxkets.....	60
Figure 30: DDoS attack on tx_packets.....	61
Figure 31: Separation of Graphs using rssi	61
Figure 32: DDoS attack on rssi.....	62
Figure 33: Wireshark as a monitoring tool	63
Figure 34: Reconnaissance using nmap.....	65
Figure 35: CoppeliaSim Scenario.....	66
Figure 36: DDoS attack on Ar. Drone.....	67
Figure 37: DDoS attack	68
Figure 38: Wireshark for confirmation	68

LIST OF TABLES

Table 1: Pros and Cons of Hybrid Cyber Range	20
Table 2: Results of simple mininet using D-ITG.....	46
Table 3: Maximum Delay for Single Topology	48
Table 4: Average Jitter for Single Topology	48
Table 5: Average Bitrate for Single Topology	49
Table 6: Maximum Delay for Linear Topology	50
Table 7: Average jitter for Linear Topology.....	51
Table 8: Average Bitrate for Linear Topology	51
Table 9: Maximum Delay for Tree Topology	52
Table 10: Average jitter for Tree Topology	53
Table 11: Average Bitrate for Tree Topology.....	54
Table 12: Average Bitrate using POX Controller	55

ACRONYMS

D-ITG	Distributed Internet Traffic Generator
IOT	Internet Of Things
MFA	Multi-Factor Authentication
CR	Cyber Range
DDOS	Distributed Denial Of Service Attack
DOS	Denial Of Service Attack
SDN	Software-Defined Networking
VM	Virtual Machine
CISA	Cybersecurity and Infrastructure Security Agency
SQL	Structured Query Language
MITM	Man In The Middle
IT	Internet Technology
ECSO	European Cyber Security Organization

“I think computer viruses should count as life. I think it says something about human nature that the only form of life we have created so far is purely destructive. We’ve created life in our own image.”

--Stephen Hawking

1. INTRODUCTION

1.1 Introduction

Cyber-attacks have been gaining in frequency and size recently. Hacker targets have changed, their goals are no longer large distribution centres or company servers. Government organizations, hospitals and critical infrastructure are increasingly being attacked. These threats need to be addressed [\[7\]](#). With the increasing number of cyber-attacks, an increasing number of security experts are needed. Previously information technology security was not so important, so security was not given such attention. However, it is becoming increasingly clear that security is currently one of the most important sectors in information technology and that security experts equipped with the knowledge and experience need to be trained as soon as possible to be able to face these new security threats.[\[15\]](#)

The rapidly growing IoT represents a new attack surface. Many IoT devices lack inherent security features, so it's essential to have experienced security operators work in their defence. So the training of such operators is important in order to prevent that a IOT system become the entry point and the target of cyber-attacks.

A cyber range is a platform that provides hands-on cybersecurity practice to teams of professionals. This approach guarantees that client infrastructure and data is never at risk as a result of cybersecurity training. Cyber range customers are government agencies and military, private sectors organisation and universities.

This document aims is to give an overview of cyber ranges, starting form their definition, features, their types of cyber ranges along with their usage in the case of IOT.

The cyber range for IoT must simulate the large number of devices and the distributed, perimeter-free environments in which they are deployed. The Mirai botnet is an example of an exploitation of the LAN Mistrust problem domain. The botnet was used in October 2016 to conduct a

Distributed Denial of Service (DDoS) attack against people's routers across the world and caused many of the most popular websites at the time unavailable. This attack served as a proof of concept (poc) that IoT devices could be utilized in wide-scale networking attacks and derivatives of the Mirai malware have been found since.

This thesis considers some tools for simulating different kinds of network and use them to create and experiment with some attack scenarios in IOT in a cyberrange style. Our goal is to show that is relatively easy create such scenarios and study in a syntetic way how attacks may be carried out in IOT.

We foucs our treatment on IoT network that are based on software-defined network (SDN), since it is an pop up technology and is a shortcut to the future of infrastructure in network engineering.

We simulate these networks using the simulator MININET and its extensions. MININET creates a realistic virtual network, running real kernel, switch and application code, on asingle machine (VM, cloud or naive), in seconds, with a single command.

In the first part of this thesis, I have combined a software defined network (SDN) with centralized management by providing network separation, control planes and data. Many cloud computing environments, as well as data centres and service providers, are using this feature as this is one of the most important one. Here I have used Mininet to manifest the relevancy of SDN for different scalability. I study the performance by using POX controller that are implemented in Python using Distributed Internet Traffic Generator(D-ITG) and Mininet. During this Thesis, I have used three network topologies, single, linear and tree. The parameters of performance are maximum delay, average bitrate and average jitter, there are other parameters also. Experimental results show that the linear topology using POX controller performs better as compared to single and tree topology using POX controller. For the second part, I have used the Mininet-WIFI, a fork of Mininet extended with WIFI stations and access points. In this part, I considered a mobility and a telemetry scenarios and simulate a DDOS attack on them. In the case of Mobility I created a scenario in which 3 mobile phones are trying to connect to a mobile tower and in between an attacker gain access to one of the mobile and attacks the other devices making the whole system crash. For the Telemetry scenario, I created 4 stations and then tested the bandwidth for the first and the last station after that performing the DDOS attack manually making all the stations show the same data. Finally, I considered a network of drones. To do that I used CoppeliaSim which is one of the most

versatile and powerful robot simulation platforms. I created a scenario of where an attacker performs a DDOS attack on a drone to make its speed slow down.

1.2 Structure

The Thesis is structure as follows:

1. Chapters from two to eight introduce the theoretical background required to understand main concepts and the implementation. In particular they cover the following topics:
 - Cyber Security Training and Cyber Range
 - Software-Defined Network (SDN)
 - Security issues of SDN
 - Mininet Tool
 - D-ITG Distributed Internet Traffic Generator
2. Chapter eight contains a detailed description of the implementation of Mininet on SDN along with the implementation of Mininet-WIFI using Internet of Things IOT scenarios like Mobility and Telemetry.
3. Chapter nine draws some conclusions and presents some future work.

2. CYBER SECURITY TRAINING

Cybersecurity is important because it protects all categories of data from theft and damage. This includes sensitive data, personally identifiable information, protected health information, personal information, intellectual property, and governmental and industry information systems. Hence this section presents them from that point of view.

Before illustrating the cyber-attacks in detail, I will give a brief description of cyber security in the Cyber-Physical World. Recent times have demonstrated how much the modern critical infrastructures (e.g., energy, essential services, people, and goods transportation) depend on the global communication networks. However, in the current Cyber-Physical World convergence, sophisticated attacks to the cyber layer can provoke severe damages to both physical structures and the operations of infrastructure affecting not only its functionality and safety but also triggering cascade effects in other systems because of the tight interdependence of the systems that characterises the modern society. Hence, critical infrastructure must integrate the current cyber-security approach based on risk avoidance with a broader perspective provided by the emerging cyber-resilience paradigm. Cyber resilience is aimed to absorb the consequences of these attacks and to recover the functionality quickly and safely through adaptation. Several high-level frameworks and conceptualisations have been proposed but a formal definition capable of translating cyber resilience into an operational tool for decision makers considering all aspects of such a multifaceted concept is still missing. [\[14\]](#)

2.1 Cyber Security Issues

A security issue refers to any poisonous attack that look forward to access data illegally, to distrupts digital operations or cause information damage. Cyber issues can arise from various sources, including corporate organisations, criminal organisations, terrorist groups, dissatisfied employees, and lone hackers.

In recent years, many cyber-attacks which were highly profiles had resulted in loss of sensitive data. For example, the 2017 Equifax breach

compromised the personal data of roughly 143 million consumers, including birth dates, addresses and Social Security numbers. In 2018, Marriott International disclosed that hackers accessed its servers and stole the data of roughly 500 million customers. In both instances, the cyber security threat was enabled by the organization's failure to implement, test and retest technical safeguards, such as encryption, authentication, and firewalls.

Cyber attackers can use a company's or an individual's sensitive data to steal information or gain access to their financial accounts, among other potentially damaging actions, which is why cyber security professionals are essential for keeping private data protected.

2.2 Cyber Security Scenarios

Professionals in cyber security should have deep knowledge as well as an understanding of these types of the cyber security issues mentioned below:

- Malware [\[16\]](#)
Malware is malicious software such as spyware, ransomware, viruses, and worms. Malware is activated when a user clicks on a malicious link or attachment, which leads to installing dangerous software. Cisco reports that malware, once activated, can:
 - Block access to key network components (ransomware);
 - Install additional harmful software;
 - Covertly obtain information by transmitting data from the hard drive (spyware);
 - Disrupt individual parts of the system, making it inoperable.
- Emotet [\[16\]](#)
The Cybersecurity and Infrastructure Security Agency (CISA) describes Emotet as “an advanced, modular banking Trojan that primarily functions as a downloader or dropper of other banking Trojans. Emotet continues to be among the most costly and destructive malware.”
- Denial of Service [\[16\]](#)
A denial of service (DoS) is a type of cyber-attack that floods a computer or network so it can't respond to any requests. A distributed DoS (DDoS) does the same thing, but the attack originates from a computer network. Cyber attackers often use a flood attack to disrupt the “handshake” process and carry out a DoS. Several other techniques may be used, and some cyber

attackers use the time that a network is disabled to launch other attacks. A botnet is a type of DDoS in which millions of systems can be infected with malware and controlled by a hacker. Botnets, sometimes called zombie systems, target, and overwhelm a target's processing capabilities. Botnets are in different geographic locations and hard to trace.

- **Man in the Middle [\[16\]](#)**
A man-in-the-middle (MITM) attack occurs when hackers insert themselves into a two-party communication. After interrupting the traffic, they can filter and steal data, according to Cisco. MITM attacks often occur when a visitor uses an unsecured public Wi-Fi network. Attackers insert themselves between the visitor and the network, and then use malware to install software and use data maliciously.
- **Phishing [\[16\]](#)**
Phishing attacks use fake communication, such as an email, to trick the receiver into opening it and carrying out the instructions inside, such as providing a credit card number. "The goal is to steal sensitive data like credit card and login information or to install malware on the victim's machine," Cisco reports.
- **SQL Injection [\[16\]](#)**
A Structured Query Language (SQL) injection is a type of cyber-attack that results from inserting malicious code into a server that uses SQL. When infected, the server releases information. Submitting the malicious code can be as simple as entering it into a vulnerable website search box.
- **Password Attacks [\[16\]](#)**
With the right password, a cyber attacker has access to a wealth of information. Social engineering is a type of password attack that Data Insider defines as "a strategy cyber attackers use that relies heavily on human interaction and often involves tricking people into breaking standard security practices." Other types of password attacks include accessing a password database or outright guessing.
- **Business E-mail Account Takeover**
This attack occurs when a malicious user gains access to a legitimate user's email account. For example, once an attacker

gains access to the credentials from a phishing email that was sent out to employees, the attacker will then have access to that user's email.[\[17\]](#). We can protect by following: -

- Multi-Factor Authentication (MFA). See previous descriptions of MFA.
- Only enable external (outside your network) email access for the specific countries in which your employees work.

3. CYBER RANGE

In this chapter, I will introduce Cyber Range, its features, the types of Cyber Range along with the cost of a Cyber Range. The goal of this chapter is to get detailed information on the Cyber Range.

3.1 What is a Cyber Range?

Cyber Ranges give a virtual and multipurpose environment for organisations so that they can check and reveal how they integrate people, technology, and processes to protect their strategic assets and information services. Using cyber ranges they can improve the security and performance of their cyber and information technology (IT) infrastructure as they can represent real world scenarios and threats in a virtual environment. Cyber ranges also present an opportunity to enhance organisational training capabilities. The environment of Cyber Range runs on an infrastructure which is virtual like servers and networks. These may vary from basic security information controls like IDSs (Intrusion Detection systems), end-point protection and Firewalls to more advanced machine learning and data analysis solutions.

The first definition of a next-generation cyber range was coined by the European Cyber Security Organization (ECSO), which defines a cyber range as: “a platform for the development, delivery, and use of interactive simulation environments. A simulation environment is a representation of an organisation’s ICT, OT, mobile and physical systems, applications, and infrastructures, including the simulation of attacks, users, and their activities and of any other Internet, public or third-party services which the simulated environment may depend upon. A cyber range includes a combination of core technologies for the realisation and use of the simulation environment and of additional components which are, in turn, desirable or required for achieving specific cyber range use cases” [\[1\]](#)

According to Ficco and Palmieri, cyber ranges can consist of physical infrastructure, be completely virtualised, or a hybrid between the two. The suitability of each option depends on the wider context on which the cyber range is being built.

In the image shown below we can see the concept of the next generation cyber ranges.

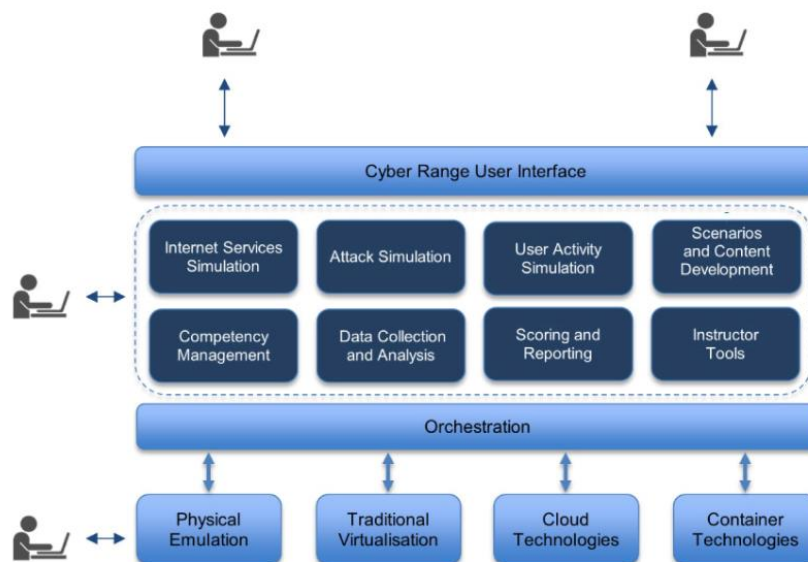


Figure 1: The concept of Next Generation Cyber Range

3.2 Cyber Range Features

Conventional training and education models are not sufficient to fulfil the skills of the cybersecurity gap. Cyber Ranges provide key enabling technologies to invoke, monitor and predict the performance and training done by professionals of cyber security. Cyber Ranges establish confidence in both the cybersecurity workforce aspirant and cybersecurity workforce employers that training will foresee job success. The critical features of cyber ranges are analysed in this section as catalysts in closing the cybersecurity workforce skills gap. Including: components which are technical, realistic and fidelity, usability & accessibility, elasticity & scalability, and learning & curriculum outcomes.

3.3 Types of Cyber Range Based on Virtual Technologies

Cyber Ranges can be divided into three types. And they are Conventional Cyber Ranges, Cloud Based Cyber Ranges and Hybrid Cyber Ranges.

3.3.1 Conventional Cyber Ranges

Conventional cyber ranges are based on traditional virtualisation, including conventional hypervisor base virtualisation and container technology. Cyber Ranges many times rely on one or other, or a combination of both. Cyber Ranges which are built on conventional virtualisation are uncharacterized by a strong level of symmetry and are usually associated to cyber range locations which are physical with huge virtual deployments. That is the reason conventional cyber ranges are not referred to as next generation cyber range.

