

Summer 8-1-2021

Discrete Moving Target Defense Application and Benchmarking in Software-Defined Networking

charan gudla

University of Southern Mississippi, School of Computing

Follow this and additional works at: <https://aquila.usm.edu/dissertations>

Recommended Citation

gudla, charan, "Discrete Moving Target Defense Application and Benchmarking in Software-Defined Networking" (2021). *Dissertations*. 1927.

<https://aquila.usm.edu/dissertations/1927>

This Dissertation is brought to you for free and open access by The Aquila Digital Community. It has been accepted for inclusion in Dissertations by an authorized administrator of The Aquila Digital Community. For more information, please contact Joshua.Cromwell@usm.edu.

DISCRETE MOVING TARGET DEFENSE APPLICATION AND BENCHMARKING
IN SOFTWARE-DEFINED NETWORKING

by

Charan Gudla

A Dissertation
Submitted to the Graduate School,
the College of Arts and Sciences
and the School of Computing Sciences and Computer Engineering
at The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved by:

Dr. Andrew H. Sung, Committee Chair
Dr. Dia Ali
Dr. Sungwook Lee
Dr. Ramakalavathi Marapareddy
Dr. Ras B. Pandey

August 2021

COPYRIGHT BY

Charan Gudla

2021

Published by the Graduate School



THE UNIVERSITY OF
SOUTHERN
MISSISSIPPI®

ABSTRACT

Moving Target Defense is a technique focused on disrupting certain phases of a cyber-attack. The static nature of the existing networks gives the adversaries an adequate amount of time to gather enough data concerning the target and succeed in mounting an attack. The random host address mutation is a well-known MTD technique that hides the actual IP address from external scanners. When the host establishes a session of transmitting or receiving data, due to mutation interval, the session is interrupted, leading to the host's unavailability. Moving the network configuration creates overhead on the controller and additional switching costs resulting in latency, poor performance, packet loss, and jitter.

In this dissertation, we proposed a novel discrete MTD technique in software-defined networking (SDN) to individualize the mutation interval for each host. The host IP address is changed at different intervals to avoid the termination of the existing sessions and to increase complexity in understanding mutation intervals for the attacker. We use the flow statistics of each host to determine if the host is in a session of transmitting or receiving data. Individualizing the mutation interval of each host enhances the defender game strategy making it complex in determining the pattern of mutation interval. Since the mutation of the host address is achieved using a pool of virtual (temporary) host addresses, a subnet game strategy is introduced to increase complexity in determining the network topology. A benchmarking framework is developed to measure the performance, scalability, and reliability of the MTD network with the traditional network. The analysis shows the discrete MTD network outperforms the random MTD network in all tests.

ACKNOWLEDGMENTS

I want to thank and express my deepest gratitude to my supervisor, Dr. Andrew H. Sung. You are the best advisor and mentor I have ever met, and thank you for your excellent guidance, encouragement, and patience over the years. I want to thank Dr. Dia Ali, who has been a great mentor since my journey began at the University of Southern Mississippi. I want to thank other committee members Dr. Ramakalavathi Marapareddy, Dr. Ras B. Pandey, and Dr. Sungwook Lee, for their precious time and valuable suggestions and input in the dissertation.

I want to thank Md Shohel Rana, collaborator in my research work, and we had a great time working together. I want to thank my friends Trung T. Nguyen and Amartya Hatua for giving me support and motivation all the time.

I want to thank the School of Computing Sciences and Computer Engineering for its tremendous support during the last five years. I want to thank Ms. Chrissy Hudson and Ms. Sherry Smith for all their help in my journey.

Finally, I want to thank my family members and friends for the encouragement, support, and motivation towards this journey and the new beginning.

DEDICATION

This work is dedicated to my family and friends who are with me on this journey.

TABLE OF CONTENTS

ABSTRACT.....	ii
ACKNOWLEDGMENTS	iii
DEDICATION.....	iv
TABLE OF CONTENTS.....	v
LIST OF TABLES	ix
LIST OF ILLUSTRATIONS.....	x
LIST OF ABBREVIATIONS.....	xiv
CHAPTER I - INTRODUCTION	15
1.1 Motivation.....	15
1.2 Contributions.....	15
1.3 Dissertation Structure.....	17
CHAPTER II- BACKGROUND	18
2.1 Moving Target Defense	18
2.1.1 Moving Target Defense Techniques Categories.....	18
2.1.2 Cyber Kill Chain.....	20
2.2 Software-Defined Networking (SDN)	20
CHAPTER III– SOFTWARE AND NETWORKING TOOLS	22
3.1 SDN Architecture.....	22
3.2 OpenFlow.....	24

3.3	OpenFlow Switch.....	24
3.3.1	Flow Tables.....	24
3.4	RYU Controller.....	25
3.5	Mininet.....	26
CHAPTER IV – MOVING TARGET DEFENSE APPLICATION AND ANALYSIS..		28
4.1	MAC Address Mutation in Drones	28
4.1.1	Related Work	29
4.1.2	Hacking Techniques.....	29
4.1.2.1	Data Packet Capture.....	30
4.1.2.2	Denial-of-Service Attack	30
4.1.3	Cyber-Attack on Drones	32
4.1.4	Defense against cyber attacks	33
4.1.4.1	Wireless Network Encryption.....	36
4.1.4.2	Intrusion Detection System.....	36
4.1.4.3	Moving Target Defense	37
4.1.5	Configuration	38
4.1.6	Results.....	39
4.2	Random Host Address Mutation and Analysis in SDN.....	42
4.2.1	Related Work	43
4.2.2	Experimental Setup and MTD Application	47

4.2.3 TCP and UDP Traffic Analysis	51
4.3 Discrete Host Address Mutation and Analysis in SDN	57
4.3.1 Proposed Methodology	59
4.3.2 Architecture.....	62
4.3.3 Traffic Generation and Reconnaissance	64
4.3.4 Results and Analysis	65
4.4 Discrete Host Address Mutation with Subnet Game Strategy and Benchmarking	70
4.4.1 Subnet Game Strategy.....	70
4.4.2 Benchmarking and Analysis	71
4.4.2.1 Benchmarking Topology	71
4.4.2.2 Benchmarking Performance.....	74
4.4.2.2.1 Network Topology Discovery Time	74
4.4.2.2.2 Asynchronous Message Processing Time.....	76
4.4.2.2.3 Asynchronous Message Processing Rate.....	78
4.4.2.2.4 Reactive Path Provisioning Time.....	80
4.4.2.2.5 Reactive Path Provisioning Rate.....	82
4.4.2.2.6 Proactive Path Provisioning Time	84
4.4.2.2.7 Proactive Path Provisioning Rate.....	85
4.4.2.2.8 Network Topology Change Detection Time.....	87
4.4.2.3 Benchmarking Scalability	88

4.4.2.3.1 Control Sessions Capacity	88
4.4.2.3.2 Forwarding Table Capacity.....	90
4.4.2.4 Benchmarking Reliability	92
4.4.2.4.1 Controller Failover Time	92
4.4.2.4.2 Network Re-provisioning Time	93
CHAPTER V – CONCLUSION AND FUTURE WORK	96
REFERENCES	100

LIST OF TABLES

Table 4.1 Parameters	51
Table 4.2 Overheads	56
Table 4.3 Packet Loss	57
Table 4.4 Notations	62
Table 4.4 Notations (continued)	63
Table 4.5 Snort Filtered Events	67
Table 4.6 Topologies	72

LIST OF ILLUSTRATIONS

Figure 1.1 Moving Target Defense Categories	19
Figure 1.2 Cyber Kill Chain.....	20
Figure 2.1 SDN Concept.....	21
Figure 3.1 SDN Architecture and OpenFlow	23
Figure 3.2 OpenFlow Switch	25
Figure 3.3 RYU Controller	26
Figure 4.1 Denial of Service Attack	31
Figure 4.2 Man-in-the-Middle Attack.....	31
Figure 4.3 MAC Address Detection	32
Figure 4.4 Communication Link Before DoS Attack	33
Figure 4.5 Communication Link After DoS Attack.....	33
Figure 4.6 System Architecture	34
Figure 4.7 Base Station	34
Figure 4.8 Raspberry Pi	35
Figure 4.9 ROS Model.....	35
Figure 4.10 WPA Binaries in AR Drone	36
Figure 4.11 Kismet IDS	37
Figure 4.12 Moving Target Defense Model	37
Figure 4.13 MAC Address Mutation Model.....	39
Figure 4.14 Data Capture Attack	39
Figure 4.15 Kismet IDS Alerts	40
Figure 4.16 New MAC Address After Mutation 1	40

Figure 4.17 New MAC Address After Mutation 2	40
Figure 4.18 Unsuccessful Cyber-Attack	41
Figure 4.19 Navigational Data from the Drone	41
Figure 4.20 Acceleration and Velocity Plots	42
Figure 4.21 MTD Topology.....	48
Figure 4.22 Communication Between Hosts	48
Figure 4.23 Algorithm Ryu Controller	50
Figure 4.24 Test 1 TCP Bandwidth Results.....	52
Figure 4.25 Test 2 TCP Bandwidth Results.....	52
Figure 4.26 Test 3 TCP Bandwidth Results.....	53
Figure 4.27 Test 4 TCP Bandwidth Bidirectional Traffic Results.....	53
Figure 4.28 Test 5 UDP Bandwidth Results	54
Figure 4.29 Jitter Measurement	54
Figure 4.30 Network Topology with Two Hosts	59
Figure 4.31 Reconnaissance Using Kali	64
Figure 4.32 SGUIL for Packet Analysis	65
Figure 4.33 Nmap Scan for Live Hosts	65
Figure 4.34 Nmap Results.....	66
Figure 4.35 SNORT Session Results	66
Figure 4.36 SNORT Alerts	67
Figure 4.37 Hosts Mutations at the Same Time.....	68
Figure 4.38 Hosts Sessions	68
Figure 4.39 individualized Hosts Mutations.....	69