3.3.2 Cloud Based Cyber Ranges

Cloud-based cyber ranges are based on cloud technology. Currently there are many types of cyber ranges which are like based on cloud. First one is Public Cloud Cyber ranges. These are developed on cloud providers which are public such as Google, AWS, MS Azure. These are strongly integrated with public cloud providers and are available only as cloud service without an on-premises option. Second one is Private Cloud Cyber Ranges. These ranges do not rely on public cloud providers instead they are developed on similar cloud technologies, thus giving the flexibility to be offered as deployed on premise or a cloud service.

3.3.3 Hybrid Cyber Ranges

A hybrid cyber range is a combination of any of the above cyber range types that we have already discussed. It is possible to create vast training resources by utilizing an array of different cyber range types, so to as fine-tune your training requirements. Hybrid range examples: The Virginia Cyber Range is a very well-known example that uses a wide variety of cyber range types. Another example is the European Future Internet Research & Experimentation, which was originally started in 2008. [\[2\]](#)

Table 1: Pros and Cons of Hybrid Cyber Range

S.No.	Pros Of Hybrid Cyber Ranges	Cons of Hybrid Cyber Ranges
1	You can pick and choose the elements that suit your environment the best	Maintaining a highly customized cyber range can be challenging
2	You can save money by selecting components that meet your budget	Making changes to an existing cyber range can be difficult if it is too uniquely customized

3.3.4 Cost of Cyber Range

Some financial institutions are working on designing a series of cyber ranges. “We’re looking at a couple of different cost models,” Brady said. The ranges cost up to \$500,000 to stand up, he said, but then each exercise must be planned and built— at a cost of \$50,000 to \$100,000 each. Those ongoing costs would be met through fees paid by exercise participants and by security vendors and customers who want to use the environment to evaluate their tools.

4. SOFTWARE DEFINED NETWORK (SDN)

Software-defined networking (SDN) technology is an approach to network management that enables dynamic, programmatically efficient network configuration to improve network performance and monitoring, making it more like cloud computing than traditional network management. In this chapter I will give a detailed description about Software Defined Networking along with its architecture. Then I will explain the main layers of SDN and at last OpenFlow architecture of SDN.

4.1 What is Software-Defined networking?

Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. The OpenFlow® protocol is a foundational element for building SDN solutions. For an in-depth understanding of SDN-based networking and use cases, check out the open-source micro-book, "Software-Defined Networks: A Systems Approach".[\[3\]](#)

SDN enables the network programming behaviour in a manner which is centrally controlled by a software application using APIs. By opening conventionally closed network platforms and implementing a common SDN controller, operators can manage the entire network and devices consistently, nevertheless of the complexity of underlying network technology.

4.2 Software Defined Network architecture

Software-Defined Network (SDN) is new networking, which permits centralized, programmable control logic planes and data plane

conceptual that can overcome the drawbacks of present network infrastructures [4]. SDN is designed to make a network flexible and is logically centralized through SDN controllers. SDN controller provides a centralized vision of the whole network [5]. The reason behind SDN is to keep the data forwarding plane separated from the control plane so that network operators and service providers can directly control and manage their own virtualized resources and networks beyond using hardware mechanisms. It is an application used to control packet forwarding of all the devices in the network and manages flow control for improved network management [6]. Manual configuration is reduced, for individual network devices because of the forwarding policies. In SDN, the control and data planes are distinguished which in turn allows control to be programmable and manageable such that control remains centralized and the data plane to be shortened and conceptual [7].

The motivation that SDN is necessary for the network operator and service providers are as follows [4]:

1. Network operators and service providers need to use SDN technologies to control and manage the network easily and efficiently.
2. The high complexity of operations and management in networks can be reduced with software-defined network rather than configured networks.
3. Network operators and service providers should provide an interaction method between the infrastructure layer and network layer so that services should be securely isolated from existing traffic.
4. The controller is aware of the necessities of the applications and resources available in the network infrastructure by using its north bound and south bound interface [4].
5. Application program interface (API) open to grow a benefit from network infrastructure in terms of services in application layer, devices in infrastructure layer and programmability in control layer.
6. Figure 2 shows the typical architecture of SDN. It consists of three open interfaces [7]
7. Northbound interfaces provide an interface between the control plane and the application layer. It is used to provide routing-related information, management related information

and policy-related information.

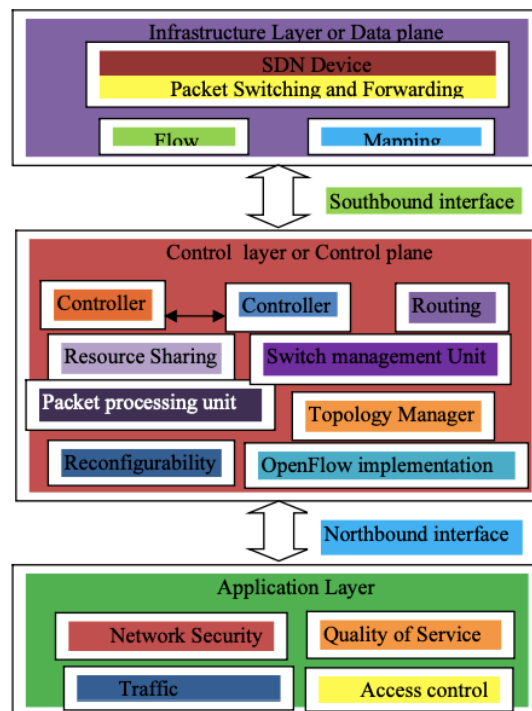


Figure 2: Architecture Of SDN [7]

4.3 Three Main layers of SDN

The three main layers of a SDN are the following:

- Infrastructure Layer or data plane
- Control Layer or control plane
- Application Layer

In the rest of this section we explain each of them.

4.3.1 Infrastructure layer or data plane

The infrastructure layer responsible for transferring data between SDN devices (both physical devices and virtual devices) used to perform packet switching and forwarding. Flow table is used to store the rule populated by the controller for controlling and directing the packets [8]. Mapping table is used to store the data transfer between the different network and SDN devices communicate with control plane through south bound interface.

4.3.2 Control layer or Control Plane

The Control layer is placed between the infrastructure layer and application layer of the SDN network where network services are specified [7]. This layer manages an overall view of the network. The Control plane is directly programmable in a centralized manner to provide hardware abstractions to SDN applications. It contains, a set of controllers that interact with network services and SDN devices through the northbound interface and southbound interface. Software that manages all the resources in network infrastructure is called Controller. The network infrastructure consists of packet switching and forwarding, a mapping table and a flow table that provide an abstract view of the overall network and the controllers that interact with each other through their east westbound interface to provide a stable view of the whole network infrastructure. Important functions/modules of a controller follows [7]:

- Packet processing unit: With respect to the protocol, it processes packet headers and their payloads.
- Switch management unit: Modification in switches and message arrival can be informed by the controller.
- Topology manager: it maintains up to date information about network topology and identifies changes in topology.
- Routing: by using forwarding table routing, the manager finds out the routes to reach the destination address based on protocol.
- OpenFlow implementation: the controller performs the function related to OpenFlow protocol such as action, table entry, flow rules, and message queues.
- Management interface: It provides access to functions that the controller provides
- SDN controller usually remains aware of all available network routes and can send packets based on different network characteristics are the important benefit.

4.3.3. Application Layer

It contains end-user applications that perform SDN communications and network services [4]. SDN services include access control management, network security, traffic engineering, load balancing, quality of service and other network function virtualization services.

The network flow of SDN devices in data plane is affected by communicating their necessities over northbound interface.

4.3.4. Advantages of SDN [\[8\]](#):

It allows a quicker response to modifying traffic (group of spoofed packets) and conditions.

It also supports more opportunities for dynamic provisioning, load balancing, monitoring specific traffic engineering, increasing the utilization of network resources, and improving better occasions to implement many different types of solutions.

4.4 Open Flow Architecture

SDN is implemented using the OpenFlow communication protocol which will access data from the data plane of a switch or router to the control plane through the network. Communication between the control plane and data plane is provided by this protocol. Packets can move with centralized decisions by using the Open Flow protocol [\[3\]](#). Switch operations remain in control of the OpenFlow controller. The action may be either Reactive or Proactive.

The reactive approach means that a switch will remain unaware of actions when a packet arrives. So, the packet is sent to SDN Controller [\[3\]](#). By using the OpenFlow protocol SDN controller is responsible for inserting a flow entry into the flow table of the switch. Switches totally dependent on the SDN controller and this is considered a major drawback of this approach.

The proactive approach overcomes the drawback of the reactive approach. Each entry in the flow table is prepopulated by flow entries of each switch in the case of the Proactive approach. It does not disrupt traffic, even if the switch loses connectivity with the control plane.

In OpenFlow architecture, we have a set of OpenFlow instructions or commands that are transmitted from an OpenFlow control plane to the open flow switch [\[3\]](#). An OpenFlow switch performs the following operations: 1) Depending on the packet header fields identify and categorize packets from an incoming port; packets can be processed in various ways by changing the header field. 2) Drop or push the packets to a respective egress port (outgoing port) or to the OpenFlow control plane.

Figure 2 describes the typical architecture of an OpenFlow switch. An OpenFlow switch consists of number of flow tables (organizing flows

in table), group table (collection of action to be performed) and secure channel [7]. Flow tables consist of flow entries which are forwarded. Each entry matches its correspondent flow and packets, and then provides the functions that need to be performed on the sent packets. These flow entries have some set of parameters: 1) the match fields used to match the incoming packets (it uses the information there in the packet header, ingress port (incoming port), and metadata); 2) counters, used to make up the statistical data for each flow, e.g., the count of received packets, amount and time limit for a particular flow; and 3) a certain group of commands, which apply when there is a match in table and specify how to manage the matching packets [6].

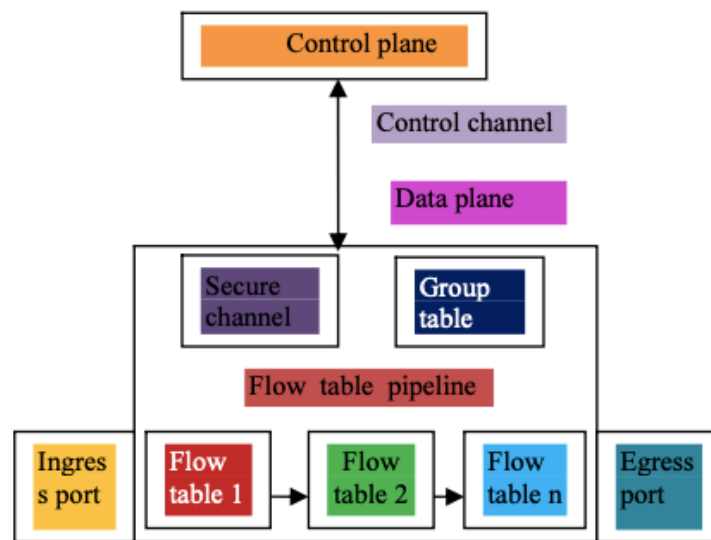


Figure 3: Architecture of Open Flow Switch

Group table consists of set of group entries. Each entry in group has certain specific semantics dependent in group type which consists of collection of action buckets. Action bucket state the action, which are performed in a group. The group will select one or more bucket for each packet. So SDN controller will manage all communications by using open flow.

5. SECURITY ISSUES OF SDN

In this chapter, I will discuss the security issues which are related to Software Defined Networking. After that, I will introduce the three layers of the attack. I also provide a detailed presentation of the Distributed Denial of Service attacks together with its detection and prevention.

5.1 Attacks in SDN

Software-defined networks are replacing traditional networking systems due to their centralized control approach. The privacy, integrity and confidentiality of the system may get affected due to the attacks on the system's vulnerabilities which ultimately reduce the performance and efficiency of the network [\[21\]](#). The security challenges to SDN include open programmable API in which the open nature of the APIs makes the vulnerabilities more visible to the attackers. Unauthorized access to the central controller may cause huge damage to the information and inject malicious codes into the system.

There are nine main issues related to SDN. For each issue a description is provided below:

1. **Forwarding Device Attack:** The network traffic can be disturbed by access points or switches, which are controlled by malicious users. These devices may launch denial of service (DoS) attacks that can result in network failure or disruption.
2. **Single Control Plane:** Due to the use of a central controller, any problem arising in the network results in the failure of the central controller. The approach to solve this problem is to use either horizontal or hierarchical controller distributions.
3. **Vulnerability of Communication Channel:** SDN southbound APIs such as Open Flow protocol uses TLS for communication of control data. However, TLS is often disabled administratively, and this makes the protocol prone to man-in-the-middle attacks and not suitable for implementation of secure channel.

4. **Fake Traffic Flows:** A non-malicious faulty device or an attacker can launch this or a DoS attack to dissipate the resources in forwarding devices or controllers.
5. **Authenticity:** It refers to the property that entities in networks are actually the ones they claim to be. The issue of authenticity for forwarding devices in SDWN networks is similar to that in traditional networks; it can result as a hindrance in network performance [\[18\]](#), [\[19\]](#).
6. **Confidentiality:** it prevents the exposure of information to unauthorized users. If this property is not ensured, unauthorized users may access network information or data [\[20\]](#).
7. **Availability:** It ensures that authorized users can access data, devices, and services whenever they need them.
8. **Open Programmable API:** The open nature of API makes the vulnerabilities more transparent to attackers.
9. **Man-in-the-Middle-Monitors:** The switches and the controllers are not directly connected for the transmission of information. An attacker can mount a “man-in-the-middle” attack to monitor, steal or misuse the information without being caught thus leading to black hole attack.

More attacks on SDN consist of application-layer attacks, control layer attacks and infrastructure layer attacks. I briefly describe these attacks below:

- The Application layer attacks include two attacks: the rule insertion and the malicious code. Rules insertion: creating and implementing security rules for SDN in different domains lead to various conflicts. Malicious Code: injecting different programs lead to various attacks where attackers inject malicious code which leads to the corruption or loss of data.
- The Control layer includes the Denial of Service attack and the Attacks from applications. Denial of Service Attacks: these attacks can occur at channels, controllers or between the controller and the switches. Attacks from Applications: the attacker who gets illegal access from the application layer gets sensitive data about the network which leads to attacks against the control layer.
- The Infrastructure layer attacks include Dos attacks and Man-in-the-middle attacks. In the first attack, An attacker can down the buffer flow and the flow table by transmitting frequent large mysterious packets, which will generate new rules to be inserted into flow tables. In the man-in-the-middle Attack, The switches and controllers are not directly connected for the transfer of information so the “man-in-the-middle” monitors can intercept important

information without being detected and can result in eavesdropping and black hole attack. Security is analysed through attack graph and alert correlation model. Attack graph measures the ability to overcome the attacks whereas the alert correlation model classifies the alerts.

5.2 DoS attack

A denial-of-service (DoS) attack is a type of cyber-attack in which a malicious actor aims to render a computer or other device unavailable to its intended users by interrupting the device's normal functioning. DoS attacks typically function by overwhelming or flooding a targeted machine with requests until normal traffic is unable to be processed, resulting in denial-of service to additional users. A DoS attack is characterized by using a single computer to launch the attack. The primary focus of a DoS attack is to oversaturate the capacity of a targeted machine, resulting in denial-of-service to additional requests. The multiple attack vectors of DoS attacks can be grouped by their similarities. In this thesis, I used the hping3 to simulate a DoS attack in SDN. The tool hping3 allows you to send manipulated packets, where you control the size, quantity, and fragmentation of packets. You can use different strategies to overload the target and bypass or attack firewalls. Hping3 can be useful for security or capability testing purposes, using it you can test firewalls effectively and if a server can handle a big number of packets.