Figure 4.40 Private IP Addresses	70
Figure 4.41 Example Subnet Topology	71
Figure 4.42 Subnet Pools	71
Figure 4.43 Leaf-Spine Topology	72
Figure 4.44 Leaf-Spine Topology with Controller	73
Figure 4.45 Network Topology Discovery Time	75
Figure 4.46 Network Topology Discovery Time Benchmark	75
Figure 4.47 Asynchronous Message Processing Time	77
Figure 4.48 Asynchronous Message Processing Time Benchmark	77
Figure 4.49 Asynchronous Message Processing Rate	78
Figure 4.50 Asynchronous Message Processing Rate Benchmark	79
Figure 4.51 Reactive Path Provisioning Time	81
Figure 4.52 Reactive Path Provisioning Time Benchmark	81
Figure 4.53 Reactive Path Provisioning Rate	82
Figure 4.54 Reactive Path Provisioning Rate Benchmark	83
Figure 4.55 Proactive Path Provisioning Time	84
Figure 4.56 Proactive Path Provisioning Time Benchmark	85
Figure 4.57 Proactive Path Provisioning Rate	86
Figure 4.58 Proactive Path Provisioning Rate Benchmark	87
Figure 4.59 Network Topology Change Detection Time Benchmark	88
Figure 4.60 Control Sessions Capacity	89
Figure 4.61 Control Sessions Capacity Benchmark	89
Figure 4.62 Forwarding Table Capacity	90

Figure 4.63 Forwarding Table Capacity Benchmark.....	91
Figure 4.64 Controller Failover Time Benchmark	92
Figure 4.65 Network Re-provisioning Time Benchmark	93
Figure 4.66 Packet Loss	94

LIST OF ABBREVIATIONS

<i>API</i>	Application Program Interface
<i>ARP</i>	Address Resolution Protocol
<i>DDoS</i>	Distributed Denial of Service
<i>DoS</i>	Denial of Service
<i>IDS</i>	Intrusion Detection System
<i>MIMA</i>	Man-in-the-Middle Attack
<i>MTD</i>	Moving Target Defense
<i>ROS</i>	Robot Operating System
<i>SDN</i>	Software-Defined Networking
<i>TCP</i>	Transmission Control Protocol
<i>TLS</i>	Transport Layer Security
<i>UDP</i>	User Datagram Protocol
<i>UAV</i>	Unmanned Aerial Vehicle

CHAPTER I - INTRODUCTION

1.1 Motivation

Moving Target Defense (MTD) [1] protects the network by changing the configurations frequently. The attacker gathers the information of the network at the initial phase of the cyber kill chain [2]. The traditional network configuration does not change from time to time, and it is static. This static configuration will give the attacker sufficient time to gather network information. Once the attacker collects the data, there is no looking back because the static network does not change its configuration. MTD technique aims to disrupt the reconnaissance attacks on the network by changing the network configuration frequently. Even though the attacker collects the information of the network, the attacks fail because the network configuration will have changed.

Software-Defined Networking (SDN) [3] simplifies the need to have the physical infrastructure, and it allows to centralize the control plane to manage the whole network [4]. It separates the control plane from the network device's data plane. It also simplifies the network management, operational cost and enables the programmability of the controller.

1.2 Contributions

The research work contributions are as follows:

1. A MAC address dynamic mutation technique [5] is developed and implemented on the drone wireless network, including intrusion detection system and enhanced security with wireless network encryption. The drone's MAC address is used to launch a cyber-attack. The hacker collects the configuration of the drone network by scanning. Once the information is gathered, the hacker does not need to re-scan

since the wireless network of the drone is static. We developed an MTD technique to mutate the static MAC address of the drone frequently. Since the MAC address is changing from time to time, the attacker will be unsuccessful in launching an attack on the network using the network configuration information collected previously.

2. A random host address mutation technique [6] is developed in software-defined networking (SDN), and network analysis is shown. The mutation of the host IP addresses of all hosts is implemented at a random time in the network. The IP address mutation is a widely researched technique, and it provides network security by assigning a virtual IP address to the host at a frequent time interval. The mutation technique provides security to the network and also creates an overhead for changing the host's configuration by the SDN controller. We implemented the host address mutation technique, and the performance of the network is shown by benchmarking with the traditional network.
3. In software-defined networking, we have developed a new discrete host address mutation technique [7] to individualize the mutation interval of each host in the network. In random host address mutation, the IP addresses of all hosts are changed at the same interval, terminating the established session between the hosts deteriorating the network stability. To overcome this backlog, a discrete host address mutation is developed to individualize the mutation interval of each host. Individualizing the mutation interval of each host makes it complex for the attacker to figure out the pattern of the mutation interval. The mutation interval of each host

is based on the flow statistics of the host monitored by the controller. The controller changes the IP address of the host when there is no exchange of data.

4. We developed an IP subnet game strategy using private virtual IP addresses. The virtual IP addresses are selected randomly from the pool of different IP subnets to make it complex for the attacker to understand the network topology. A benchmarking framework is developed to measure the stability of the network in terms of performance, scalability, and reliability.

1.3 Dissertation Structure

The dissertation structure is as follows. In chapter I, the objectives of the research work, motivation, and contribution are outlined. Chapter II introduces the research topic and related work. In chapter III, we discuss the software and networking tools used in this research. Chapter IV discusses the proposed moving target defense techniques for network security, and benchmarking results are demonstrated. Finally, chapter V concludes the research objectives and outcomes along with the future works.

CHAPTER II- BACKGROUND

2.1 Moving Target Defense

MTD reactively changes the configuration of the system across multiple planes. The mutation of the attack surface increases complexity and uncertainty in understanding the system behavior leading to rising in scanning costs for the attacker. The MTD technique changes the static system into a dynamic, enhancing the security of the system. Reconnaissance [8] is the initial phase of the cyber kill chain [2].

The attacker gathers the system's data by scanning the network extensively using multiple hacking tools. If the system is static, the data collected by the attacker will be used to initiate the cyber-attack on the network. The operational cost for scanning the network is high, and when the system is static, the attacker avoids scanning the system multiple times. Since the MTD technique changes the attack surface [9] frequently, the attacker has to put the high cost in scanning the system numerous times.

The dynamic changes in the system configuration can evade these attacks since the attacker uses static system information. Even though the cyber-attack is launched on the system, the attack fails due to the system configuration change.

2.1.1 Moving Target Defense Techniques Categories

The categories [2] that moving target defense techniques can be applied are shown in Figure 1.1.

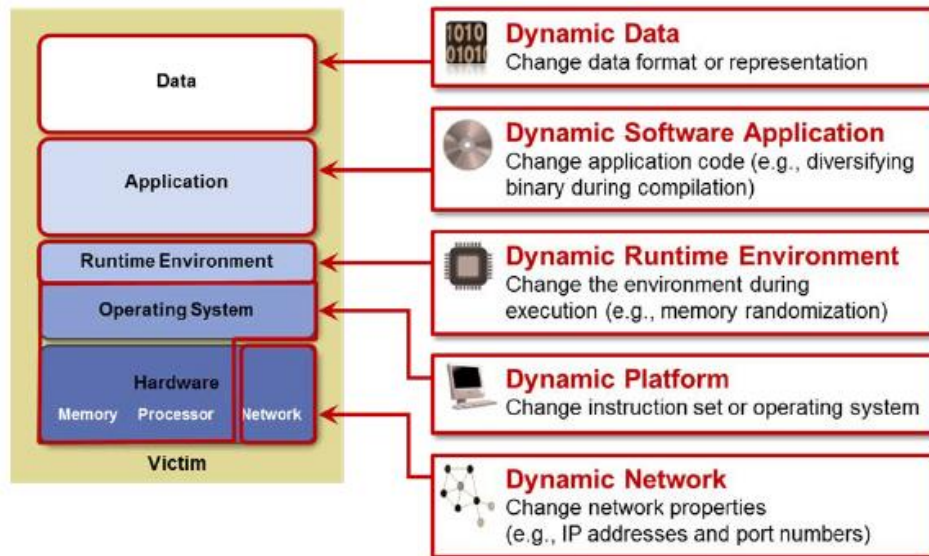


Figure 1.1 Moving Target Defense Categories

1. **Data Mutation:** The data format, encoding, representation, and syntax are changed dynamically.
2. **Software Application Code Mutation:** The application program code is changed dynamically by modifying instructions, grouping, format, and order.
3. **Runtime Environment:** The application environment is changed dynamically during execution.
 - (a) **Mutation of Address Space:** The memory layout where the program code is located is changed dynamically.
 - (b) **Mutation of Instruction Set:** The application's interface is changed dynamically.
4. **Platform Properties Mutation:** The mutation of the operating system's version, the architecture of the CPU, etc.
5. **Mutation of Network Properties:** The mutation of the network configuration of the system, which includes IP address, MAC address, and port number, etc.

2.1.2 Cyber Kill Chain

Cyber kill chain [2] refers to the series of steps followed by the attacker in successfully launching a cyber-attack, as shown in Figure 1.2. The following are the different phases of the cyber kill chain:

1. **Reconnaissance/Scanning:** This is the initial phase in which the attacker scans the network and collects the data.
2. **Access:** Using the gathered network information, the attacker identifies the network properties, configuration, and vulnerabilities of the network. The vulnerabilities identified helps the attacker to make the initial communication with the target.
3. **Exploit Development:** The vulnerability identified by the attacker is used to develop an exploit for privilege escalation.
4. **Attack Launch:** The cyber-attack on the network is launched on the target by delivering the exploit developed using phishing, USB drive, etc.
5. **Persistence:** The attacker should be persistent in the network to take over or control the network by creating and installing backdoors.



Figure 1.2 Cyber Kill Chain

2.2 Software-Defined Networking (SDN)

Software-defined networking [10] [11] separates the data plane and control plane of the network when compared to a traditional network, as shown in Figure 2.1. The controller in the control plane is centralized and takes the decisions of the network flow

from one point to another. The data plane transmits the data according to the path provided by the controller. The network devices forward the traffic to the controller if the network flow path is not available in the data plane. According to the destination address, the controller will decide the route, install the flow in the network devices, and use that flow for future data traffic forwarding. The controller uses the popular OpenFlow protocol [12] to communicate with network devices. The controller gives the flexibility by programmability to control and configure the network devices when required.

Even though the SDN architecture provides many advantages, limitations [13] also exist. In terms of reliability, if the controller fails [14], then the network will be down because of no control over the network devices and configuration. The backup controller will take over if configured. In SDN, the attackers will likely target the controller to take down the entire network until the backup controller becomes active. The controller has limited control session capacity, which is a drawback in terms of scalability.