5.2.1 Detection of DOS attack Wireshark Network Monitoring

Wireshark is the world's foremost and widely used network protocol analyser. It allows inspecting the traffic that is flowing on your network at a microscopic level and is the de facto (and often de jure) standard.

5.2.2 How to prevent and Stop DoS attack:

Prevention is the best solution, for DDoS attacks. There are different methods from which we can protect the system from an attack. These methods are not fully definitive but they can work as tips to avert a devastating DDOS attack. The methods are below:

- Partner with the Right ISP and hosting power: if you do not host your own web service or application, then it is very important to choose the right ISP or hosting provider. The provider should provide you with adequate security best practices and a response plan for stopping a DDoS attack. Also,

a good ISP should have staff that is experienced in stopping DDOS attacks. So, if you have identified symptoms of a DDoS attack, you can simply call your ISP or hosting provider and ask for their help. Depending on the strength of the DDOS attack, they might have detected and stopped it before you.

- Protect the network perimeter by establishing a few technical measures to least partially mitigate the effect of any DDOS attack, especially on early Detection. A first mitigation is to aggressively time out half-open connections whenever possible, drop malformed and spoofed packages as early as possible, set Rate limit to routers to prevent volumetric DDoS attacks, set lower thresholds for SYN, ICMP and UDP flood, establish a botnet detection system to detect botnet activities as early as possible.
- Increase the bandwidth: one of the aspects of protecting and stopping DDOS attacks is about having more bandwidths.
- Develop a DDOS response Plan: as discussed, when a DDOS attack has been identified, it could be too late. Therefore, the best way to stop a DDOS attack is to create a detailed response plan that comprehensively lists the required, pre-planned response steps when an attack is detected. This plan should include whom to call (ISP provider, DDoS mitigation services, etc), what steps should be taken by each member of IT and security teams and will you need to communicate it to your customers, vendors, and third-party stakeholders.

DDOS attacks can be exceedingly difficult to detect and stop, yet the potential damage can be devastating. Developing a DDOS response plan can be crucial as well as investing in the necessary security infrastructure like bot mitigation software, ISP providers with large enough bandwidth and others.

6. MININET TOOL

Now we got the knowledge about cyber security, cyber ranges, Software Defined Networking and Distributed Denial of Service attacks. In this chapter, I will introduce Mininet the tool which is used for creating virtual networks. I will also discuss Mininet characteristics, installation, and basic commands along with a scenario for creating a simple Mininet topology.

6.1 Introduction to Mininet

Mininet is a very popular tool used to create a virtual network on a single machine. It runs a collection of end-hosts, switches, routers, and links on a single Linux kernel. It uses lightweight virtualization to make a single system look like a complete network, running the same kernel, system, and user code. Running many network devices implemented through a network is very costly and difficult to implement. Mininet can be used to reduce these problems. It utilizes virtualization for the purpose of simulating a real network by decoupling data forwarding plane from the control plane. Mininet simplifies developing, sharing, and experimenting with OpenFlow and Software-Defined Networking systems.

Mininet provides the following SDN controllers:

1. Floodlight Controller: it is a very popular Open SDN Controller. It works well with physical- and virtual- switches that use the OpenFlow protocol. It is Java-based but provides a wide range of APIs to work with.
2. Pox Controller: POX is a networking software platform written in Python. POX started life as an OpenFlow controller, but can now also function as an OpenFlow switch, and can be useful for writing networking software in general. It currently supports OpenFlow 1.0 and includes special support for the Open vSwitch/Nicira extensions.
3. RYU Controller: Ryu is a component-based software-defined

networking framework. Ryu provides software components with a well-defined API that makes it easy for developers to create new network management and control applications. Ryu supports various protocols for managing network devices.

6.2 Installation of Mininet

There are two ways of installing the Mininet one is to directly download the virtual image from the mininet website i.e. <http://mininet.org/download/>. And then install that virtual image in Virtual box. After that directly run it in your system. Second method is the naive solution from the source.

To install natively from source, first you need to get the source code: “git clone <https://github.com/mininet/mininet>”. Note that the above git command will check out the latest and greatest Mininet (which we recommend!) If you want to run the last tagged/released version of Mininet - or any other version - you may check that version out explicitly:

Now use the following commands: -

```
cd mininet
git tag
git checkout -b mininet-2.3.0 2.3.0
cd ..
```

Now we have the source tree. We can now install mininet By using command: - mininet/util/install.sh

Once the installation is over we can directly use mininet by using :
- sudo mn command

6.2.1 Installation of Pox controller

POX requires Python 2.7. In practice, it also mostly runs with Python. Install pox: - The best way to work with POX is as a git repository. You can also grab it as a tarball or zipball, but source control is generally a good thing:- USING the command as : - “git clone

<https://github.com/noxrepo/pox>”
cd pox

Running pox is ~/pox\$ git checkout dart This command is to use the dart branch.

6.3 Characteristics of Mininet[\[10\]](#)

The main features on Mininet follows:

- 1 Flexibility: Mininet is able to create new topologies and new features in software, with the help of programming languages and common operating systems.
- 2 Applicability: Mininet can simulate several network scenarios.
- 3 Interactivity: the simulated network can be deployed in the real network.
- 4 Scalability: large networks can be scaled with the help of the prototyping environment using hundreds or thousands of switches on only a computer.
- 5 Realistic: the prototype behaviour represents real time behaviour with a high degree of confidence, so code modification should not be required for using applications and protocols stacks
- 6 Shareable: The other collaborators are able to reuse the created prototypes, which can then run and modify the experiments.

6.4 Basic commands with Mininet [\[9\]](#)

Only a single console is required to control and manage entire virtual networks. In our scenario we use the basic commands such as ping, pingall, pingallfull, dump, net and iperf. Below, we provide an explanation of these commands:

- Ping: check the connectivity between different hosts by using ICMP protocol.
- Pingall: check connectivity between all hosts and tell which hosts are connected to each other.
- Pingallfull: similar to pingall but it gives more detail about how the hosts are connected. It also tells minimum, maximum and average time between two hosts in millisecond.
- Dump: display the IP address and process identification of the host.
- Net: list out links available in network between interface, host and switch.
- Iperf: test bandwidth between hosts in TCP connection.
- Sudo mn: It is used to start the mininet. We can also add the switches and topology along with the IP address.

6.5 Creating a network with Mininet

In this subsection we show how to create a network with Mininet. We can create a network with a single command. For example, the following command

“`sudo mn -switch ovs -controller rsf -topo tree,depth=2,fanout=8 -test pingall`”.

starts a network with a tree topology of depth two and fanout 8 (i.e., 64 hosts connected to 9 switches). The created network uses Open vSwitch switches under the control of the OpenFlow/Stanford reference controller, and runs the pingall test to check connectivity between every pair of nodes. (This takes about 100.781 seconds on my laptop.) As shown, in the image below: -

```
h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60
h62 h63 h64
h62 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h
21 h22 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40
h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60
h61 h63 h64
h63 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h
21 h22 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40
h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60
h61 h62 h64
h64 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h
21 h22 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40
h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60
h61 h62 h63
*** Results: 0% dropped (4032/4032 received)
*** Stopping 1 controllers
c0
*** Stopping 72 links
.....
*** Stopping 9 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9
*** Stopping 64 hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22
h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42
h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h6
2 h63 h64
*** Done
completed in 100.781 seconds
root@raghav-VirtualBox:~#
```

Figure 4: Creation Of Network In Mininet

6.6 Topology in Mininet

Here are some of the commands that can create different topologies in Mininet.

The command

`sudo mn`

It is used to start as base Mininet which consists of 2 host 1 switch and one controller. The network made by Mininet is shown below:

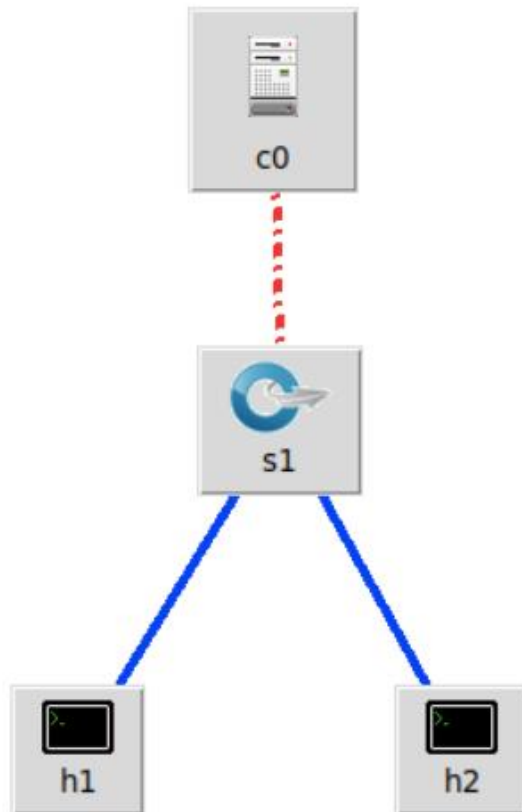


Figure 5: simple topology

`sudo mn - -topo=single,4:`

The resulting network consists of 4 hosts, 1 switch and one controller. It would look like the image below:

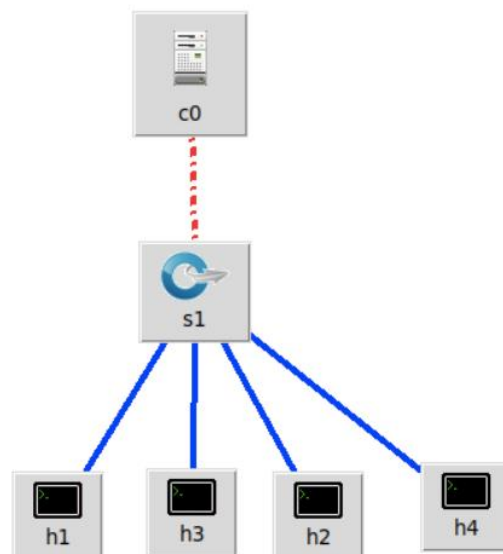


Figure 6: single topology of 4 host

Instead, a network created with the following command:

`sudo mn -topo=linear,4:`

consists of 4 switch, 4 host and one controller, it looks like:

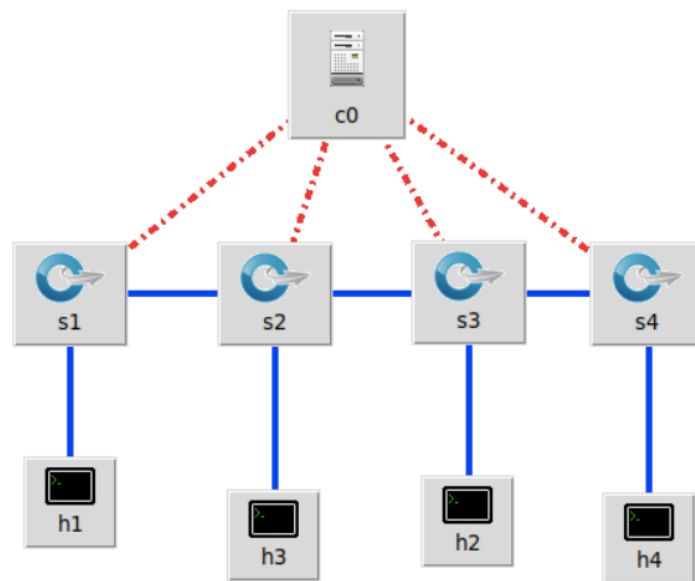


Figure 7: Linear topology of 4 switch and 4 host

Finally, the command
`sudo mn - -topo=tree,2,2:`
creates a network that consists of 4 host, 3 switches and 1 controller connect to each other in a tree like structure.

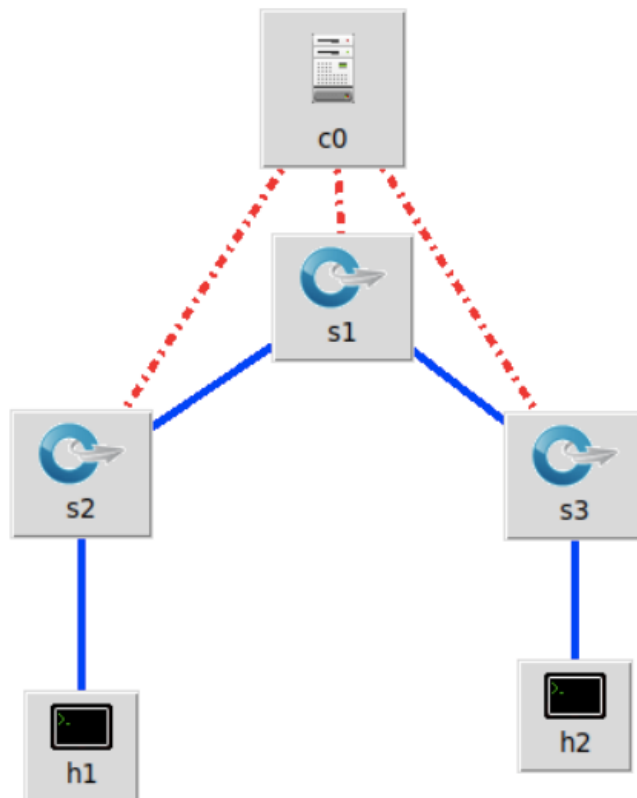


Figure 8: Tree Topology using Depth as 2

7. D-ITG DISTRIBUTED INTERNET TRAFFIC GENERATOR

As if now, we know how to use the Mininet tool. Now let's integrate it with a traffic generator named as D-ITG which means Distributed Internet Traffic Generator. In this chapter I will give a detail description about D-ITG tool along with its installation, architecture and its components used to generate traffic.

Distributed Internet Traffic Generator (D-ITG) is a platform capable to produce traffic that accurately adheres to patterns defined by the inter departure time between packets (IDT) and the packet size (PS) stochastic processes. Such processes are implemented as an i.i.d. sequence of random variables. A rich variety of probability distributions is available: constant, uniform, exponential, Pareto, Cauchy, normal, Poisson and gamma. Also, D-ITG embeds some models proposed to emulate sources of various protocols: TCP, UDP, ICMP, DNS, Telnet and VoIP (G.711, G.723, G.729, Voice Activity Detection, Compressed RTP). This means that the user simply chooses one of the supported protocols and the distribution of both IDT and PS will be automatically set. D-ITG can perform both one-way-delay (OWD) and round-trip-time (RTT) measurement, packet loss evaluation, jitter, and throughput measurement. For each generation experiment it is possible to set a seed for the random variables involved. This option gives the possibility to repeat many times the same traffic pattern by using the same seed. Also, D-ITG permits the setting of TOS (DS) and TTL packet fields.[\[11\]](#) D-ITG allows to store information both on the receiver side and the sender side.

7.1 Installation

D-ITG can be installed by two procedure one is to get the image from GitHub and the other by getting the zip file through naive source. Getting the zip file from the naive source the commands are:

```
Login into Mininet VM.  
$ sudo apt-get install unzip  
$ sudo apt-get install g++  
$ wget http://traffic.comics.unina.it/software/ITG/codice/D-ITG-2.8.1-r1023-src.zip  
$ unzip D-ITG-2.8.1-r1023-src.zip  
$ cd D-ITG-2.8.1-r1023/src  
$ make
```

this will install the D-ITG files and then we can use the tool

Cloning the files from GitHub the commands used are as follows:-

```
git clone https://github.com/davidfdzp/ditg.git  
cd ditg  
cd src  
make
```

7.2 D-ITG Software Architecture

D-ITG platform exhibits a distributed multi-component architecture. The communication between sender and receiver is done by using a separate signalling channel and ruled by a protocol for the configuration of the experiment (Traffic Specification Protocol - TSP).