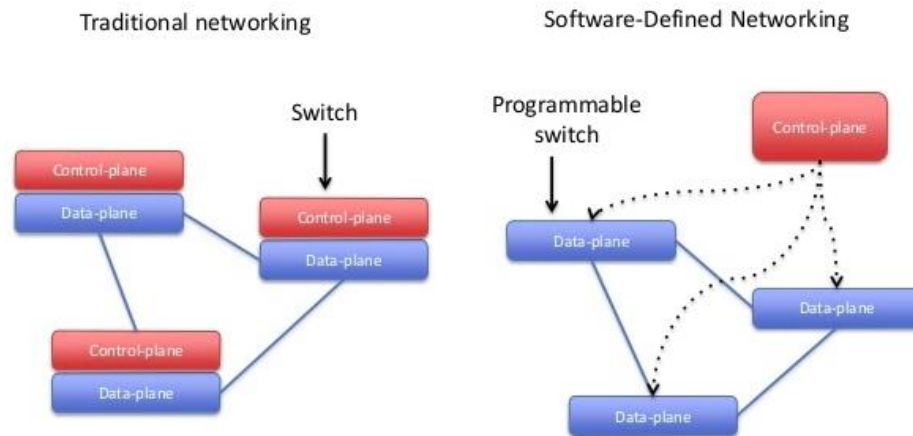


Figure 2.1 SDN Concept

CHAPTER III– SOFTWARE AND NETWORKING TOOLS

This chapter introduces the software tools used in this research: SDN, OpenFlow, RYU controller, and Mininet.

3.1 SDN Architecture

OpenFlow [12] is an open standard for a communications protocol that makes the control plane to decouple from the forwarding plane of numerous devices and communicate with it from a single point, allowing for more functionality and programmability.

Network devices, controllers, and applications are the essential components of SDN. Features for deciding incoming traffic forwarding are included in SDN devices. The SDN controller manages network devices and provides SDN applications with an abstraction of the network infrastructure at the southbound. The controller enables an SDN application to specify traffic flows and pathways on network devices in terms of common packet characteristics to meet its demands and to respond to changing user and traffic/network conditions. As shown in Figure 3.1, the Open Networking Foundation defines a high-level design for SDN [15] with three primary layers.

Infrastructure Layer. All the physical and virtual network device will be present in this layer. A packet-processing component, an abstraction layer, and an application program interface (API) at northbound for communication with the controller make up an SDN device. An SDN device is abstracted as a set of flow tables by the abstraction layer. By assessing incoming packets against flow table entries, the packet processing function determines forwarding action.

Control Layer. This layer provides logically centralized control capability that supervises network forwarding behavior. All SDN devices that make up the network architecture are controlled by an SDN controller, which uses a southbound API to implement policy decisions like routing, forwarding, and load balancing. Through a northbound interface, it gives apps an abstract view of the entire network.

Application Layer. End-user apps that employ SDN communications and network services make up this layer [16]. Applications can manage the underlying infrastructure behavior by adding flows to forward packets through the optimal path between the endpoints, load balancing across multiple paths or endpoints, reacting network topology changes such as the addition of new devices and paths, link failures, or redirecting traffic.

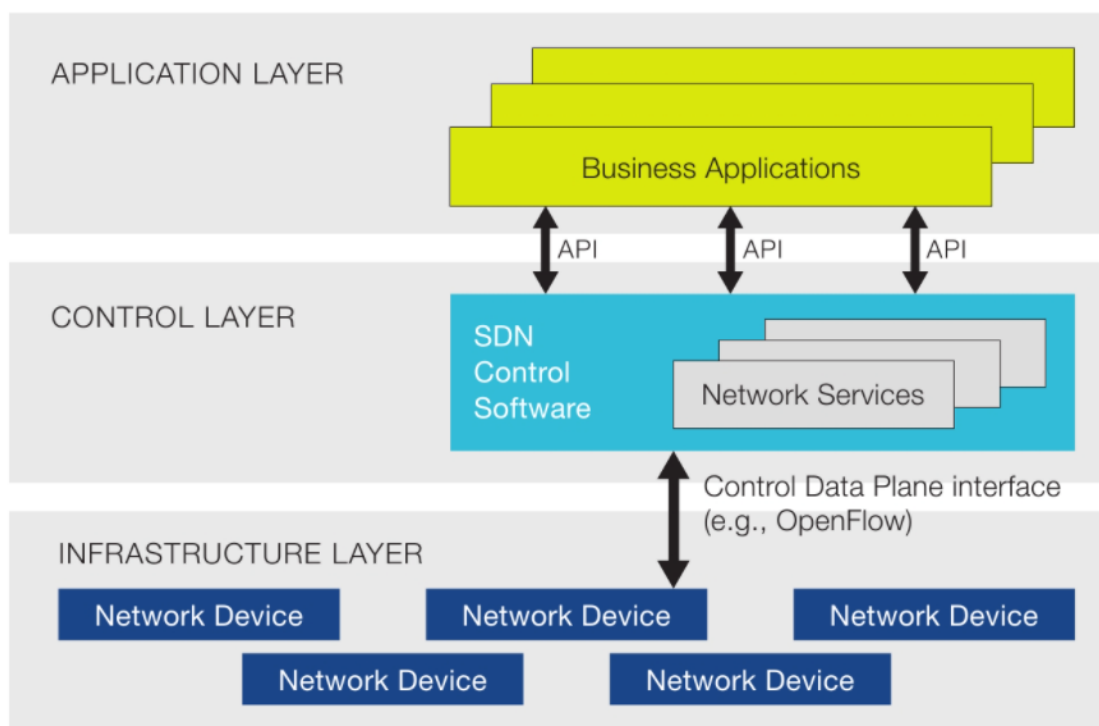


Figure 3.1 SDN Architecture and OpenFlow

3.2 OpenFlow

The southbound connection between an OpenFlow switch controller is defined by OpenFlow, a standardized protocol (Open Networking Foundation 2013). The communication messages between the two are sent over a secure channel, which is implemented over TCP using a Transport Layer Security (TLS) connection. The controller defines and programs the switch's packet forwarding behavior through the exchange of commands and packets. The switch then executes packet forwarding and reports its configuration status.

The characteristics of user traffic are used to classify it into flows. An OpenFlow switch monitors for packets and forwards them according to the flow to which they belong. A flow is a path in which packets are sent from one network endpoint (or group of endpoints) to another (or set of endpoints). Endpoints can be IP-TCP/UDP address pairs, VLAN endpoints, or switch input-output ports.

3.3 OpenFlow Switch

The OpenFlow logical switch contains at least one flow table and a group table to check the flow and forward data, as shown in Figure 3.2.

3.3.1 Flow Tables

The controller can add, delete, and update flow entries in flow tables both reactively (in reaction to packets) and proactively (in advance) using the OpenFlow switch protocol.

Reactive Flow Entries. The path from one node to another is called a flow. A reactive flow entry is created in the flow table when a host tries to communicate with

another host, and if the path exists, then the controller reactively installs the flow in the flow table.

Proactive Flow Entries: The flow entry is installed in the flow table ahead of time before the hosts communicate with each other.

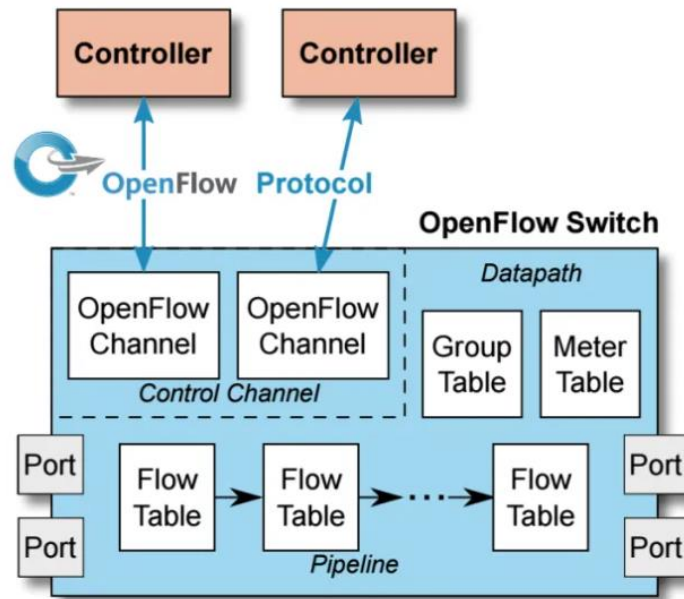


Figure 3.2 OpenFlow Switch

3.4 RYU Controller

Ryu is a Nippon Telegraph and Telephone Corporation Labs-sponsored open-source controller [17] written entirely in Python. It supports OpenFlow and connects with OpenStack. It is a centralized controller [18] with a simple API that creates new control applications and network management for network developers and operators. Components written in other programming languages can also be supported by Ryu. Ryu is popularly used for cloud infrastructures, data centers, and carrier networks. In communications, infrastructure services, event management, and application management, Ryu components

can be useful. NETCONF, OF-config, and OpenFlow 1.0 to 1.5 are among the network management protocols supported by Ryu, as shown in Figure 3.3.

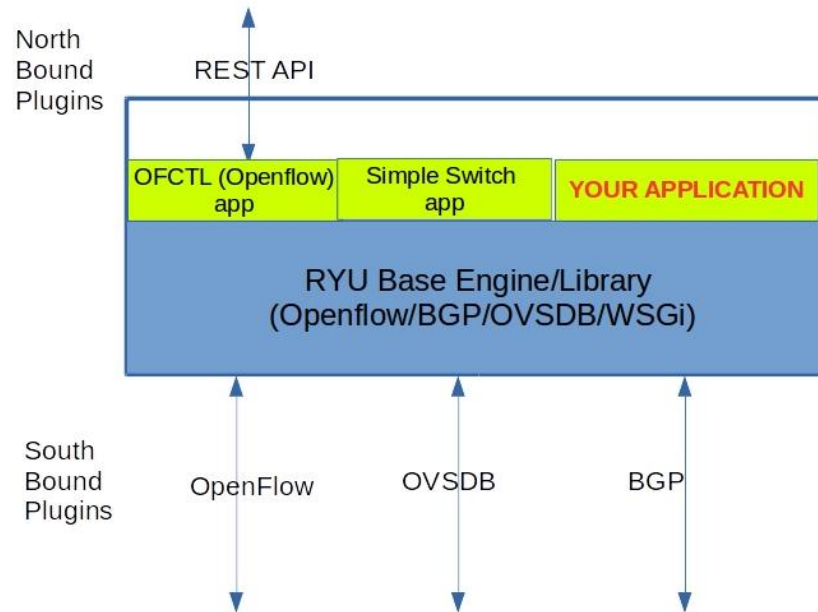


Figure 3.3 RYU Controller

3.5 Mininet

Mininet [19] is a network emulator, or more precisely, an orchestration framework for network emulation. On a single Linux kernel, it runs a collection of switches, end-hosts, routers, and links. It employs lightweight virtualization to make a single machine appear to be a full network with the same kernel, operating system, and user code. You can ssh into a Mininet host and execute any software you want. Your programs can send packets over what appears to be a real Ethernet interface, with a set link speed and latency. With a certain level of queueing, packets are processed by the switch, router, or middlebox. When two applications interact through Mininet, such as using an iPerf tool, the performance measured should be comparable to native machines.

In brief, Mininet's switches, virtual hosts, controllers, and connections are real, and they are made of software [20] rather than hardware and behave similarly to discrete hardware pieces for the most part. It is usually possible to construct a Mininet network that looks like a hardware network or, in reverse, a hardware network that looks like a Mininet network and runs the same apps and binary code on any platform.

CHAPTER IV – MOVING TARGET DEFENSE APPLICATION AND ANALYSIS

This chapter introduces the moving target defense techniques for network security. The application of these techniques not only provides security [21] but also creates overheads on the controller resulting in data loss. We proposed a new discrete host address mutation technique that can be more stable and reduce data loss.

4.1 MAC Address Mutation in Drones

The number of Unmanned Aerial Vehicles (UAVs), sometimes known as drones [22], is continuously increasing. Because they weigh less, cost less, and manageable, they are utilized in the military [23], aid in monitoring [24] [25], emergency disasters [26], and rescue operations [27]. In the telecommunications business [28], UAVs are employed to extend wireless network coverage. Amazon Prime Air [29] is a service from Amazon that will deliver products using drones.

Drones have many advantages, but they are vulnerable to physical difficulties and cyber-attacks. Satellites, cellular phones, Wi-Fi, GPS, and ZigBee, are all common ways to send and receive data via a network. In 2009, Iraqi rebels hacked into the feed [30] of the predator drone. In 2011, a virus attacked the networks utilized by US Air Force drone pilots at Air Force Base, Creech, Nevada [31]. Lockheed Martin RQ-170 Sentinel is an American drone [32] captured by an Iranian cyberwarfare outfit in 2011. Without the operator's knowledge, the predator drone video feeds were made public online [33]. Such attacks are carried out with the use of low-cost wireless network jammers and GPS spoofing devices.

Contributions. In this study, various vulnerabilities of UAVs and hacking techniques, as well as existing defense measures for countering cyber-attacks, are investigated. We built

a base station and used a popular hacking method on the UAV Parrot AR Drone to demonstrate the drone's weaknesses and exploitation. It demonstrates that an attacker can do significant harm by crashing drone or hack it and take control by compromising the wireless network between the operator and the drone. The experiment demonstrates the significance of protecting UAV systems against cyber-attacks.

4.1.1 Related Work

Various defense strategies against drone attacks have been presented. Nils Miro Rodday et al. proposed [34] using safe encryption techniques for Wi-Fi access in their paper. Johann Pleban et al. demonstrated [35] how to encrypt a wireless network using the drone as a client and the RC as an access point in their paper. To prevent an adversary from hacking into the drone, the open Wi-Fi network is encrypted with WPA. Chaitanya Rani et al. outlined [36] the flaws in encryption detection systems and proposed encryption detection systems as a protection strategy. A risk assessment scheme for communication infrastructure and services was created by Kim Hartmann and Christoph Steup [37]. The severity of a cyber-attack was assessed by James Goppert et al., who devised [38] a metric to reflect the system's period of complete failure. Robert Mitchell and Ing-Ray Chen [39] created a behavior rule-based UAV intrusion detection system in order to capture and continue harmful activity when a UAV is attacked.

4.1.2 Hacking Techniques

UAV wireless network attacking tactics are explored in this section. The results of our hacking experiment on the most popular drones are displayed below. When an attacker knows the drone's MAC address he wants to attack, he can gain access to its wireless network. Drone assaults on wireless networks can take the following forms:

- Data packet capture
- Denial of service (DoS) attack
- Man-in-the-middle attack (MIMA)

4.1.2.1 Data Packet Capture

The hacker uses a data packet capture method to collect the required information about the target. For example, the remote-control device controlling the drone, MAC addresses of the drone, wireless network channel, the encryption type (WEP/WPA/WPA2/OPN), etc., are sent out by the drone's wireless network, which can be collected. The tools used to capture wireless network frames are Aircrack-ng and Wireshark.

4.1.2.2 Denial-of-Service Attack

De-authentication flood attacks (DoS) [41] compromise wireless network [40] access points. The targeted access point's RAM is depleted by continuous de-authentication requests sent by the hacker. As a result, the clients cannot reach the access point because there is no memory left to reconnect, leaving them without a connection. The DoS attack will target the MAC address access point to disconnect all the devices connected to it or target a specific MAC address (drone or remote controller) to disconnect it from the network. The DoS attack is shown in Figure. 4.1.

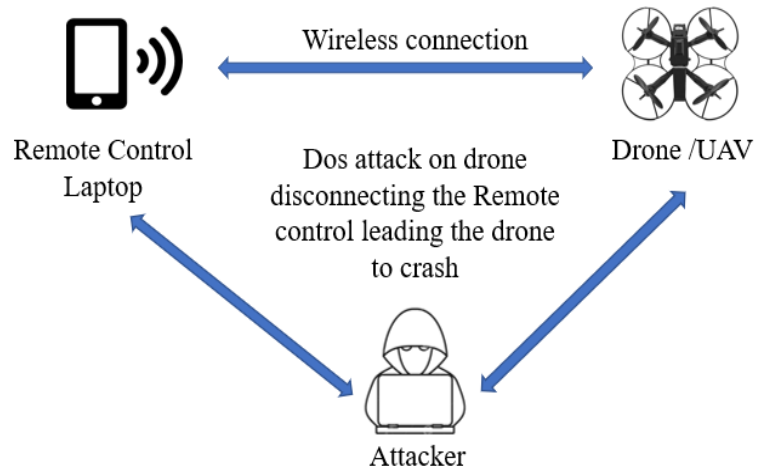


Figure 4.1 Denial of Service Attack

4.1.2.3 Man-in-the-Middle Attack

The attacker spoofs the communication network between the drone and the operator of the remote-control device (RCD) and takes control of it. He can transmit the authentication commands to the drone as if he were the original RC user as shown in Figure 4.2. The hacker will be able to see the location and drone's data feed without the drone's or the RC user's knowledge. If the wireless network is password-protected, Aircrack-ng and crunch programs can access the authentication keys through the handshake protocol.

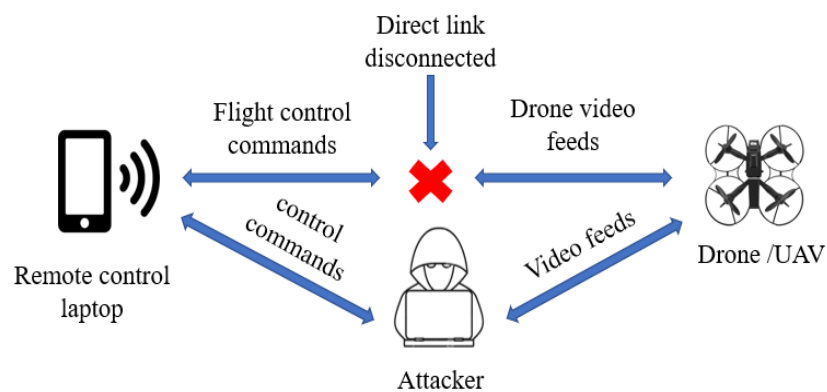


Figure 4.2 Man-in-the-Middle Attack

4.1.3 Cyber-Attack on Drones

On the drone static network, DoS attack is implemented on Parrot A.R drone. By repeatedly transmitting de-authentication commands, the remote-control device is unplugged from the drone. The drone will either crash or the attacker will be able to take control of it by attaching it to his device. A wireless bridge adapter Alfa AWUS036NHA is utilized in a virtual computer running Kali Linux. Aircrack-ng [42] is a toolkit that includes everything needed to take down a drone. The commands for attacking the drone are as follows:

```
# iwconfig wlan0 mode monitor
```

```
# ifconfig up
```

```
# aireplay-ng -9 wlan0
```

```
# airodump-ng wlan0
```

BSSID	PWR	Beacons	#Data	#/s	CH	MB	ENC	CIPHER	AUTH	ESSID
A0:14:3D:F8:07:0B	-35	21	420	0	1	54e	OPN			ardrone2
00:11:A5:1C:07:FA	-56	26	1	0	6	54	WPA2	TKIP	PSK	
00:02:6F:E6:1D:04	-81	15	70	0	11	54e	WPA	CCMP	PSK	
60:02:92:E0:7A:18	-90	1	0	0	1	54e	WPA2	CCMP	PSK	CBCI-B827-2.4
60:02:92:E0:7A:1B	-90	0	0	0	1	54e	WPA2	CCMP	MGT	XFINITY
60:02:92:E0:7A:1A	-91	1	0	0	1	54e	OPN			xfinitywifi

BSSID	STATION	PWR	Rate	Lost	Frames	Probe
(not associated)	B4:F7:A1:D0:9B:80	-69	0 - 1	21	3	
(not associated)	5C:AF:06:62:BE:FE	-91	0 - 1	0	1	
A0:14:3D:F8:07:0B	08:E6:89:35:F4:04	-25	0e- 0e	0	421	
00:11:A5:1C:07:FA	00:1E:04:FA:00:BB	-1	54 - 0	0	1	

Figure 4.3 MAC Address Detection

A data capture attack on a wireless network is launched using the above commands, resulting in collecting beacon frames containing destination and source MAC addresses of the devices in the network. The MAC address of the drone and the controlling device commanding the drone are displayed in Figure 4.3.

```
root@kali: ~# aireplay-ng -0 0 -a drone BSSID -c remotecontrol BSSID wlan0
```

The drone will be disconnected from the network leading it to crash or take over by the attacker, as shown in Figure 4.4 and Figure 4.5.



Figure 4.4 Communication Link Before DoS Attack



Figure 4.5 Communication Link After DoS Attack

4.1.4 Defense against cyber attacks

We proposed the following defense techniques for the drone's wireless network security.

- Wireless network encryption

- Intrusion detection system (IDS)
- Moving target defense (MTD)

Figure 4.6 and Figure 4.7 shows the architecture and base station model.