7.3 D-ITG Components

A short description of the components of D-ITG architecture follows:-

ITGSend

Sender Component of the D-ITG Platform. The script mode enables ITGSend to simultaneously generate several flows. Each flow is managed by a single thread, with a separate thread acting as a master and coordinating the other threads. To generate *n* flows, the script file must contain *n* lines, each of which is used to specify the characteristics of one flow. The command which I used is “. /ITGSend -T UDP -a 10.0.0.2 -c 100 -C 10 -t 15000 -l sender.log -x receiver.log”. The detailed description is as follows:

- -T {protocol type} this means to Set the protocol type. Valid values are UDP, TCP, ICMP, SCTP, DCCP. Default is UDP. If you choose ICMP you must specify the type of message. Root privileges are needed under Linux.
- -a {destination address} this set the destination address by default is local host
- -c {pkts size} is the constant payload size.

- -t {duration} is used to set the generation duration. It is expressed in milliseconds. Default is 10000ms.
- -l {log file} Set the protocol type. Valid values are UDP, TCP, ICMP, SCTP, DCCP. Default is UDP. If you choose ICMP you must specify the type of message. Root privileges are needed under Linux.
- -x {receiver log file} Generate the log file at the receiver side. The default log file name is /tmp/ITGRecv.log under Linux and ITGRecv.log under Windows.

ITGRecv

Receiver Component of the D-ITG Platform. It can simultaneously receive flows from different senders. ITGRecv always works as a concurrent daemon, listening for new TSP connections. When a TSP connection request arrives, ITGRecv generates a new thread that is responsible for the management of the communication with the sender. Each single flow is received by a separate thread. Like ITGSend, ITGRecv can store information either locally or remotely by using the log server ITGLog.[\[11\]](#)

ITGLog

ITGLog is a “log server”, running on a different host than ITGSend and ITGRecv, which receives and stores the log information from multiple senders and receivers. The logging activities is handled using a signaling protocol. This protocol allows each sender/receiver to register on, and to leave, the log server. The log information can be sent using either a reliable channel (TCP) or an unreliable channel (UDP).[\[11\]](#)

ITGDec

The ITGDec decoder is the utility to analyze the results of the experiments conducted by using the D-ITG generation platform. ITGDec parses the log files generated by ITGSend and ITGRecv and calculates the average values of bi- trate, delay and jitter either on the whole duration of the experiment or on variable-sized time intervals. You can analyze the binary log file only on the operating system used to create that file. You can use another operating system if the log file is in text format. The Total time of the experiment is calculated as the difference between receiving time of last and first packet. [\[12\]](#). By using ./ITGDec sender.log we get the results of sender and ./ITGDec receiver.log we get the results on receiver side. The total Results shown contains: - Number of flows, Total time, Total packets, Minimum delay, maximum delay, average delay, average jitter, delay standard deviation, Bytes received, Average Bitrate, Average packet rate, Packets dropped, Average loss-burst size and Error lines. As shown in the Figure 9 below.


```

***** TOTAL RESULTS *****
-----
Number of flows      =      1
Total time           =    14.950627 s
Total packets        =    150
Minimum delay        =    0.000091 s
Maximum delay        =    0.003307 s
Average delay        =    0.000183 s
Average jitter       =    0.000091 s
Delay standard deviation = 0.000303 s
Bytes received       =    15000
Average bitrate      =    8.026419 Kbit/s
Average packet rate  =    10.033024 pkt/s
Packets dropped       =      0 (0.00 %)
Average loss-burst size =      0 pkt
Error lines          =      0
-----
root@raghav-VirtualBox:/home/raghav/DITG/ditg/bin#

```

Figure 9: Results of receiver.log File

ITGPlot

ITGplot is an Octave (<http://www.octave.org>) script that enables to draw plots starting from the data contained in delay.dat, bitrate.dat, jitter.dat or packet-loss.dat. The plot is saved (in the EPS format) in a file having the same name as the input file and the .eps extension. It is possible to save the plots in other formats by changing the graphicformat string in ITGplot. The available formats are those provided by gnuplot (run gnuplot and type 'set term' to see the list of the available terminal types). It is also possible to give a title to the plot by setting the environment variable DITG PLOT TITLE. [\[12\]](#)

ITGapi

ITGapi is a C++ API that enables to remotely control traffic generation. For this purpose, after having launched ITGSend in daemon mode (ITGSend - Q) on one or more traffic source nodes, it is possible to use the following function to remotely coordinate the traffic generation from different sender:

int DITGsend(char *sender, char *message); sender is the IP address of ITGSend and message is the string you would type at command line (except the name of the ITGSend executable file). Returns 0 in case of success, -1 otherwise. ITGSend, when used in daemon mode, sends messages back to the application that issued the generation of the traffic flow. Two types of messages are used, one to acknowledge the start of the generation process and the other to signal its end. The manager application can catch those messages by using the function:

int catchManagerMsg(char **senderIP, char **msg); [\[12\]](#)

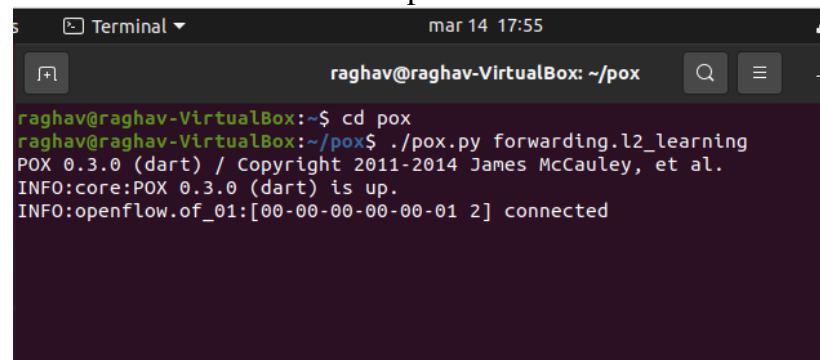
8. EXPERIMENTAL RESULTS

In this chapter, I present the experiments to simulate how a network reacts to a Dos attack. To do that I used traffic generator D-ITG. The purpose of doing this is to get the performance of the SDN controllers like the pox controller with a network simulation tool. First, I will do experiments by using single, linear and tree topology and then I will compare the maximum delay, average jitter, and average bitrate among those three topologies. In the second part of this chapter, I will integrate the Mininet with IoT by using the Mininet-WIFI which is an extended version of Mininet to use both WIFI stations and access points. I created two scenarios with Mininet-WIFI by using Mobility and Telemetry and then perform a DDOS attack on them to make the network stop. Finally, I used the CoppeliaSim tool to create a simulation of the drone. I created a scenario in which a drone is moving from one point to another and then an attacker attacks and make the speed of the drone slow. In this chapter I present all these experiments along with their code to simulate them with Mininet.

8.1 D-ITG TRAFFIC GENERATOR IN MININET

Here I show how to use D-ITG tool for generating traffic in Mininet. First of all, installation of the Mininet along with pox controller and D-ITG tool is mandatory. Inside the pox controller directory, then I start the controller with the forwarding algorithm with the command `./pox.py forwarding.l2_learning`

This command will start the pox controller as shown in the image below:

A terminal window titled 'Terminal' with a timestamp of 'mar 14 17:55'. The user is logged in as 'raghav' on a 'raghav-VirtualBox' machine, with the current directory being '~/pox'. The terminal shows the following commands and output:

```
raghav@raghav-VirtualBox:~$ cd pox
raghav@raghav-VirtualBox:~/pox$ ./pox.py forwarding.l2_learning
POX 0.3.0 (dart) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.3.0 (dart) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
```

Figure 10: Starting of POX controller

Then I start the Mininet by using the command
`sudo mn --controller=remote,ip=127.0.0.1,port=6633.`

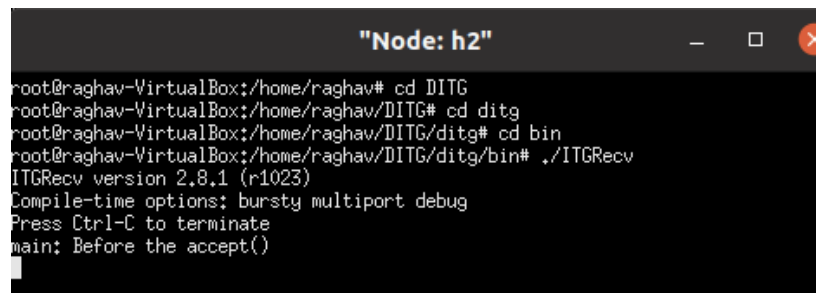
This command will connect to the pox controller with mentioned IP address along with port number of the pox controller. The topology will add 2 host as h1 and h2 along with s1 switch and c0 as the controller.

Then I use xterm command for using the terminal window of the host. Moew precisely, the Command is

`xterm h1 h2`

Now I kept h2 host as the receiver and h1 as the sender. So, on the receiver I use the following commands to receive the traffic:

- `Cd DITG/ditg/bin`
- `./ITGRecv` as shown in the image below:
-



```
root@raghav-VirtualBox:/home/raghav# cd DITG
root@raghav-VirtualBox:/home/raghav/DITG# cd ditg
root@raghav-VirtualBox:/home/raghav/DITG/ditg# cd bin
root@raghav-VirtualBox:/home/raghav/DITG/ditg/bin# ./ITGRecv
ITGRecv version 2.8.1 (r1023)
Compile-time options: bursty multiport debug
Press Ctrl-C to terminate
main: Before the accept()
```

Figure 11: - ./ITGRecv receiver side

The sender Host 1 will send the traffic to receiver side. The commands I used are same to get in the bin folder of DITG tool. Now I have to use the ITGSend function of the DITG tool to send UDP packets to the receiver side with the following command:

`./ITGSend -T UDP -a 10.0.0.2 -c 100 -C 10 -t 15000 -l sender.log -x receiver.log`

The command sends the UDP packets to host 2 with the IP address as 10.0.0.2 and the log files are generated on both the sides by the name as receiver.log and sender.log. The image of the command starting is attached below:

```
Firefox Web Browser "Node: h1"
root@raghav-VirtualBox:/home/raghav# cd DITG
root@raghav-VirtualBox:/home/raghav/DITG# cd ditg/bin
root@raghav-VirtualBox:/home/raghav/DITG/ditg/bin# ./ITGSend -T UDP -a 10.0.0.2
-c 100 -C 10 -t 15000 -l sender.log -x reciver.log
*ITGSend version 2.8.1 (r1023)
*Compile-time options: bursty multiport debug
main: SignalingTime Time: 1647275722.940710 sec
*modeCommandLine: mode started
flowParser: started
flowParser: TOKEN 0: -T
flowParser: TOKEN 1: UDP
flowParser: TOKEN 2: -a
flowParser: TOKEN 3: 10.0.0.2
flowParser: TOKEN 4: -c
flowParser: TOKEN 5: 100
flowParser: TOKEN 6: -C
flowParser: TOKEN 7: 10
flowParser: TOKEN 8: -t
flowParser: TOKEN 9: 15000
flowParser: TOKEN 10: -l
flowParser: TOKEN 11: sender.log
flowParser: TOKEN 12: -x
flowParser: TOKEN 13: reciver.log
flowParser: Parsing option T
flowParser: Level 4 Protocol: UDP
flowParser: Parsing option a
flowParser: Parsing option c
flowParser: Parsing option C
flowParser: Parsing option t
flowParser: Parsing option l
flowParser: Parsing option x
flowParser: Terminate Parser flow 1
identifySignalManager: flowId=1
identifySignalManager: Open a new signaling channel, chanId=4200296, flowId=1
createSignalChan: sent TSP_CONNECT
sendNameLog: sent TSP_SEND_NAME_LOG
signalManager: signalManager() started
signalManager: received msg code: 1
signalManager: perform a new request to send
```

Figure 12: - ./ITGSend

As shown in the Figure 12, ./ITGSend is sending UDP packets to the destination IP address and two files are getting generated by the name of sender and receiver.

Now to analyse the logs in host 1 I used the ITGDec command along with the sender log file. The sender results are shown in the figure below.

```
"Node: h1"
To 10.0.0.2:8999
-----
Total time           = 14.960540 s
Total packets        = 150
Minimum delay        = 0.000000 s
Maximum delay        = 0.000000 s
Average delay        = 0.000000 s
Average jitter       = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received       = 15000
Average bitrate      = 8.021101 Kbit/s
Average packet rate  = 10.026376 pkt/s
Packets dropped       = 0 (0.00 %)
Average loss-burst size = 0.000000 pkt
Packets duplicated    = 0
First sequence number = 1
Last sequence number  = 150
Loss Events           = 0
-----

***** TOTAL RESULTS *****
-----
Number of flows      = 1
Total time           = 14.960540 s
Total packets        = 150
Minimum delay        = 0.000000 s
Maximum delay        = 0.000000 s
Average delay        = 0.000000 s
Average jitter       = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received       = 15000
Average bitrate      = 8.021101 Kbit/s
Average packet rate  = 10.026376 pkt/s
Packets dropped       = 0 (0.00 %)
Average loss-burst size = 0 pkt
Error lines          = 0
-----
root@raghav-VirtualBox:/home/raghav/DITG/ditg/bin#
```

Figure 13: - sender.log file

On the receiver we will see the receiver log files for that first we need to stop the ./ITGRecv command by using “control + c” keys. Then use the ITGDec command below to get the receiver log file:

ITGDec receiver.log

the result is shown below:

```

From 10.0.0.1:46122
To 10.0.0.2:8999
-----
Total time           = 14.953157 s
Total packets        = 150
Minimum delay        = 0.000082 s
Maximum delay        = 0.007593 s
Average delay        = 0.000193 s
Average jitter       = 0.000098 s
Delay standard deviation = 0.000611 s
Bytes received       = 15000
Average bitrate      = 8.025061 Kbit/s
Average packet rate  = 10.031326 pkt/s
Packets dropped       = 0 (0.00 %)
Average loss-burst size = 0.000000 pkt
Packets duplicated    = 0
First sequence number = 1
Last sequence number  = 150
Loss Events           = 0
-----

***** TOTAL RESULTS *****
-----
Number of flows      = 1
Total time           = 14.953157 s
Total packets        = 150
Minimum delay        = 0.000082 s
Maximum delay        = 0.007593 s
Average delay        = 0.000193 s
Average jitter       = 0.000098 s
Delay standard deviation = 0.000611 s
Bytes received       = 15000
Average bitrate      = 8.025061 Kbit/s
Average packet rate  = 10.031326 pkt/s
Packets dropped       = 0 (0.00 %)
Average loss-burst size = 0 pkt
Error lines          = 0
-----

root@raghav-VirtualBox:/home/raghav/DITG/ditg/bin#

```

Figure 14: receiver log file.

In the figure we can see that there are 150 packets and the minimum as well as the maximum delay along with average delay, average jitter, average bitrate, and minimum delay.