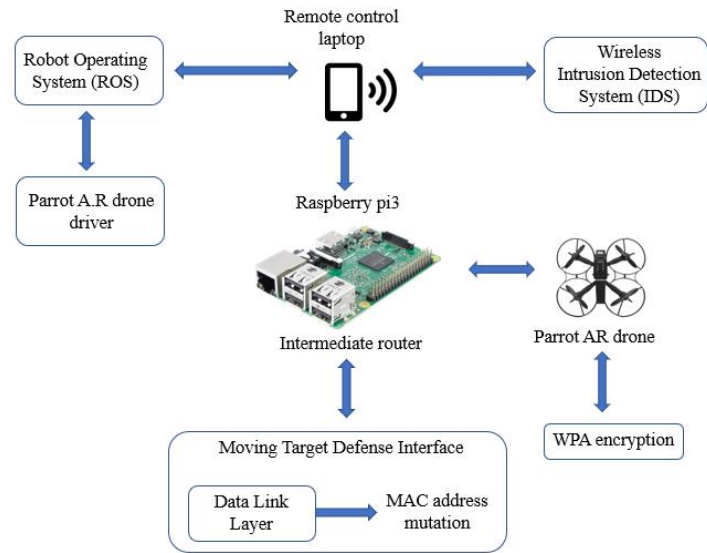


Figure 4.6 System Architecture

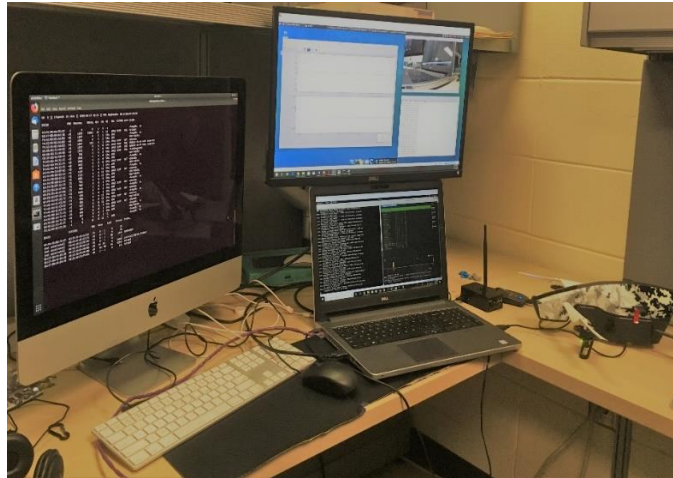


Figure 4.7 Base Station

The Raspberry Pi, as shown in Figure 4.8 is a low-cost computer that may be used for a variety of projects. We are utilizing it as an intermediate router [43] to create a secure wireless network between the drone and the remote control. It is set up to function as a

hotspot, bringing devices into the network and establishing a communication link between them. The drone sends live video feed to the laptop via raspberry pi router and controlling commands to the drone are send using laptop via raspberry pi router. WPA2 encryption is used to secure the raspberry pi wireless network.



Figure 4.8 Raspberry Pi

The Robot Operating System (ROS) in Figure 4.9 contains tools and libraries to develop reliable robotic applications. As part of this, ROS includes an AR drone driver that can communicate and control the drone. We can create autonomous tasks for the drone using ROS [44].

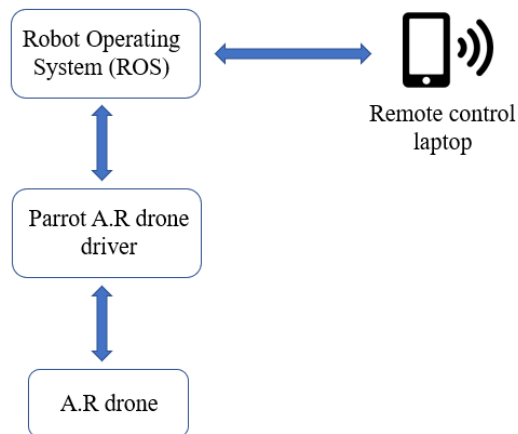


Figure 4.9 ROS Model

4.1.4.1 Wireless Network Encryption

Multiple devices can connect to the AR drone because it will operate as it's network is unencrypted and open, and it acts as an access point, but only one device can control it. The drone is compromised when the real user is disconnected, and the false user reconnects to the drone. Installing the WPA supplicant compiled libraries [45] into the drone libraries encrypts the drone's wireless network with WPA2 security, as shown in Figure 4.10. To achieve this, the drone's bin folder should have the binaries WPA CLI, WPA pass, and WPA supplicant. After the binaries have been installed successfully, the drone will stop functioning as an access point and connect to the specified access point name and password (in our case, it will connect to the raspberry pi).

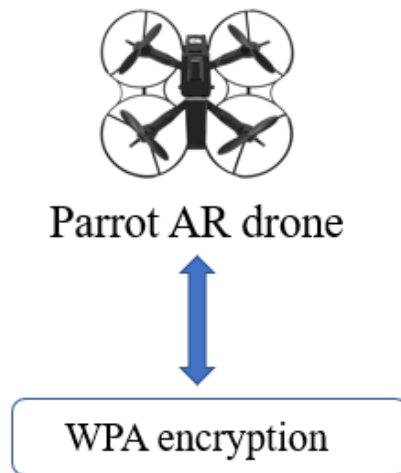


Figure 4.10 WPA Binaries in AR Drone

4.1.4.2 Intrusion Detection System

The wireless network is monitored in real-time by an intrusion detection system. Intrusion is defined as unauthorized access to a network without the knowledge of the network administrator. The systems can be hacked or spoofed, giving the unauthorized person immediate access.

IDS monitors the network but does not prevent cyber-attacks. It detects the anomalies in the network and notifies the user in case of any suspicious activity. We used Kismet wireless IDS [46] for network monitoring, as shown in Figure 4.11.

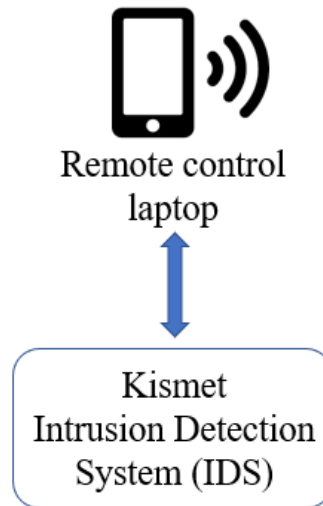


Figure 4.11 Kismet IDS

4.1.4.3 Moving Target Defense

The MAC address mutation technique is implemented in raspberry pi. Figure. 4. 12 shows the MTD model.

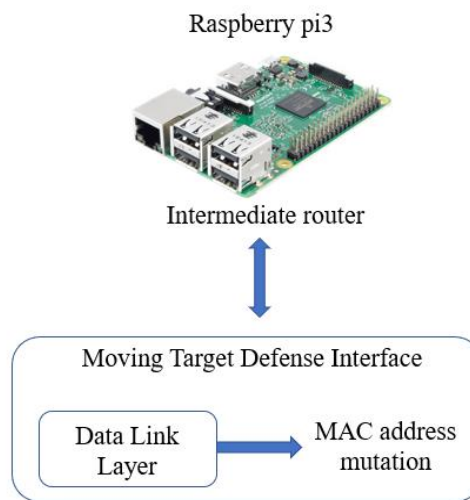


Figure 4.12 Moving Target Defense Model

4.1.5 Configuration

Kismet configuration file is created to monitor and detect malicious activities on the network. The alerts configured are shown below:

```
#kismet.conf

alert=LUCENTTEST,10/min,1/sec alert=DEAUTHFLOOD,10/min,2/sec

alert=NETSTUMBLER,10/min,1/sec alert=WELLENREITER,10/min,1/sec

alert=AIRJACKSSID,5/min,1/sec alert=PROBENOJOIN,10/min,1/sec

alert=BCASTDISCON,10/min,2/sec alert=CHANCHANGE,5/min,1/sec

alert=BSSTIMESTAMP,10/min,1/sec alert=MSFBCOMSSID,10/min,1/sec

alert=DISASSOCTRAFFIC,10/min,1/sec alert=NULLPROBERESP,10/min,1/sec

alert=MSFNETGEARBEACON,10/min,1/sec

alert=LONGSSID,10/min,1/sec alert=MSFDLINKRATE,10/min,1/sec

alert=DEAUTHCODEINVALID,10/min,1/sec

alert=DISCONCODEINVALID,10/min,1/sec

# Do we have a GPS?

gps=false

# Log file directory

configdir=/var/log/kismet/
```

The MAC address mutation technique is implemented using the macchanger tool libraries by executing the following script, and the model is shown in Figure 4.13.

```
#!/bin/bash

macchanger --show wlan0

Ifconfig wlan0 down
```

```

macchanger -r -b wlan0

Ifconfig wlan0 up

macchanger --show wlan0

sudo service network-manager start

```

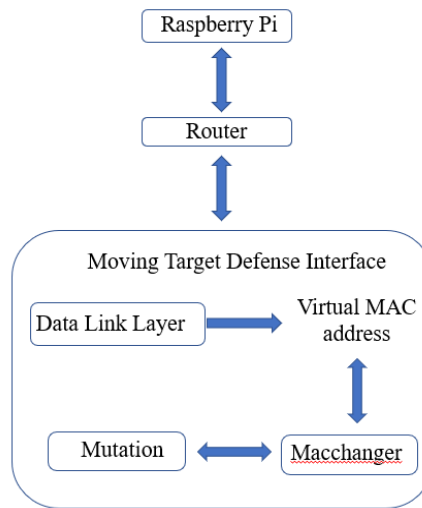


Figure 4.13 MAC Address Mutation Model

4.1.6 Results

We implemented Data capture attack and DoS attack on the wireless network of the drone. Kismet IDS detects the cyber-attack, and an alert is generated for the malicious activity on the network. Figure 4.14 shows the data captured by scanning the network, including MAC addresses and network encryption type. The MAC addresses under the station columns are the addresses of the drone and laptop controlling the drone.

CH 2][Elapsed: 36 s][2018-06-06 20:01											
BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID		
B8:27:EB:84:B4:3B	-33	21	77	0	1	54	WPA2	CCMP	PSK	hotspot	
BSSID	STATION		PWR	Rate	Lost	Frames	Probe				
B8:27:EB:84:B4:3B	A0:14:3D:F8:07:0B	-20	0	- 0e	49	7					
B8:27:EB:84:B4:3B	00:1E:64:FA:0D:BB	-28	54	- 6e	0	6					

Figure 4.14 Data Capture Attack

```
root@kali: ~# aireplay-ng -0 0 -a drone BSSID -c remotecontrol BSSID wlan0
```

The above command is used to implement DoS attack on the wireless network of the drone. The MAC address of the raspberry pi is targeted to take down the network and crash the drone. Figure 4.15 shows the DoS attack on the raspberry pi whose network is named as hotspot detected by the kismet IDS.

```
INFO: Detected new managed network "hotspot", BSSID B8:27:EB:84:B4:3B,  
encryption yes, channel 1, 72.20 mbit  
ALERT: BCASTDISCON Network BSSID B8:27:EB:84:B4:3B broadcast deauthenticate  
/disassociation of all clients, possible DoS  
ALERT: BCASTDISCON Network BSSID B8:27:EB:84:B4:3B broadcast deauthenticate  
/disassociation of all clients, possible DoS  
ALERT: BCASTDISCON Network BSSID B8:27:EB:84:B4:3B broadcast deauthenticate  
/disassociation of all clients, possible DoS  
ALERT: BCASTDISCON Network BSSID B8:27:EB:84:B4:3B broadcast deauthenticate  
/disassociation of all clients, possible DoS  
ALERT: BCASTDISCON Network BSSID B8:27:EB:84:B4:3B broadcast deauthenticate  
/disassociation of all clients, possible DoS
```

Figure 4.15 Kismet IDS Alerts

As soon as the alerts are generated by the IDS, if the mutation time interval is not reached, the MAC address of the hotspot is changed immediately to prevent the cyber-attack. Since the MAC address is changed, the attacker launching the attack with the old MAC address will be failed. Figure 4.16 shows the kismet detecting the new MAC address of the hotspot after mutation.

```
INFO: Detected new managed network "hotspot", BSSID FE:DA:B4:38:B3:EE,  
encryption yes, channel 1, 72.20 mbit
```

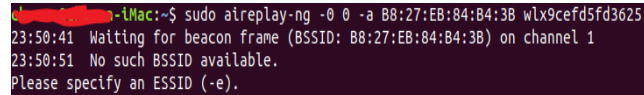
Figure 4.16 New MAC Address After Mutation 1

After a short interval, the MAC address will be changed again dynamically, and kismet detects the new MAC address of the drone in Figure 4.17.

```
INFO: Detected new managed network "hotspot", BSSID 2E:5F:6C:5B:2D:EE,  
encryption yes, channel 1, 72.20 mbit
```

Figure 4.17 New MAC Address After Mutation 2

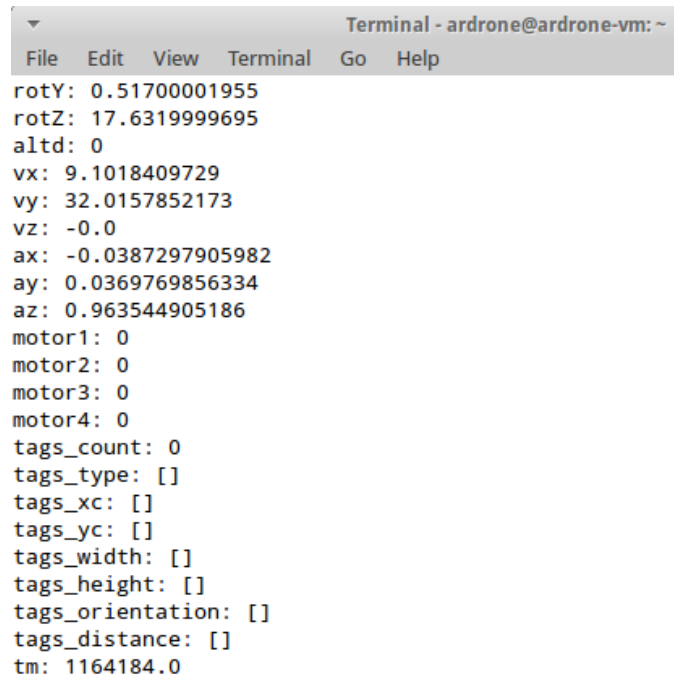
The attacker continues to implement cyber-attack on the wireless network using configuration gathered in data capture attack. Since the network is dynamic and the MAC address is changed dynamically, the attack on the network fails, as shown in Figure 4.18.



```
ardrone@ardrone-vm:~$ sudo aireplay-ng -0 0 -a B8:27:EB:84:B4:3B wlx9cefd5fd3625
23:50:41 Waiting for beacon frame (BSSID: B8:27:EB:84:B4:3B) on channel 1
23:50:51 No such BSSID available.
Please specify an ESSID (-e).
```

Figure 4.18 Unsuccessful Cyber-Attack

Some of the useful data transmitted from the drone to the base station are camera feeds, altitude, motor speeds, navigational data, acceleration, and velocity values which are shown in Figure 4.19 and Figure 4.20.



```
Terminal - ardrone@ardrone-vm: ~
File Edit View Terminal Go Help
rotY: 0.51700001955
rotZ: 17.6319999695
altd: 0
vx: 9.1018409729
vy: 32.0157852173
vz: -0.0
ax: -0.0387297905982
ay: 0.0369769856334
az: 0.963544905186
motor1: 0
motor2: 0
motor3: 0
motor4: 0
tags_count: 0
tags_type: []
tags_xc: []
tags_yc: []
tags_width: []
tags_height: []
tags_orientation: []
tags_distance: []
tm: 1164184.0
```

Figure 4.19 Navigational Data from the Drone

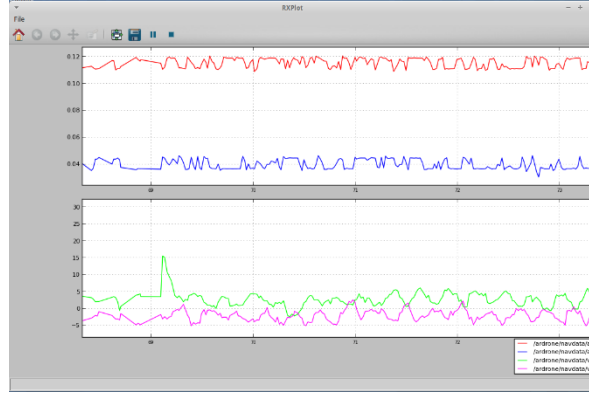


Figure 4.20 Acceleration and Velocity Plots

4.2 Random Host Address Mutation and Analysis in SDN

As discussed in sections 2.1 and 2.2, The MTD technique changes the static system into a dynamic, enhancing the system's security, and Software-defined networking separates the data plane and control plane of the network compared to a traditional network.

Contribution. We developed a random host address mutation technique in software-defined networking, and network analysis is shown. The mutation of the host IP addresses of all hosts is implemented at a random time in the network. The IP address mutation is a widely researched technique, and it provides network security by assigning a virtual IP address to the host at a time interval. Thus, the mutation technique not only provides security to the network but also it creates an overhead for changing the host's configuration by the SDN controller. We implemented the host address mutation technique, and the performance of the network is shown by benchmarking against the traditional network.

4.2.1 Related Work

The adversaries in Dynamic Network Address Translation (DYNAT) [47] [48] use most of their time monitoring the network. To prevent malicious scanning, DYNAT substitutes the TCP/IP header information. Trusted users are given predefined essential parameters that ensure the service's availability. The network overhead can be high depending on the fields obfuscated and deployment. Obfuscating the MAC address and deploying it on a switched network, for example, could cause the switches to overheat and drastically increase ARP traffic complexity to route switch port packets accordingly. To handle the routing overhead, more hardware may be necessary.

Revere [49] is a technique in which an open overlay is created. An overlay network is a dynamic network example that may change pathways, rearrange itself, and adapt to dynamically downed links or nodes. The additional traffic on the network is caused by the control messages transmitted between nodes. Unknown network overheads can be imposed by reconfiguration and routing.

RITAS [50] (Randomized Intrusion-Tolerant Asynchronous Services) is an acronym for Randomized Intrusion-Tolerant Asynchronous Services. On top of IPSec and TCP, it creates fault-tolerant consensus-based protocols. To run multiple resources, additional resources are needed and when the protocols are in the process of negotiation, additional time is required creating execution overhead. Running additional services than required due to mutation also created memory overhead. To each packet header an extra 24 bytes will be added by IPSec. As protocols come to an agreement, more network traffic is generated. An aggregate of 30% latency to each protocol is added by IPSec.

Antonatos et al. suggested Network Address Space Randomization (NASR) [51]. In order to detect worm attacks, endpoints that have been in the process of becoming infected or already infected are examined. DHCP changes the information of endpoints. Due to the mutation of IP addresses, connections are discarded or dropped, and network overhead is created.

A Mutable Network (MUTE) [52] allows network hosts to modify their host addresses and port numbers. The system configuration and real IP address are not changed to the current network it is a virtual overlay. Over the virtual relay, the traffic is routed independently. Host IP address data is synchronized through encrypted channels. An overhead on network infrastructure, including switches and routers, will be present. Additional routing overhead may cause network infrastructure to fail.

Inside a bigger outer virtual overlay network, Dynamic Backbone (DynaBone) [53] produces several inner virtual overlay networks. Inner networks improve diversity by using alternative networking, routing protocols, and hosting a distinct protocol or service. The hosts of the outer overlay network are unaware of the inner networks that make up one network. At the internal overlays, sensors are used to monitor traffic and performance. Depending on the routing protocols and networking utilized in the network, additional latency will be introduced. There is also a reduction in bandwidth. Overheads are caused by encryption and authentication protocols. There is no way of knowing how much network infrastructure and additional routing will be used.

Active Repositioning in Cyberspace for Synchronized Evasion (ARCSYNE) [54] is a mechanism that changes VPN gateway IP addresses using the gateway kernel OS. Using a clocking mechanism, while engaging in hopping, a secret is shared by the gateway. At each

clock tick, a virtual IP address using secret is generated by the gateway. Each gateway calculates the IP addresses of the other gateways in the same way. IP address hopping and streaming services have no effect on the gateways. For a grace period, the gateways allow data packets even after the IP address has changed. The time it takes for a data packet to go from one gateway to the next is the grace period. Updating the data packets address information has an impact on delivery delays, which is referred to as network overhead.

Random Host Mutation (RHM) [55] is a system that frequently modifies routable IP addresses. Temporary IP addresses are assigned to hosts that are mapped to actual IP addresses by RHM. The virtual IP addresses have a brief lifespan and are replaced at random and in a consistent manner. A special (MTG) gateway at the network edge converts the real IP address to a virtual IP address. Hostnames that DNS translates to the real IP address can be used to contact the IP address changing hosts, which was then converted to a virtual IP address before being sent to the originating hosts. Hosts can communicate with one other using their real IP addresses with the help of MT Controller permission. Sessions are kept open until the current flows are closed. TAP network kernel devices or OpenFlow virtual switches [56] are used for address mapping. For maintaining sessions during mutation, address-space overhead is created. Because of the frequent mutations that cause routing-update overhead, the routing table has grown in size.