The values of these are shown in the table below:

Table 2: Results of simple mininet using D-ITG

S.No.	Host	Minimum Delay in 's'	Maximum Delay in 's'	Average Delay in 's'	Average Bitrate in 'kbit/s'
1	2	0.000082	0.007593	0.000193	8.025061

8.2 Experimenting with SDN

To evaluate the system, I created single controller topology in Mininet where the topology consists of 50,100,200,300,400 and 500 hosts. I used a script to flood the capacity of the switches. After flooding the capacity, I analysed the behaviour of the switches with respect to the controller. For filling the capacity, different number of packets from hosts to different hosts simultaneously to understand the network behaviour under the load. I have evaluated the performance using POX controller.

8.2.1 Calculations for Single Topology

For single topology, the controller in Mininet with port no and IP address of the machine. The POX controller is invoked using command `“./pox.py forwarding.l2_pairs openflow.of_01 -- port=6633”` in one machine to establish the connection between the switch and the controller. So, to evaluate the performance, I fill the capacity with a different number of packets. I flood switches by simultaneously sending different packets. I calculate the maximum delay, average bitrate, and load jitter by observing the flow of the packets. Then, I observed the switch controller communication to compare it with the other topology. Once the traffic bridge between the hosts and client is started with the help of xterm and D-ITG tool, I see how the controller responds and switch manages the traffic.

After running the topologies and performing the experiments I have calculated the important parameters that are affecting the traffic in the network. I have calculated the Maximum delay which emphasizes on the behaviour and differentiate between linear, single and tree topology network performance. Another parameter I calculated was the average jitter. It is one of the important factors for networking infrastructure which is to be able to handle the load in a functional manner. As the load in any network cannot be predictable and can any time reach the maximum. So, I calculated the average jitter to highlight the reaction of the topology. I have also evaluated the network based on performance using average bitrate. I flooded the switches with different flows. I sent many packets using D-ITG tool to different host and observe the flow, plotted the graph for different parameters that are affecting the network performance. I analysed the output of the D_ITG tool and plotted a graph for different values. The performance is evaluated for POX controllers.

Below the graph in Figure 15 and Table 3 show a maximum delay for the value collected for different packets for a single controller. X-axis represents the number of hosts for each experiment and y-axis represents the delay calculated for different packets in seconds.

Table 3: Maximum Delay for Single Topology

S.No.	Number Of Host	POX controller
1	50	0.001947
2	100	0.002048
3	200	0.002113
4	300	0.001383
5	400	0.002250
6	500	0.001222

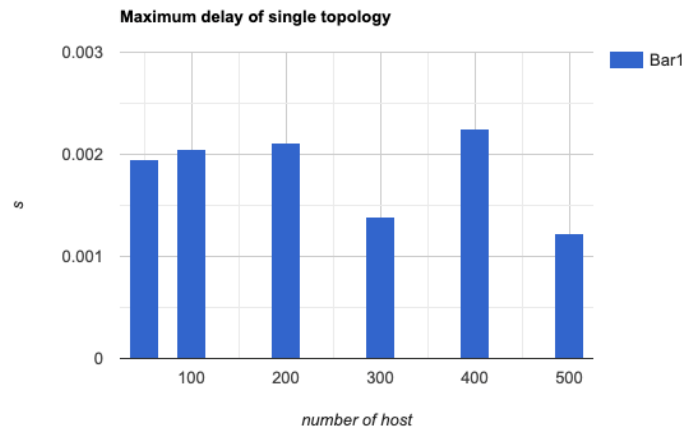


Figure 15: Maximum Delay for Single Topology

Below the graph in Figure 16 and Table 4 show an average jitter for the value collected for different packets for a single controller. X-axis represents the number of hosts for each experiment and y-axis represents the average jitter calculated for different packets in seconds.

Table 4: Average Jitter for Single Topology

S.No.	Number Of Host	POX controller
1	50	0.000070
2	100	0.000051
3	200	0.000044
4	300	0.000053
5	400	0.000056
6	500	0.000074

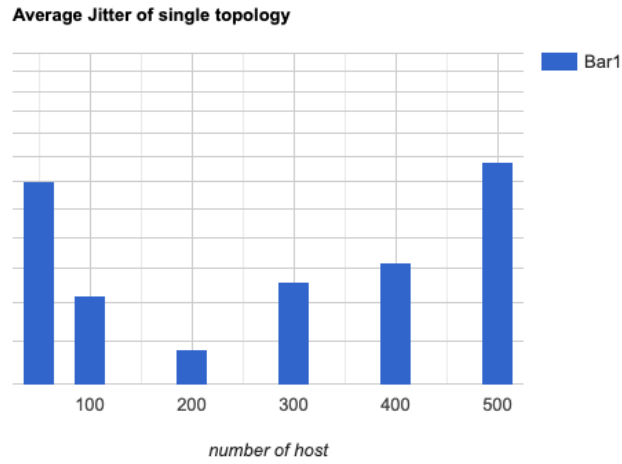


Figure 16: Average Jitter Of Single Topology

The graph in Figure 17 and Table 5 represent the average bitrate for the value collected for different packets for a single controller. X-axis represents the number of hosts for each experiment and y-axis represents the average jitter calculated for different packets in seconds.

Table 5: Average Bitrate for Single Topology

S.No.	Number Of Host	POX controller
1	50	8.027560
2	100	16.010545
3	200	16.006785
4	300	16.007845
5	400	16.003731
6	500	15.989496

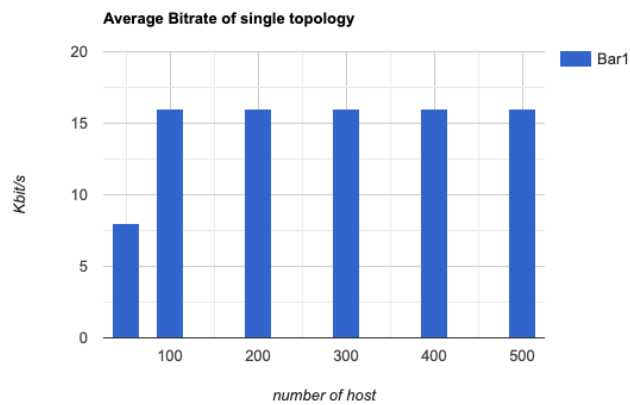


Figure 17: Average Bitrate of Single Topology

8.2.2 Calculations for Linear Topology

For the linear topology, to evaluate the system, I created single controller topology in mininet where there are 50,100 and 200 hosts and a remote controller. I have configured the controller with port number and IP address of the machine. After setting up the topologies and conducting the experiments I have calculated the important parameters affecting the network traffic.

The graph in Figure 18 and Table 6 show a maximum delay for the value collected for different packets for a single controller. X-axis represents the number of hosts for each experiment and y-axis represents the delay calculated for different packets in seconds.

Table 6: Maximum Delay for Linear Topology

S.NO.	Number Of host	POX controller
1	50	0.001605
2	100	0.002912
3	200	0.003897

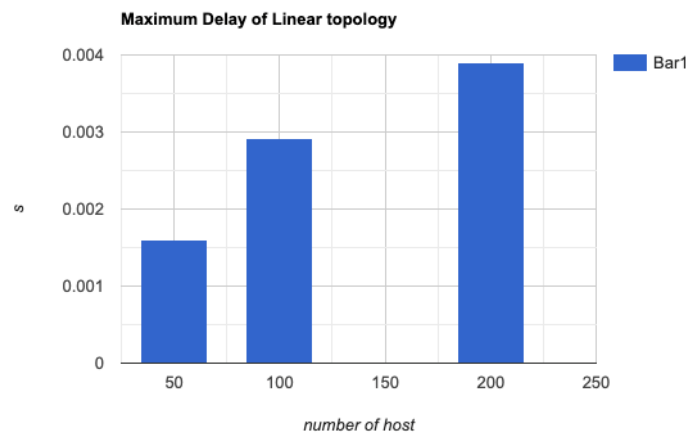


Figure 18: Maximum Delay of Linear Topology

The graph in Figure 19 and Table 7 represent an average jitter for the value collected for different packets for a single controller. The x-axis represents the number of hosts for each experiment and the y-axis represents the average jitter calculated for different packets in seconds.

Table 7: Average jitter for Linear Topology

S.NO.	Number Of host	POX controller
1	50	0.000077
2	100	0.000230
3	200	0.000266

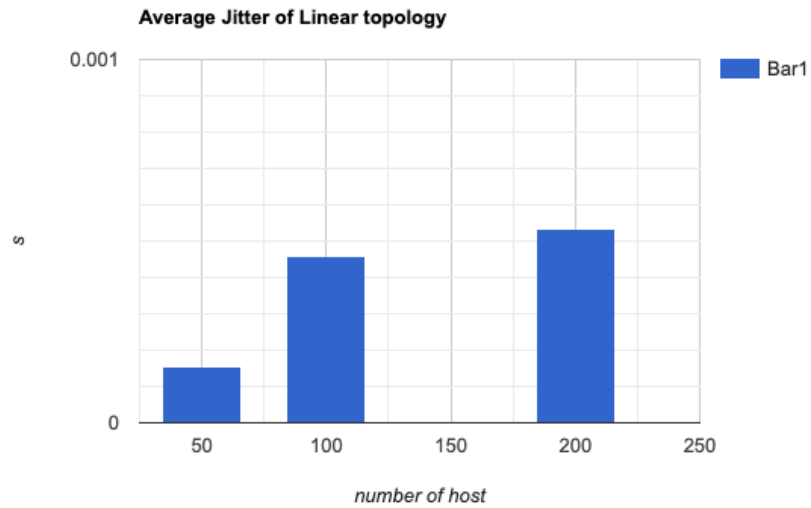


Figure 19: Average Jitter of Linear Topology

The graph in Figure 20 and Table 8 show average bitrate for the value collected for different packets for a single controller. The X-axis represents the number of hosts for each experiment and the y-axis represents the average jitter calculated for different packets in seconds.

Table 8: Average Bitrate for Linear Topology

S.NO.	Number Of host	POX controller
1	50	15.991392
2	100	23.806892
3	200	23.791837

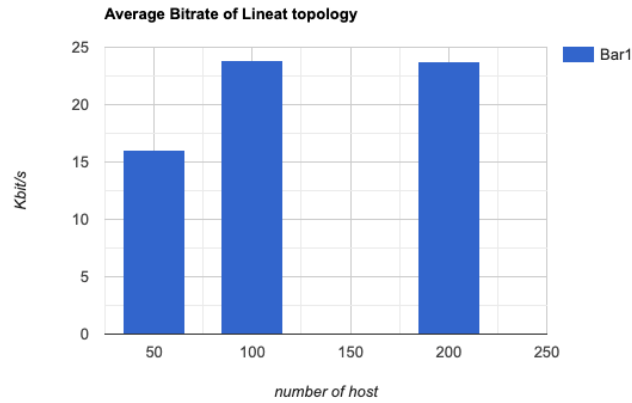


Figure 20: Average Bitrate of linear Topology

8.2.3 Calculations for Tree Topology

For the Tree topology, to evaluate the system, I created single controller topology where the host are 16, 64 and 128, the depth of tree is 4, 6 and 7 respectively, and a remote controller. I have configured the controller in Mininet with port number and IP address of the machine. After setting up the topologies and conducting the experiments I have calculated the important parameters affecting the network traffic.

The graph in Figure 21 and Table 9 show a maximum delay for the value collected for different packets for a single controller. The x-axis represents the number of hosts for each experiment and the y-axis represents the delay calculated for different packets in seconds.

Table 9: Maximum Delay for Tree Topology

S.NO.	Depth OF Tree	Number Of Host	POX controller
1	4	16	0.002122
2	6	64	0.001992
3	7	128	0.001839

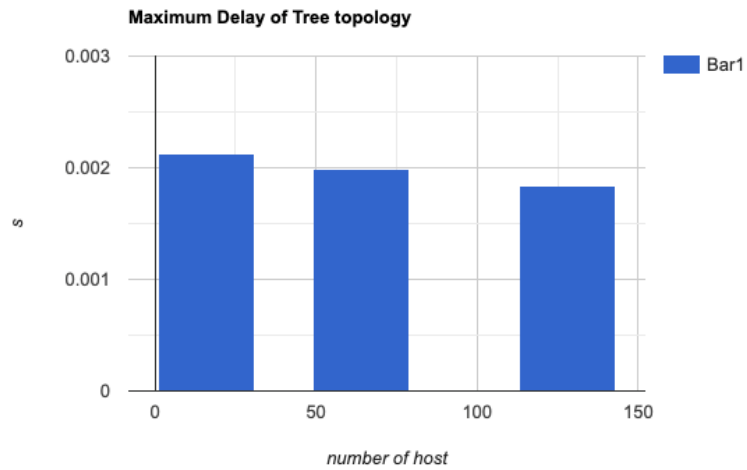


Figure 21: Maximum Delay of Tree Topology

The graph in Figure 22 and Table 10 represent the average jitter for the value collected for different packets for a single controller. The x-axis represents the number of hosts for each experiment and the y-axis represents the average jitter calculated for different packets in seconds.

Table 10: Average jitter for Tree Topology

S.NO.	Depth OF Tree	Number Of Host	POX controller
1	4	16	0.000078
2	6	64	0.000077
3	7	128	0.000126

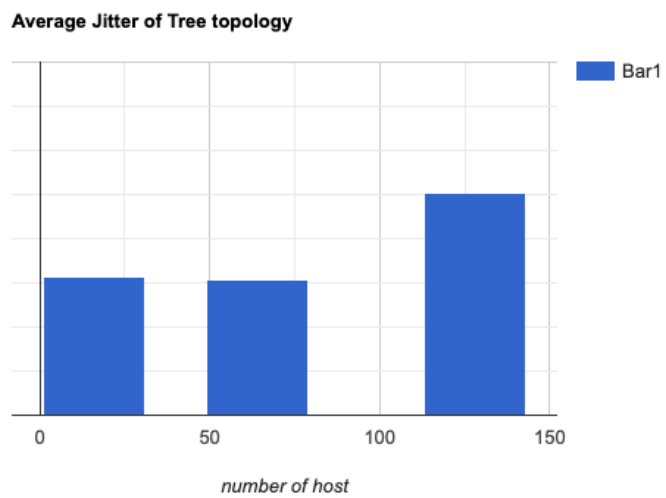


Figure 22: Average Jitter of Tree Topology

The graph in Figure 23 and Table 11 is average bitrate for the value collected for different packets for a single controller. The x-axis represents the number of hosts for each experiment and the y-axis represents the average jitter calculated for different packets in seconds.

Table 11: Average Bitrate for Tree Topology

S.NO.	Depth OF Tree	Number Of Host	POX controller
1	4	16	23.965276
2	6	64	23.915252
3	7	128	23.873451

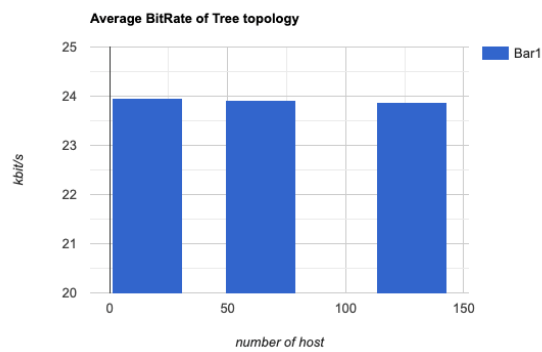


Figure 23: Average Bitrate of Tree Topology

8.2.4 Comparing the Performance of Single, Linear and Tree Topology.

Comparing the Performance of Single and Linear Topology. Figure 24 shows the maximum delay performance of single, linear and tree topology using POX controller.

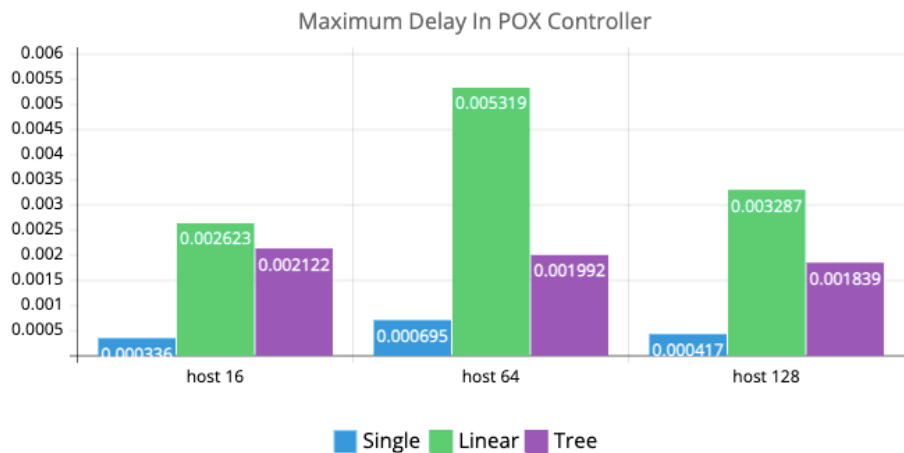


Figure 24: Maximum Delay In POX controller

Using POX controller, the maximum delay is less for single and tree topology as compared to Linear topology.

Comparing the Performance of Single and Linear Topology. Figure 25 shows the Average Jitter performance of single, linear and tree topology using POX controller.

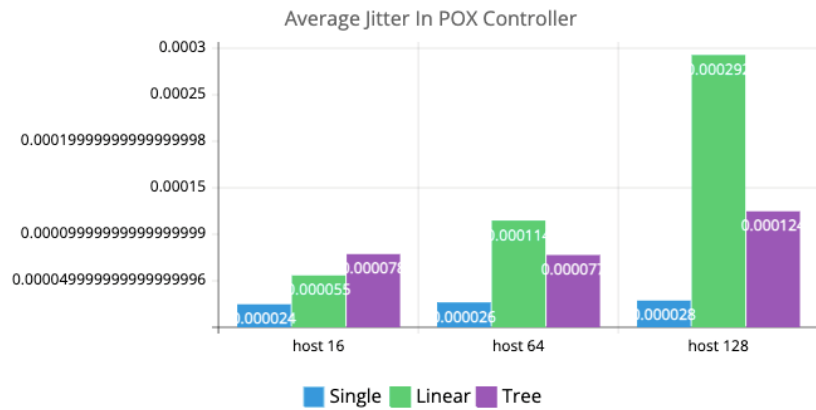


Figure 25: Average jitter in POX Controller

Using POX controller, the Average Jitter is less for single and tree topology as compared to Linear topology.

Comparing the Performance of Single and Linear Topology. Table 12 shows the Average Bitrate performance of single, linear and tree topology using POX controller.

Table 12: Average Bitrate using POX Controller

S.No.	Host	Single	Linear	Tree
1	16	23.969295	23.714925	23.965276
2	64	23.94618	23.936395	23.915252
3	128	23.969295	23.714925	23.873451

Using POX controller, the Average Jitter is less for linear and tree topology as compared to Single topology.

8.3 Mobility in Mininet WIFI

The mobility pattern of nodes in the network is considered to affect the performance of software-defined wireless networks. In Wireless Local Area Network (WLAN), the nodes are mostly considered as mobile devices such as laptops, mobile phones and tablets which are being used by people to get access to a wireless local area network. Moreover, the wireless local area network consists of fixed access points (Aps) and several mobile devices. The access point usually connects to several mobile devices for example university students on the campus and so on. Human mobility plays a major role in the evaluation of the performances of a wireless local area network. Moreover in such a network, node mobility is characterized by the trajectory, location of the mobile nodes and by migration patterns between Aps in the network. When users migrate or move from one location to another location, then they de associate and associate to APs in the network. Another important point is that the mobility of nodes is often intermittent in the time domain. In other words, a user can be related to an access point, A at time t_1 , then disconnected from it for a while, and later (at time $t_2 \geq t_1$) "re-connected to it" in the network connects to a different access point, say B. This form of mobility pattern is common, for example, in a campus network, where students move from, say, classroom to library, and turn off their laptops while moving from one location to the other.

For creating the scenario, I used Mininet WIFI. Mininet-WIFI is a fork of Mininet which allows the use of both WIFI Stations and Access Points. Mininet-WiFi-only add WIFI features, and you can work with it like you were working with Mininet. In that I used 3 mobile devices and one access point. As Mininet-WiFi supports the following mobility models like Random Walk, Truncated Levy Walk, Random Direction, Random Way Point, Gauss Markov, Reference Point and Time Variant Community. So, I used Propagation model and kept it in long distance. Then I created a host and one controller along with 3 stations which I named them as mobile1, mobile2 and mobile3. For creating the stations I used `net.addStation` function of Mininet-WIFI. Then I created the link by using the function called as `net.addLink` between ap1 and host h1 and the rest of the three station which I named as sta1, sta2 and sta3. Then I started the mobility with `net.startMobility` function and set the initial and the final position of the mobiles. As soon as the mobility is started one of the attacker which is in mobile1 starts the DDOS attack, since the stations and the access points are linked together so the whole system got stucked and the DDOS attack is

successful. For the starting and the ending of the mobility to be seen in Mininet-WiFi Graph I used the net.plotGraph function.

8.3.1 Results and Discussions

I have created a scenario of 3 mobile devices which are searching for a network and a time when they found the network they try to connect with the network. While connecting to the network one of the mobile devices is an attacker and he tries to attack other devices. While he is performing the attack, he tried to do it on both the devices but got success only on one device. As soon as the attack is performed successfully the network gets stopped and the whole system gets hanged out. The detail of this experiment is given below.

The initial state of the network is when the 3 mobile devices are trying to connect to the network where ap1 is the network tower is shown below:

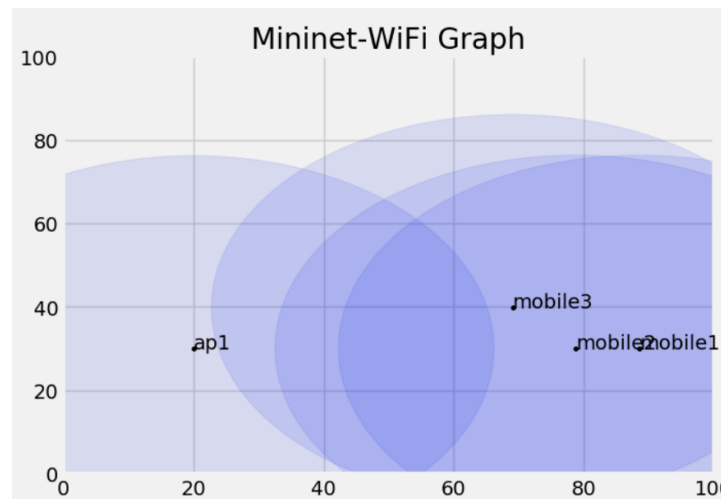


Figure 26: Initial state of Mobility

The final state of the network is when the mobile devices are connected to the network successfully without being attacked in the second image. As we can see the devices reached the network without any disturbance and no attacks are performed up till now.

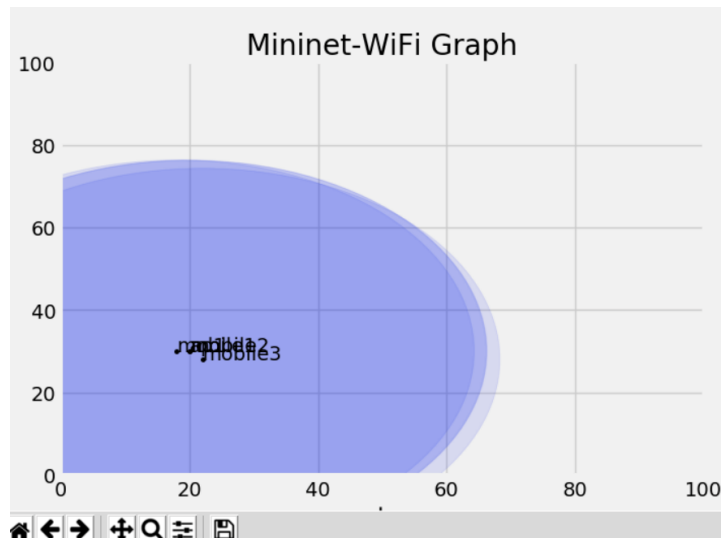


Figure 27: Final state of Mobility

Now I will show when there is an attacker between any one of the mobile devices and he tries to perform the DDOS attack on the other network which makes the whole network stop and not in movement. In the image, you can see that the attack is initiated. As in the image you can see that the attacker is a mobile 1 with an IP address of 10.0.0.2 and the victim is a mobile 3 with an IP address 10.0.0.4. The command used for attacking is “hping3 -S -flood -V -p 80 10.0.0.4” hping3 is used for performing the distributed denial of service attack and flood is used for sending multiple packets in a network. Flood shoot at discretion and replies were ignored that is why it will send packets as fast as possible. Here -p is the port number is which a TCP open port 80. As in the above image you can see the attack is started and hping is in flood mode right now.

As soon as the attack is initiated, we can see in the mininet graph that the network got stopped and all the devices were unable to connect to the network as shown in the image below:

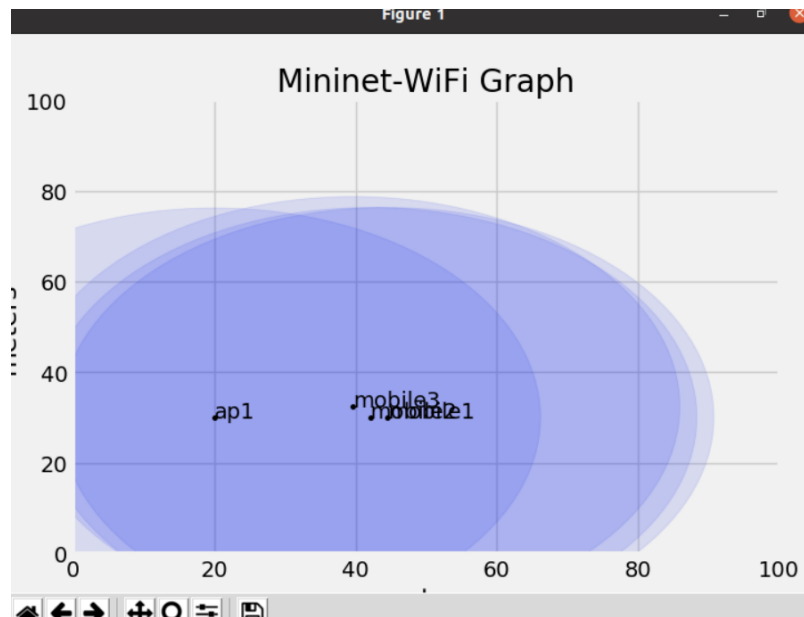


Figure 28: Mobility after DDoS attack

8.4 Telemetry in Mininet-WIFI

The term telemetry is used for technologies that measure and collect data from remote locations and transmit this data to receiving systems for monitoring and analysis. Traditional examples of telemetry are Monitoring data from space crafts, Animal tracking devices, Automobile sensors for fuel level, engine heat, vehicle speed and more, Heart monitors (EKG), Convicted felon ankle bracelets and Wearables such as Fitbit health monitoring devices. Telemetry provides the ability to access data from remote locations. Very often, these locations are difficult or expensive to get to and have limited access to power and physical networks

Mininet-wifi provides the telemetry function, through which we can capture different types of data like the following.

The first is RSSI (Received Signal Strength Indicator). RSSI, acronym for Received Signal Strength Indicator, is the value that indicates the strength of the received signal. RSSI allows you to know whether a signal is sufficient to establish a wireless connection. A negative dBm value is used. The higher the number, the better the signal. For example, an RSSI of -70 dBm or higher usually indicates that the modem is in an excellent coverage area. Put simply, the closer the RSSI value is to 0 dBm, the stronger the signal.

TX and RX. These are abbreviations for Transmit and Receive, respectively. Note that these metrics are referenced to the server being monitored.

Transmit from this server and receive to this server. Units are in Bytes. We consider also the following parameters:

tx_packets = TX packets is total number of packets transmitted. TX errors, TX dropped, and TX overruns are like RX equivalents. TX carriers is several packets that experienced loss of carriers. This usually happens when link is flapping

tx_bytes = Is the total number of bytes transmitted.

tx_dropped = is the total number of data transmitted dropped

tx_compressed = is the number of compressed data in transmission

tx_errors = errors in transmission

rx_errors = errors in receiving

rx_packets = total number of packets received

rx_bytes = total number of bytes received

rx_compressed = data compressed in receiving

8.4.1 Methodology

I used mininet-wifi telemetry to monitor the data and then performing the DDOS attack on the stations. I created the topology using 4 stations ("sta1, sta2, sta3 and sta4"), one access point("ap1") and a controller("c1"). Then I used the telemetry function for gathering the data as tx_packets, tx_bytes and rssi.

Once the network was up and running, I used the HPING tool to perform the DDOS attack and Wireshark to confirm is the attack is possible or not. After the attack I can see the simulation of attack on both the Wireshark as well as the Mininet graph. First, I will show the packets transferred in a single graph and then in separate one.

For testing the bandwidth, I used iperf command in mininet and the result I got is shown below

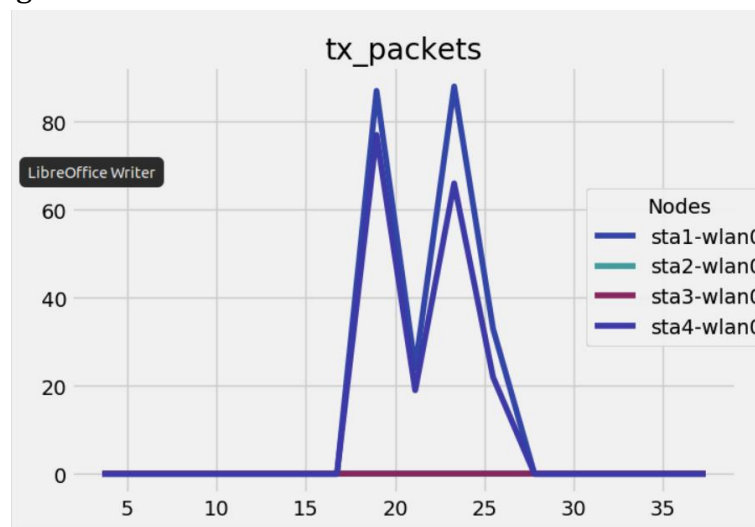


Figure 29: Testing the bandwidth with tx_paxkets

From the Mininet graph we can see the bandwidth of sta1 and sta2.