OpenFlow Random Host Mutation (OF-RHM) [57] changes host IP addresses that are routable on a regular basis. The host's actual addresses are not modified. By ensuring consistency, the host's routable short-lived virtual IP addresses are established and regularly altered. RHM gateways and RHM controllers in OF-RHM are OpenFlow switches and OpenFlow controllers. An Open Flow controller converts to real IP addresses

from virtual IP addresses. It also uses OpenFlow messages across the switch to synchronize virtual IP mutations. End-host address assignments and DNS messages are controlled, as is the implementation of flow rules in the switches. By maintaining flows and assigning IP addresses to end-hosts, address-space overhead is created. The overhead increases as the mutation rate increases. The rate of mutation and flow termination also contributes to the flow-table overhead.

Spatio-temporal address mutation [58] dynamically changes host IP addresses, adding a layer of dynamicity to the IP address bindings and host. An overhead is imposed on the controller for the random mutation computation for each interval. Address space overhead is created for maintaining sessions during mutation. If the mutation rate is high, the overhead will be considerable. Queries delivered to the DNS server at shorter intervals also cause DNS traffic overhead.

Two security functionalities are included in the AVANT-GUARD SDN [59]. First, from saturation attacks, the control plane is protected by connection migration. The second, whenever from the traffic, if the attack is detected, flow rules are changed dynamically, protecting the data plane. The technique requires additional storage for the rules, and the overhead of assessments is added to the data plane. If the control plane connects with the data plane via various trigger and payload delivery reports, there may be network overhead.

DFI (Dynamic Flow Isolation) [60] adapts to changing network situation. The situation includes a time of day and a warning from third-party tools. The systems are subjected to DFI's network access policies. On the switches, flow rules are used to govern rate limits, ingress and egress flows from endpoints. Sensor data is processed using policy decision points (PDPs), and new flow rules are established in accordance with the present policy.

Latency is the time delay that happens when the switch sends new flow rules to the controller and DFI needs to make a decision. This overhead occurs for all new flow rules, regardless of size. If the new flow rule is present in the system, the latency will be reduced.

4.2.2 Experimental Setup and MTD Application

The topology in this study was developed using the Ryu controller in a Mininet emulator. The controller's dynamic algorithm frequently changes the network host configuration. The southbound interface communicates with hardware devices using the OpenFlow 1.3 protocol.

The Ryu Controller is in charge of DNS responses, IP address mutation, switch flow installation. In the terminal session, TCPDump is enabled to capture data packets of the host. The emulated topology is shown in Figure 4.21. The controller behavior is demonstrated by capturing and comparing the traffic from both topologies.

The communication steps, as shown in Figure 4.22, are as follows:

Step 1: Host1 sends a DNS request for Host 2 IP address.

Step 2: The controller intercepts the response from the DNS server, and it picks a random virtual IP address from the pool as v2 from the available IP address pool and sends it to Host1, establishing a mapping between the real and virtual IP addresses. The controller delivers the modified DNS response with the virtual IP address to Host1.

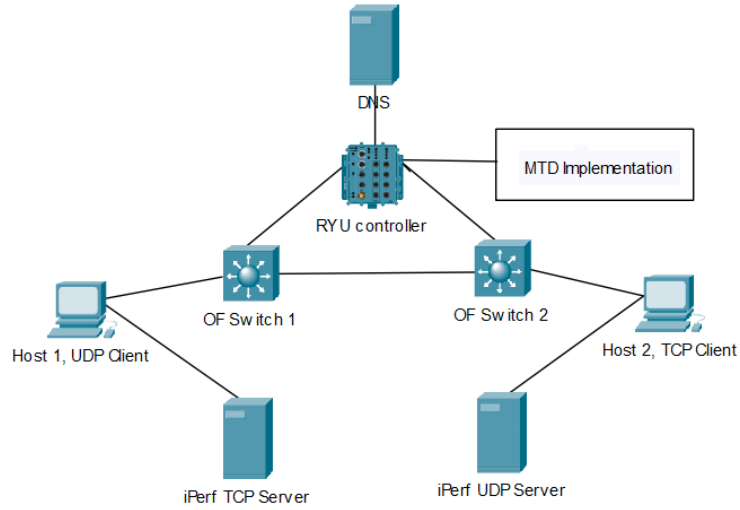


Figure 4.21 MTD Topology

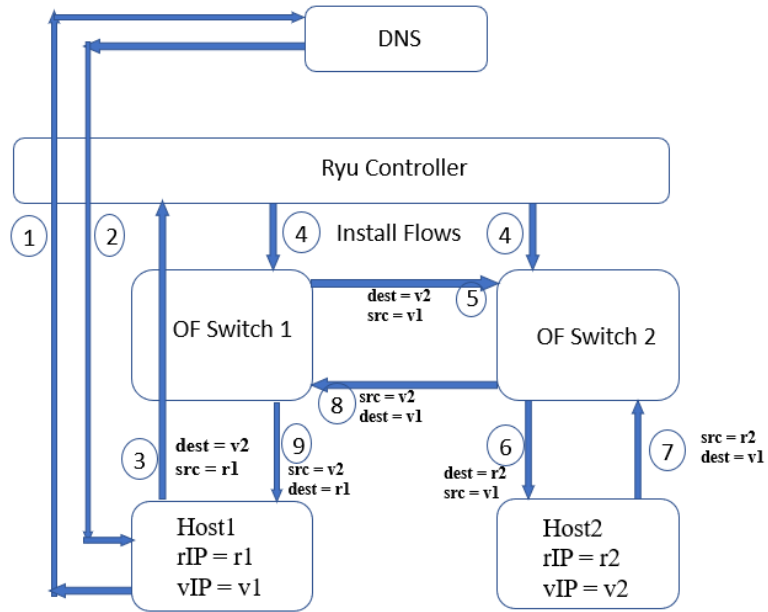


Figure 4.22 Communication Between Hosts

Step 3: Host1 delivers data packets to the virtual IP address v2 that it has received, with the source IP being its real IP address r1. Because switch1's table does not match the destination IP address, the packets are sent to the controller.

Step 4: The source and destination IP addresses are checked by the controller. If the source IP address is valid, it constructs a mapping by selecting a virtual IP address v1 from the pool at random. The mapping table will be used to check the destination IP address, and if it is identified, it will check the real IP address host and install the appropriate flows in the switches that can reach the destination host.

Step 5 & 6: The data packets are sent to Host2 in accordance with the flows. Because Host2 is directly connected to switch2, the controller will alter the destination IP to r2 to route packets to Host2 because switch2 is unaware of Host2's virtual IP address.

Step 7: When Host2 responds to the data packets, the source IP address becomes real IP address r2, and the destination IP address becomes v1. The controller transforms the physical address of r2 to the virtual address of v2.

Step 8 & 9: According to the flows, data packets are delivered to Host1. Because Host1 is physically linked to switch1, the controller will alter the destination IP to r1 to route packets to Host1 because switch1 is unaware of Host1's virtual IP address.

The real IP addresses are concealed, and the virtual IP addresses are visible to the outside world. The virtual IP addresses are regularly altered in a dynamic network to increase the attacker's cost and complexity when attempting to gather data using host addresses. Changing the IP addresses frequently increases the overhead in the system, resulting in packet loss. The controller must install switch flows for every IP address change, resulting in increased overhead, reaction time delays, and packet loss. Configuring two topologies allows for network traffic analysis.

The controller installs flows in the switches depending on the destination host IP address, which can be real or virtual. Host 1 is attached to Switch 1, and host 2 is attached to Switch 2. The source IP address is translated to a virtual IP address, and the traffic is forwarded to the destination virtual host IP address. The MTD implementation algorithm is shown in Figure 4.23.

A. Algorithm Ryu Controller

```

for all packets pkt from OF-Switches
  if pkt is DNS request
    set DNS response change rIP addr to vIP addr
    Map{rIP, vIP}
  if pkt is TCP, UDP
    if pkt.src is rIP
      pkt.srcIP = vIP addr
      Map{rIP, vIP}
    end if
    if pkt.dest is rIP connected to the switch directly
      then pkt.out to rIP address
    else
      install necessary flows in OF-switches
      towards destination
    end if
  end if
  for time interval t, mutate vIP of each host Hi
  end for
end for

```

Figure 4.23 Algorithm Ryu Controller

To generate traffic, the IPerf command-line utility is utilized. It is one of the most extensively used methods for assessing network performance. It is used in TCP and UDP connections to modulate parameters. IPerf should be set up so that one node serves as a server and the other serves as a client. Requests from the client to the server generate traffic,

which is also bidirectional. Packet loss, jitter and bandwidth are measured by generating TCP/UDP traffic.

The traffic is captured using the Command-line interface TCPDump for future investigation. TCPDump will collect traffic from both MTD-enabled and non-MTD topologies, saving it as a PCAP file for later examination.

4.2.3 TCP and UDP Traffic Analysis

The TCP UDP traffic captured by TCPDump is analyzed. For each testing session, TCPDump command line tool captures the traffic and saves as PCAP file. Packet loss, jitter and bandwidth are examined in this work. Table 4.1 shows test parameters. Using the analysis MTD network is benchmarked with traditional network.

Table 4.1 *Parameters*

Parameter (Both Topologies)	Value
Time for Test1 (TCP)	10 Sec
Time for Test2 (TCP)	20 Sec
Time for Test3 (TCP)	30 Sec
Time for Test4 (TCP Bidirectional)	10 Sec
Time for Test5 (UDP)	15 Sec