Then I used the HPING tool for performing the DDOS attack. I used the manual simulation of the attack using the station one and station 3. Station 3 is the attacked one, so the data is shown for the sta3

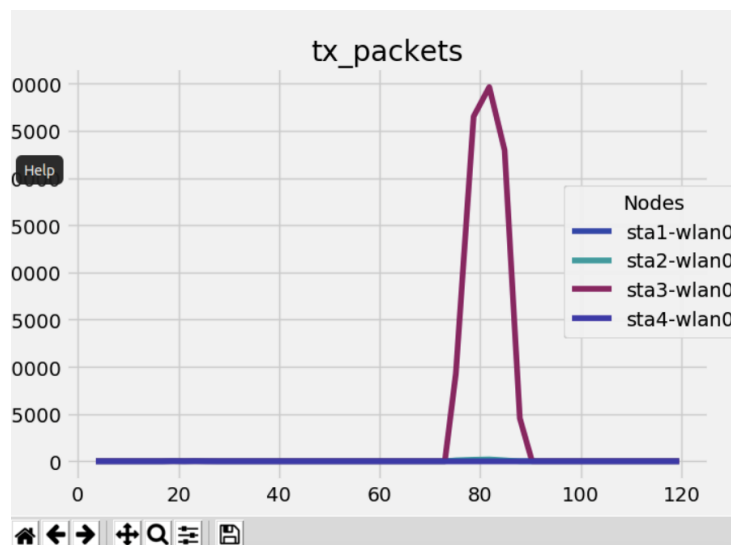


Figure 30: DDoS attack on tx_packets

Then I separated the graph for separation of Graph I changed the python code which was set to single=True and used the data type as rssi.

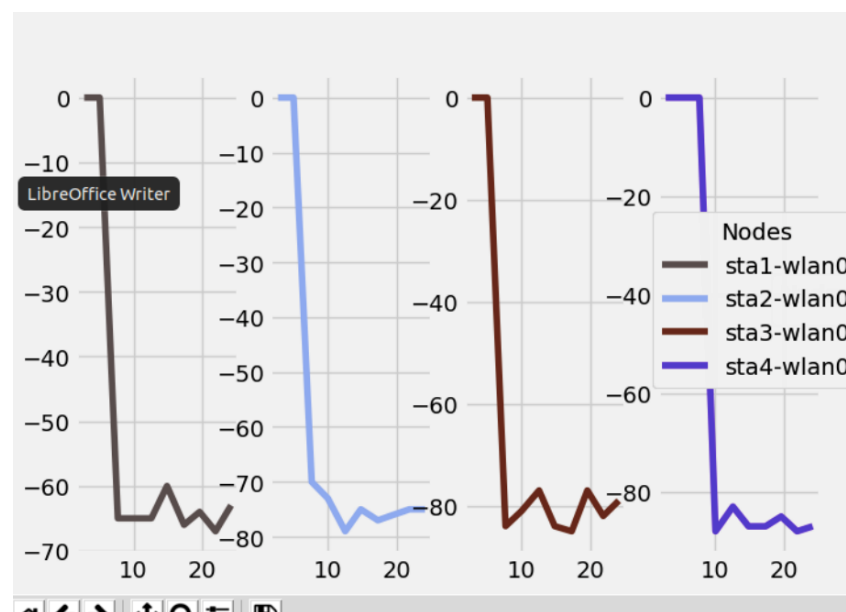


Figure 31: Separation of Graphs using rssi

After performing the DDOS attack using HPING tool all the following data is showing the same output

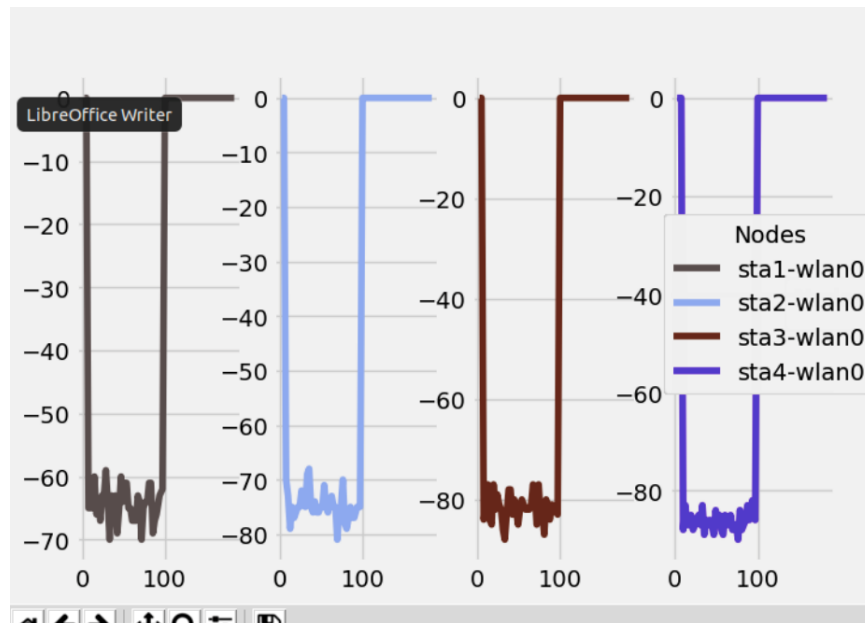


Figure 32: DDoS attack on rssi

By performing the DDoS attack the value is equal to 0 which means the signal is stronger and the DDoS attack is sending more packets that is why it is showing stronger signal. We can see in the same graph that after performing the DDoS attack all the stations are at 0 and the signal is stronger.

For confirming the attack, I used a network monitoring tool named Wireshark. Wireshark is software for protocol analysis or packet sniffer (literally packet sniffer) used for the solution of network problems, for the analysis (troubleshooting) and the development of protocols or software for communication and teaching purposes, having all the features of a standard protocol analyser. In the image given below you can see that there is a source IP and a Destination IP. The source IP is the one from where the attack is initiated, and the destination is attacked IP. It also shows the protocol on which the DDOS attack is happening i.e., TCP protocol. Wireshark also gives information regarding the time, number of packets and the length of packets transferred.

No.	Time	Source	Destination	Protocol	Length	Info
15577	13.823523017	10.0.0.1	10.0.0.3	TCP	56	59996 → 80 [SYN]
15578	13.823548931	10.0.0.1	10.0.0.3	TCP	56	59997 → 80 [SYN]
15579	13.823573947	10.0.0.1	10.0.0.3	TCP	56	59998 → 80 [SYN]
15580	13.823596814	10.0.0.1	10.0.0.3	TCP	56	59999 → 80 [SYN]
15581	13.823619775	10.0.0.1	10.0.0.3	TCP	56	59100 → 80 [SYN]
15582	13.823642735	10.0.0.1	10.0.0.3	TCP	56	59101 → 80 [SYN]
15583	13.823665917	10.0.0.1	10.0.0.3	TCP	56	59102 → 80 [SYN]
15584	13.823690813	10.0.0.1	10.0.0.3	TCP	56	59103 → 80 [SYN]
15585	13.823713797	10.0.0.1	10.0.0.3	TCP	56	59104 → 80 [SYN]
15586	13.823736205	10.0.0.1	10.0.0.3	TCP	56	59105 → 80 [SYN]
15587	13.823758681	10.0.0.1	10.0.0.3	TCP	56	59106 → 80 [SYN]
15588	13.823782769	10.0.0.1	10.0.0.3	TCP	56	59107 → 80 [SYN]
15589	13.823805643	10.0.0.1	10.0.0.3	TCP	56	59108 → 80 [SYN]
15590	13.823829278	10.0.0.1	10.0.0.3	TCP	56	[TCP Port numbe
15591	13.823852096	10.0.0.1	10.0.0.3	TCP	56	[TCP Port numbe
15592	13.823875859	10.0.0.1	10.0.0.3	TCP	56	[TCP Port numbe
15593	13.823898811	10.0.0.1	10.0.0.3	TCP	56	[TCP Port numbe
15594	13.823920918	10.0.0.1	10.0.0.3	TCP	56	[TCP Port numbe
15595	13.823942975	10.0.0.1	10.0.0.3	TCP	56	[TCP Port numbe
15596	13.823964883	10.0.0.1	10.0.0.3	TCP	56	[TCP Port numbe
15597	13.823988251	10.0.0.1	10.0.0.3	TCP	56	[TCP Port numbe
15598	13.824150118	10.0.0.1	10.0.0.3	TCP	56	[TCP Port numbe
15599	13.824180952	10.0.0.1	10.0.0.3	TCP	56	[TCP Port numbe
15600	13.824204243	10.0.0.1	10.0.0.3	TCP	56	[TCP Port numbe
15601	13.824226929	10.0.0.1	10.0.0.3	TCP	56	[TCP Port numbe

Figure 33: Wireshark as a monitoring tool

8.5 AR.Drone 2.0 Dos attack using Coppeliasim

Aerial robotics has seen considerable progress recently with several technology companies, such as Google, Amazon, and Facebook. Today, UAVs are successfully employed in many outdoor applications, such as product delivery, agriculture, area monitoring, maritime patrol, mapping, and search and rescue. In these situations, an aerial robot is expected to operate autonomously or remotely piloted for extended periods of time while navigating potentially unexplored and highly dynamic areas.

According to Wilson [21], a drone is a UAV without a pilot that can be manoeuvred by remote control or by onboard computers providing autonomous behaviours. While most people are capable of remotely piloting a drone, they usually do not care about the information contained in it (i.e., data from the camera, sonar, laser, radar, inertial measurement unit (IMU), Global Positioning System (GPS), among other sensors).

Information security [22] is a critical aspect important when UAVs are involved. If a robot does not have the minimum security for autonomous navigation, hackers can bring it down immediately. Furthermore, confidential information contained in the aerial robot can be stolen, such as camera images and GPS position. Finally, hackers can get full control of the drone and perform several unexpected activities, such as collide with objects near it.

My goal is to perform an experimental evaluation of Denial of Service (DoS) attack tool on the AR.Drone 2.0, while keeping the vehicle as safe as possible. I applied DoS attack tool, which is done by using Hping3. The main contribution of this experiment is to provide an

empirical evaluation of the effects caused by DoS attacks on UAVs such as the AR.Drone 2.0.

8.5.1 Theoretical background

In this section, I will provide a brief overview of the theoretical background behind the experimental evaluation. More specifically, firstly I will discuss some computer security concepts, then the notion of reconnaissance attacks and lastly the fundamentals of DoS attacks.

8.5.2 Computer Security Principles

Computer networks and information technology systems have made a significant impact in our society, from powerful smartphones to e-commerce and cloud computing solutions. Within this scenario, there is a need to secure the systems that hold data about citizens, corporations, and government agencies.

Computer security as a field, is the study of how to make computer systems resistant to misuse. One example of abuse is a cyber-attack. Any action taken to undermine the functions of a computer network or device can be viewed as a cyber-attack. Three different aspects decompose information security: confidentiality, integrity, and availability:

Confidentiality - is the ability of the computing system to prevent disclosure of information to unauthorized parties.

Integrity - is the ability of the computer system to guard against improper information modification or destruction.

Availability - is the ability of the system to in fact deliver its service. Suppose that Alice, the pilot, wants to send a message to the drone without anybody else learning its contents. If no one else apart from Alice and the drone can hear the message, then we say Alice and the drone have confidentiality.

Integrity is the correctness of the data, for example, ensuring that the message the drone receives the same one that Alice intended to send. Availability is being able to always have a communication channel between Alice and the drone. A typical availability attack occurs when an attacker cuts the communication channel between Alice and the drone. The focus of this paper is the study of availability attacks towards the AR.Drone 2.0, and in particular an attack known in the literature as Denial of Service.

Reconnaissance

The first step for any cyber-attack consists of information gathering about a targeted network or device. This phase is called reconnaissance. Port scan is one of the most used reconnaissance

attacks. A port scan is used to check for open or closed network ports and for used or unused services. The services may or may not have a vulnerability that the attacker could exploit. The Network Mapper or Nmap is an automated tool that discovers hosts and services on a computer network through port scans. As shown in the image below:

```
root@raghav-VirtualBox:~# nmap --top-ports 10 192.168.1.1
Starting Nmap 7.80 ( https://nmap.org ) at 2022-04-21 00:13 CEST
Nmap scan report for 192.168.1.1
Host is up (0.0028s latency).

PORT      STATE      SERVICE
21/tcp    open       ftp
22/tcp    filtered  ssh
23/tcp    filtered  telnet
25/tcp    filtered  smtp
80/tcp    filtered  http
110/tcp   filtered  pop3
139/tcp   filtered  netbios-ssn
443/tcp   filtered  https
445/tcp   filtered  microsoft-ds
3389/tcp  filtered  ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 1.32 seconds
root@raghav-VirtualBox:~#
```

Figure 34: Reconnaissance using nmap

Denial of Service Attacks

A Denial-of-Service attack is characterized by an attempt of an attacker to prevent legitimate users of a service from using the desired resources. Moore et al. groups DoS attacks in two different classes: logic attacks and resource attacks. In this paper, I am interested in analysing the impact produced by DoS attacks on the AR.Drone 2.0.

UAVs have recently experienced a massive increase in their areas of application, due to a combination of advancements in hardware (higher payloads, more precise sensors, component miniaturization) and software (more efficient algorithms, scaling in data acquisition and storage, embedded processing). These areas of application include aerial photography, surface mapping, surveillance, scene reconstruction and 3D modelling, target tracking and product delivery. The quick spread of UAVs – allied to a large variety of shapes, sizes, and prices – have also led to their adoption by a broad range of consumers, from hobbyists to billion-dollar companies, each with its own requirements and goals. For UAV simulation I used CoppeliaSim.

The CoppeliaSim is a robotics simulator, with integrated development environment, is based on a distributed control architecture: each object/model can be individually controlled via an embedded script, a plugin, a ROS node, a remote API client, or a custom solution. This makes CoppeliaSim very versatile and ideal for multi-robot applications. Controllers can be written in C/C++, Python, Java, Lua, Matlab, or Octave. CoppeliaSim is used for fast algorithm development, factory automation simulations, fast prototyping and verification, robotics related education, remote monitoring, safety double-checking, as digital twin, and much more.

Methodology

My primary goal is to understand and analyse the consequences resulting from launching the DoS attacks on UAVs such as the AR.Drone 2.0. The proposed methodology describes a scenario in which I made a movement of drone in CoppeliaSim, and an attacker launches reconnaissance and DoS attacks.

For creating the scenario, I used CoppeliaSim in which there are trees and there is a path in which the drone is moving. As shown in the image below:

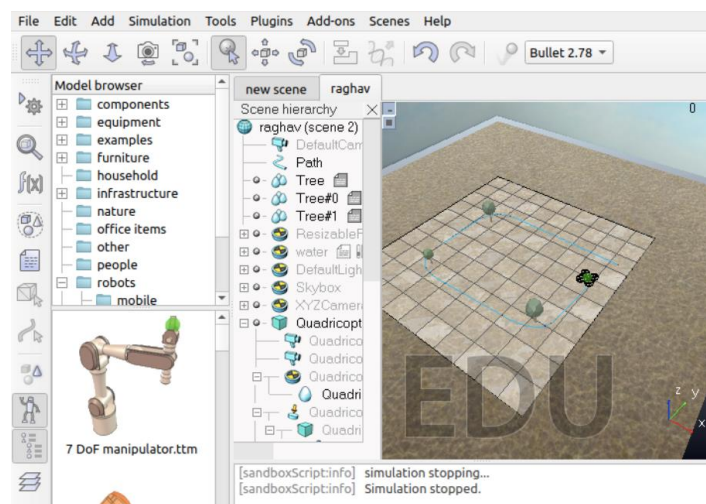


Figure 35: CoppeliaSim Scenario

For the movement of drone, I used this script in the target of quadcopter:

```
function sysCall_threadmain()
    target=sim.getObjectAssociatedWithScript(sim.handle_self)
    path=sim.getObjectHandle("Path")
    sim.followPath(target,path,1,0,0.3,15)
```

end

```
function sysCall_cleanup()
```

end

As the target is specifying that the drone is handled by self-simulation and the path is set and drone is set to follow that path. The simulation works perfectly. Now in the second part I introduced the DOS attack which makes the speed of drone slow.

For the initiation of the Dos attack firstly I used the NMAP tool to gather the information about the drone. As the AR. Drone.2.0 IP address is 198.168.1.1 so using Nmap to get the information about the open ports. As the drone works on telnet so the open ports 23 which is a tcp port.

As soon as the attacker gets the information about the drone, he simulates the Dos attack using HPING tool. This attack makes the speed of drone slow as compared to the normal one. In the image below shows at this point the attack is started and the speed gets slow.



Figure 36: DDoS attack on Ar. Drone

The hping command used to perform the DOS attack which I kept in fast and flooding mode as shown in the image below:

```

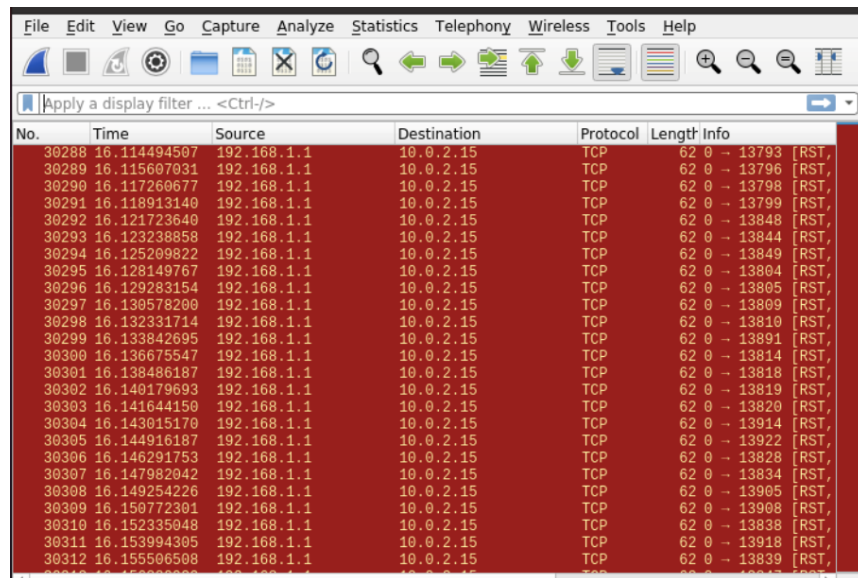
root@raghav-VirtualBox:~# hping3 --fast --flood 192.168.1.1
HPING 192.168.1.1 (enp0s3 192.168.1.1): NO FLAGS are set, 40 headers + 0 data b
ytes
hping in flood mode, no replies will be shown
^C
--- 192.168.1.1 hping statistic ---
987702 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

```

Figure 37: DDoS attack

As from the image above we can see that the packets are transmitted are 987702 which makes the drone flooded and slows the speed of the drone.

For monitoring I used Wireshark. In the image below shows that the Dos attack is carried on the drone with IP address as 198.168.1.1



No.	Time	Source	Destination	Protocol	Length	Info
30288	16.114494507	192.168.1.1	10.0.2.15	TCP	62	0 → 13793 [RST,
30289	16.115607031	192.168.1.1	10.0.2.15	TCP	62	0 → 13796 [RST,
30290	16.117260677	192.168.1.1	10.0.2.15	TCP	62	0 → 13798 [RST,
30291	16.118913140	192.168.1.1	10.0.2.15	TCP	62	0 → 13799 [RST,
30292	16.121723640	192.168.1.1	10.0.2.15	TCP	62	0 → 13848 [RST,
30293	16.123238858	192.168.1.1	10.0.2.15	TCP	62	0 → 13844 [RST,
30294	16.125209822	192.168.1.1	10.0.2.15	TCP	62	0 → 13849 [RST,
30295	16.128149767	192.168.1.1	10.0.2.15	TCP	62	0 → 13804 [RST,
30296	16.129283154	192.168.1.1	10.0.2.15	TCP	62	0 → 13805 [RST,
30297	16.130578200	192.168.1.1	10.0.2.15	TCP	62	0 → 13809 [RST,
30298	16.132331714	192.168.1.1	10.0.2.15	TCP	62	0 → 13810 [RST,
30299	16.133842695	192.168.1.1	10.0.2.15	TCP	62	0 → 13891 [RST,
30300	16.136675547	192.168.1.1	10.0.2.15	TCP	62	0 → 13814 [RST,
30301	16.138486187	192.168.1.1	10.0.2.15	TCP	62	0 → 13818 [RST,
30302	16.140179693	192.168.1.1	10.0.2.15	TCP	62	0 → 13819 [RST,
30303	16.141644150	192.168.1.1	10.0.2.15	TCP	62	0 → 13820 [RST,
30304	16.143015170	192.168.1.1	10.0.2.15	TCP	62	0 → 13914 [RST,
30305	16.144916187	192.168.1.1	10.0.2.15	TCP	62	0 → 13922 [RST,
30306	16.146291753	192.168.1.1	10.0.2.15	TCP	62	0 → 13828 [RST,
30307	16.147982042	192.168.1.1	10.0.2.15	TCP	62	0 → 13834 [RST,
30308	16.149254226	192.168.1.1	10.0.2.15	TCP	62	0 → 13905 [RST,
30309	16.150772301	192.168.1.1	10.0.2.15	TCP	62	0 → 13908 [RST,
30310	16.152335048	192.168.1.1	10.0.2.15	TCP	62	0 → 13838 [RST,
30311	16.153994305	192.168.1.1	10.0.2.15	TCP	62	0 → 13918 [RST,
30312	16.155506508	192.168.1.1	10.0.2.15	TCP	62	0 → 13839 [RST,

Figure 38: Wireshark for confirmation

As in the image we can see that the dos attack is in flooding mode and is sending packets to the destination using tcp protocol.

Future Aspects:

In future I can use the DDOS attack using Hping to stop the drone and gain access of the camera and the navigation control.

9. CONCLUSION

The goal of this thesis to consider some tools for simulation of different kinds of networks, in particular Mininet and Mininet-WIFI, and use then to create and experiment with some attack scenarios in IOT. Our goal was to show that is relatively easy create such scenarios and study in a synthetic way how attacks may be carried out in IOT. From the results of the experiments, I can conclude the following results:

- The maximum delay for single topology and tree topology is less as compared to linear topology.
- The average jitter of single and tree topology are less than linear topogy.
- The average bitrate of single topology is more than linear and tree topology.
- In the scenario of 3 mobiles trying to connect to a tower that are attacked by a dos attack it very easy to make the system stop.
- For Telemetry in IoT, the DDoS attack makes the stations to show the same data about rssi.
- For Ar. Drone 2.0 scenario Dos attack can easily make the speed of the drone slow.

9.1 Future Work

In future, one can evaluate the performance of different SDN controllers with different network simulation tools and performance metrics.

For the mobility scenario, one can gain access to the mobile tower and make the other mobiles connect to a fake unauthorised tower which can lead to the loss of crucial information.

For Ar. Drone 2.0, one can use the DDOS attack using Hping to stop the drone and gain access to the camera and the navigation control.

10. REFERENCES

- [1] <https://cyberstartupobservatory.com/cyber-range-what-it-is-what-it-is-not-and-what-it-will-be/>
- [2] <https://resources.infosecinstitute.com/topic/types-of-cyber-ranges-compared-simulations-overlays-emulations-and-hybrids/>
- [3] https://www.researchgate.net/publication/356754456_Study_on_SDN_with_Security_Issues_Using_Mininet
- [4] D. B. Hoang and M. Pham, "On software-defined networking and the design of SDN controllers", 6th International Conference on the Network of the Future (NOF), Montreal, QC, Canada, (2015), pp. 1-3, doi: 10.1109/NOF.2015.7333307.
- [5] CasimerDeCusatis, Aparicio Carranza, and Jean Delgado-Caceres "Modeling Software Defined Networks using Mininet", Proceedings of the 2nd International Conference on Computer and Information Science and Technology (CIST'16) Ottawa, Canada, (2016) Paper No. 133.
- [6] K. Benzekki, A. El Fergougui, and A. ElbelrhitiElalaoui, "Software-defined networking (SDN): A survey," Security and Communication Networks, (2017), DOI: 10.1002/sec.1737
- [7] https://www.researchgate.net/publication/356754456_Study_on_SDN_with_Security_Issues_Using_Mininet
- [8] F. Ketikci and S. Askar, "Emulation of Software Defined Networks Using Mininet in Different Simulation Environments", 6th International Conference on Intelligent Systems, Modelling and Simulation, Kuala Lumpur, Malaysia, (2015), pp. 205-210, doi: 10.1109/ISMS.2015.46.

[9] mininet.org

[10] Software Defined Networks using Mininet BY Pramod B Patil., Kanchan S. Bhagat., D K Kirange, S. D. Patil

[11] https://www.researchgate.net/publication/221406698_D-ITG_Distributed_Internet_Traffic_Generator

[12] D-ITG V. 2.6.1d Manual by Stefano Avallone, Alessio Botta,^[1]_{SEP}Alberto Dainotti, Walter de Donato, and Antonio Pescap'e University of Naples Federico II

[13]<https://github.com/omkarsuram/SDN-DDoS>

[14]<https://www.sciencegate.app/app/document/download/10.3390/electronics10050583>

[15] Building Open-Source Cyber Range To Teach Cyber Security

[16] <https://onlinedegrees.und.edu/blog/types-of-cyber-security-threats/>

[17] <https://sbscyber.com/resources/top-5-most-common-incident-response-scenarios>

[18] C. W, L. A and W. P, "A roadmap for traffic engineering in SDNOpenFlow networks. Computer Networks," pp. 1-30, 2014.

[19] K. D, R. F., and V. P., "Towards secure and dependable softwaredefined networks," in Proceedings of the second ACM SIGCOMM workshop on "Hot topics in software defined networking, 2013.

[20] Y. Zheng and P. Zhang, "A security and trust framework for virtualized networks and software-defined networking," Security and communication networks, pp. 3059-3069, 2016

[21]R. L. Wilson, "Ethical issues with use of Drone aircraft," 2014 IEEE International Symposium on Ethics in Science, Technology and Engineering, ETHICS 2014, 2014.

[22] M. T. Scholar, "Security Incident Management in Ground Transportation System Using UAVs," no. i, 2015.

11. CODE USED IN THIS THESIS

- Code used for Mobility in Mininet-WIFI

```
#!/usr/bin/python
```

```
'Setting the position of nodes and providing mobility'
```

```
import sys
```

```
from mininet.log import setLogLevel, info
```

```
from mn_wifi.cli import CLI
```

```
from mn_wifi.net import Mininet_wifi
```

```
def topology(args):
```

```
    "Create a network."
```

```
    net = Mininet_wifi()
```

```
    info("*** Creating nodes\n")
```

```
    h1 = net.addHost('h1', mac='00:00:00:00:00:01', ip='10.0.0.1/8')
```

```
    sta1 = net.addStation('mobile1', mac='00:00:00:00:00:02',
```

```
ip='10.0.0.2/8', position='90,30,0')
```

```
    sta3 = net.addStation('mobile3', mac='00:00:00:00:00:02',
```

```
ip='10.0.0.4/8', position='70,30,0')
```

```
    sta2 = net.addStation('mobile2', mac='00:00:00:00:00:03',
```

```
ip='10.0.0.3/8', position='80,30,0')
```

```
    ap1 = net.addAccessPoint('ap1', ssid='new-ssid', mode='g', channel='1',
```

```
position='20,30,0')
```

```
    c1 = net.addController('c1')
```

```
    info("*** Configuring propagation model\n")
```

```
    net.setPropagationModel(model="logDistance", exp=4.5)
```

```
    info("*** Configuring wifi nodes\n")
```

```
    net.configureWifiNodes()
```

```
    info("*** Associating and Creating links\n")
```

```
    net.addLink(ap1, h1)
```

```
    net.addLink(ap1, sta1)
```

```
    net.addLink(ap1, sta2)
```



```

net.addLink(ap1,sta3)
if '-p' not in args:
    net.plotGraph(max_x=100, max_y=100)

if '-c' in args:
    sta1.coord = ['40.0,30.0,0.0', '31.0,10.0,0.0', '31.0,30.0,0.0']
    sta2.coord = ['40.0,40.0,0.0', '55.0,31.0,0.0', '55.0,81.0,0.0']

net.startMobility(time=0, repetitions=5, reverse=False)

p1, p2, p3, p4, p5, p6= dict(), dict(), dict(), dict(), dict(), dict()
if '-c' not in args:
    p1 = {'position': '90.0,30.0,0.0'}
    p2 = {'position': '80.0,30.0,0.0'}
    p3 = {'position': '20.0,30.0,0.0'}
    p4 = {'position': '20.0,30.0,0.0'}
    p5 = {'position': '70.0,40.0,0.0'}
    p6 = {'position': '20.0,30.0,0.0'}

net.mobility(sta1, 'start', time=10, **p1)
net.mobility(sta2, 'start', time=10, **p2)
net.mobility(sta1, 'stop', time=50, **p3)
net.mobility(sta2, 'stop', time=50, **p4)
net.mobility(sta3, 'start', time=10, **p5)
net.mobility(sta3, 'stop', time=50, **p6)
net.stopMobility(time=60)

info("*** Starting network\n")
net.build()
c1.start()
ap1.start([c1])

info("*** Running CLI\n")
CLI(net)

info("*** Stopping network\n")
net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    topology(sys.argv)

```

- Code used in Telemetry in Mininet-WIFI
#!/usr/bin/python

'This uses telemetry() to enable a graph with live statistics'

```

from mininet.node import Controller
from mininet.log import setLogLevel, info

```

```

from mn_wifi.link import wmediumd
from mn_wifi.cli import CLI
from mn_wifi.net import Mininet_wifi
from mn_wifi.wmediumdConnector import interference

def topology():
    "Create a network."
    net = Mininet_wifi(controller=Controller, link=wmediumd,
                        wmediumd_mode=interference,
                        noise_th=-91, fading_cof=3)

    info("*** Creating nodes\n")
    ap1 = net.addAccessPoint('ap1', ssid='new-ssid', mode='a', channel='36',
                             position='15,30,0')
    net.addStation('sta1', mac='00:00:00:00:00:02', ip='10.0.0.1/8',
                  position='10,20,0')
    net.addStation('sta2', mac='00:00:00:00:00:03', ip='10.0.0.2/8',
                  position='20,50,0')
    net.addStation('sta3', mac='00:00:00:00:00:04', ip='10.0.0.3/8',
                  position='20,60,10')
    net.addStation('sta4', mac='00:00:00:00:00:05', ip='10.0.0.4/8',
                  position='20,70,10')
    c1 = net.addController('c1')

    info("*** Configuring Propagation Model\n")
    net.setPropagationModel(model="logDistance", exp=4)

    info("*** Configuring wifi nodes\n")
    net.configureWifiNodes()

    nodes = net.stations
    net.telemetry(nodes=nodes, single=True, data_type='rssi')

    info("*** Starting network\n")
    net.build()
    c1.start()
    ap1.start([c1])

    info("*** Running CLI\n")
    CLI(net)

    info("*** Stopping network\n")
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    topology()

```

- Code used for the movement of AR. Drone 2.0.

```
function sysCall_threadmain()  
    target=sim.getObjectAssociatedWithScript(sim.handle_self)  
    path=sim.getObjectHandle("Path")  
    sim.followPath(target,path,1,0,0.3,15)  
end
```

```
function sysCall_cleanup()
```

```
end
```