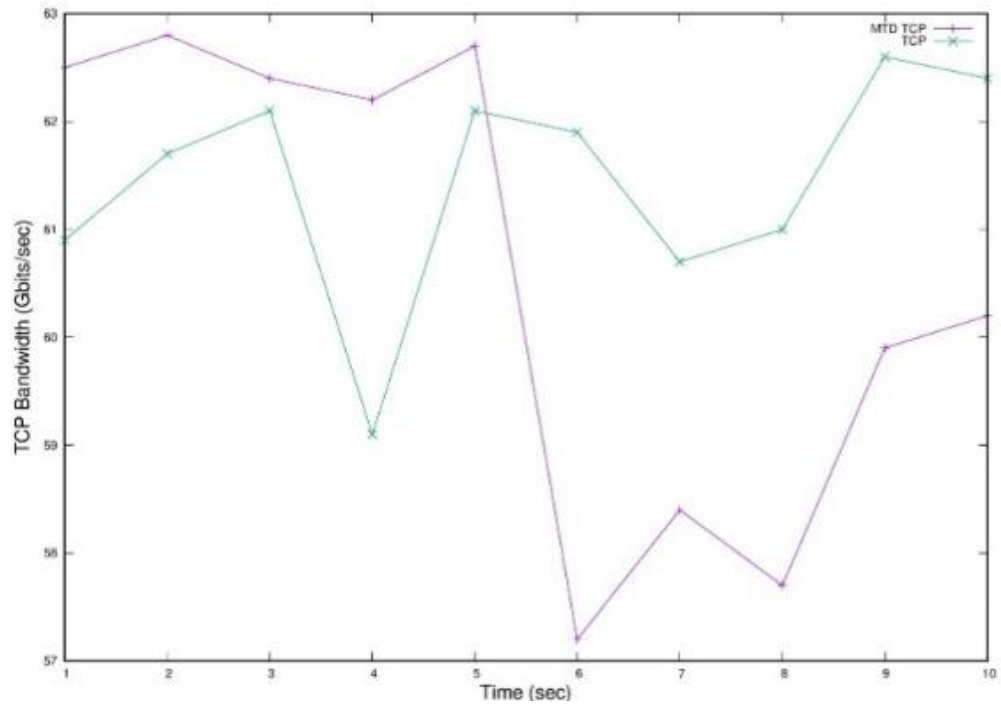


Figure 4.24 Test 1 TCP Bandwidth Results

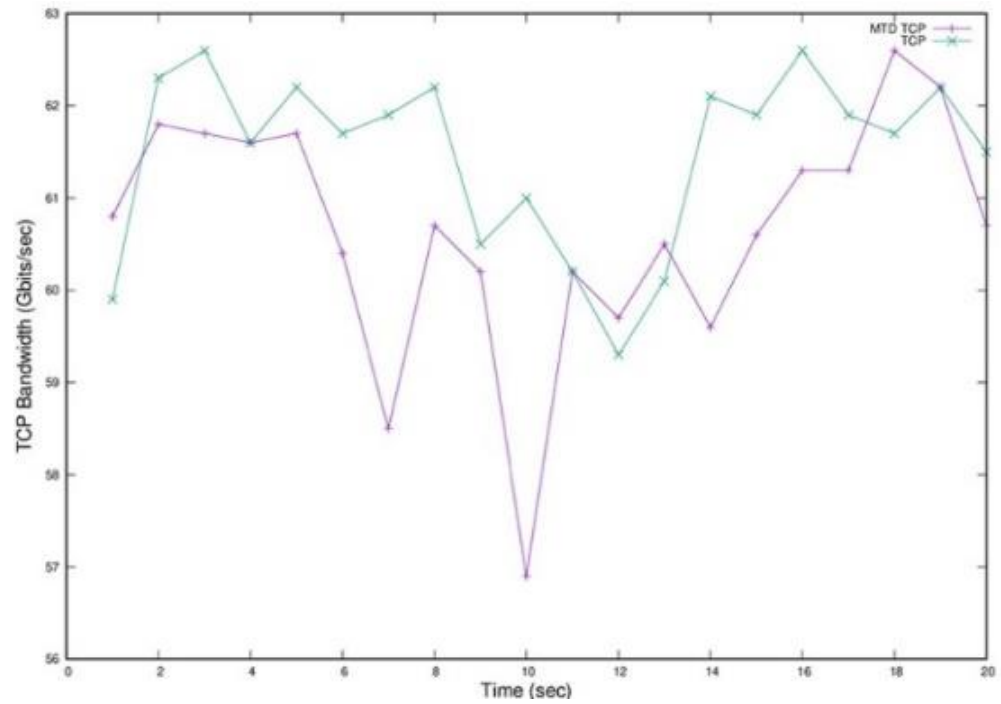


Figure 4.25 Test 2 TCP Bandwidth Results

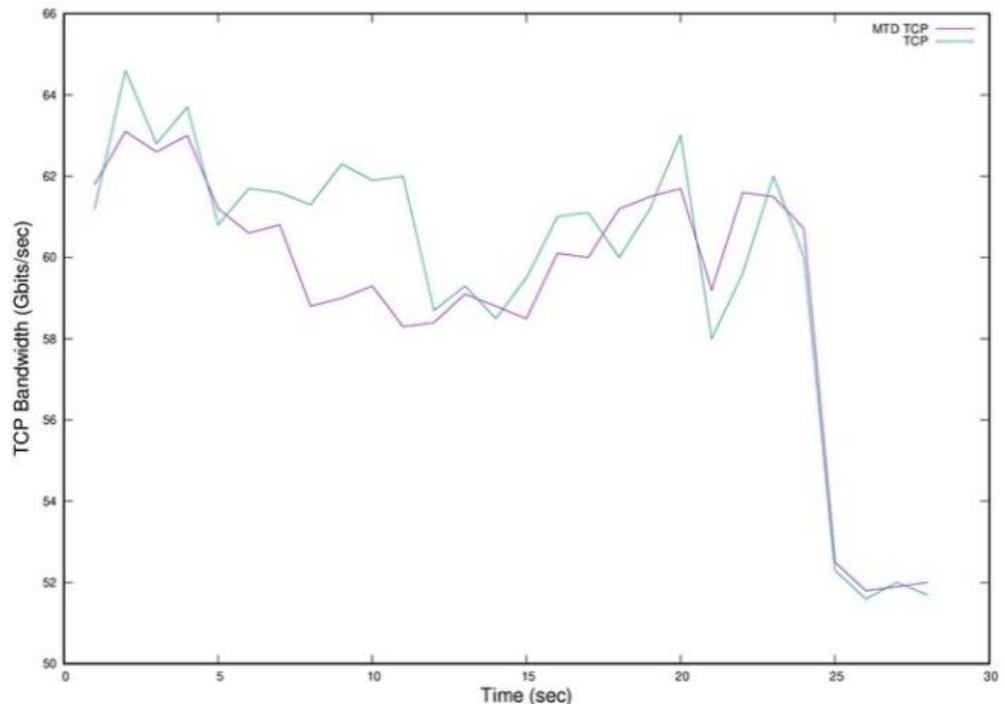


Figure 4.26 Test 3 TCP Bandwidth Results

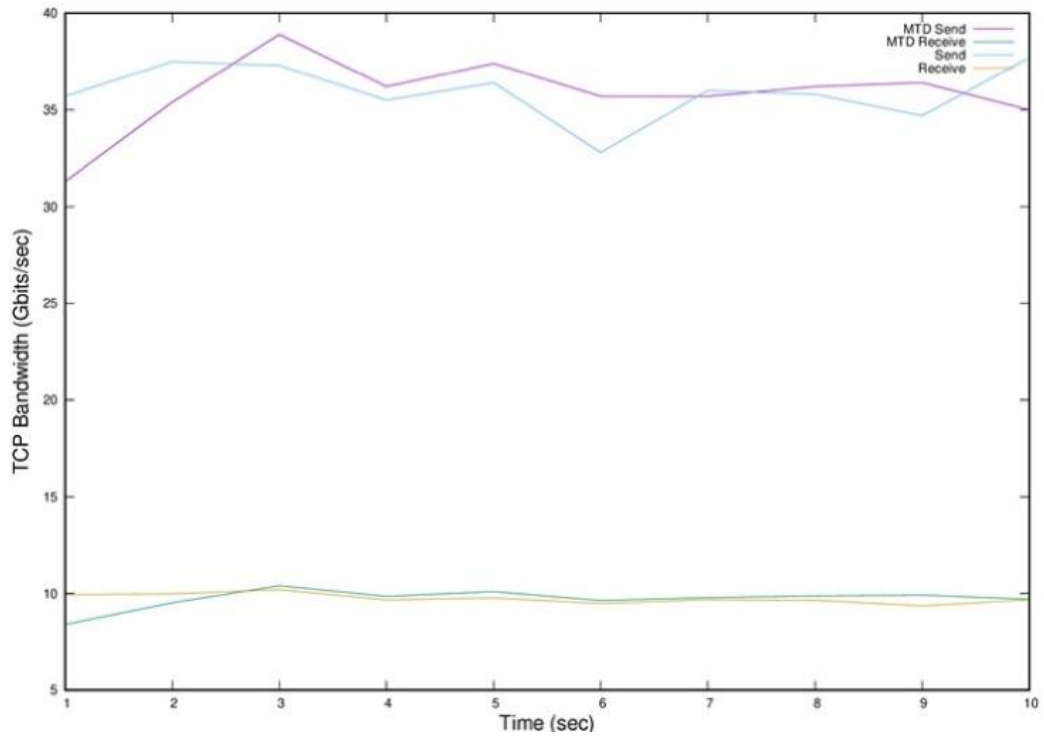


Figure 4.27 Test 4 TCP Bandwidth Bidirectional Traffic Results

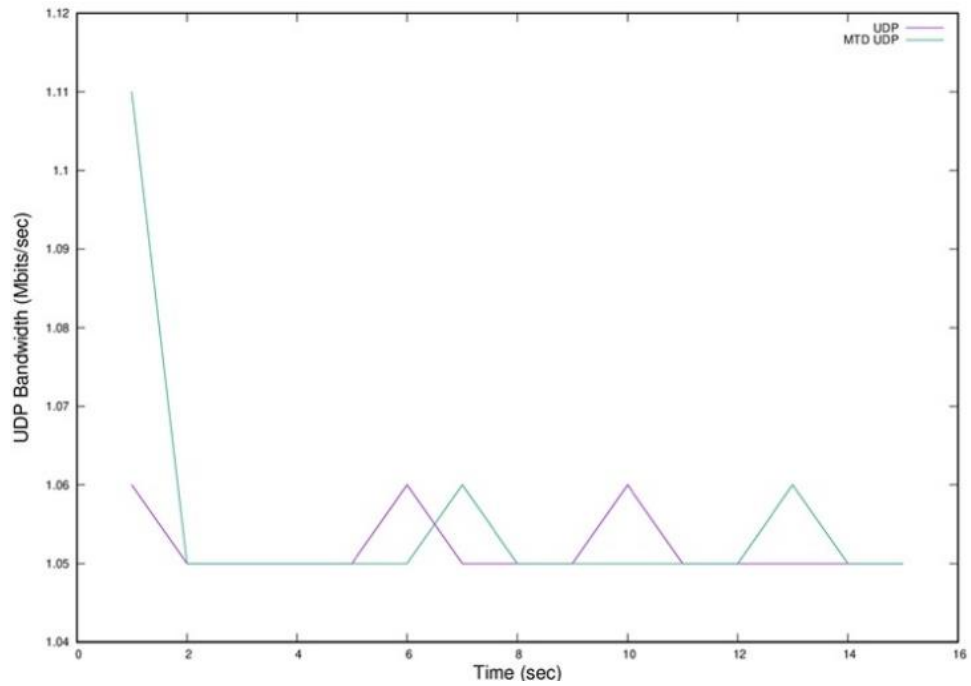


Figure 4.28 Test 5 UDP Bandwidth Results

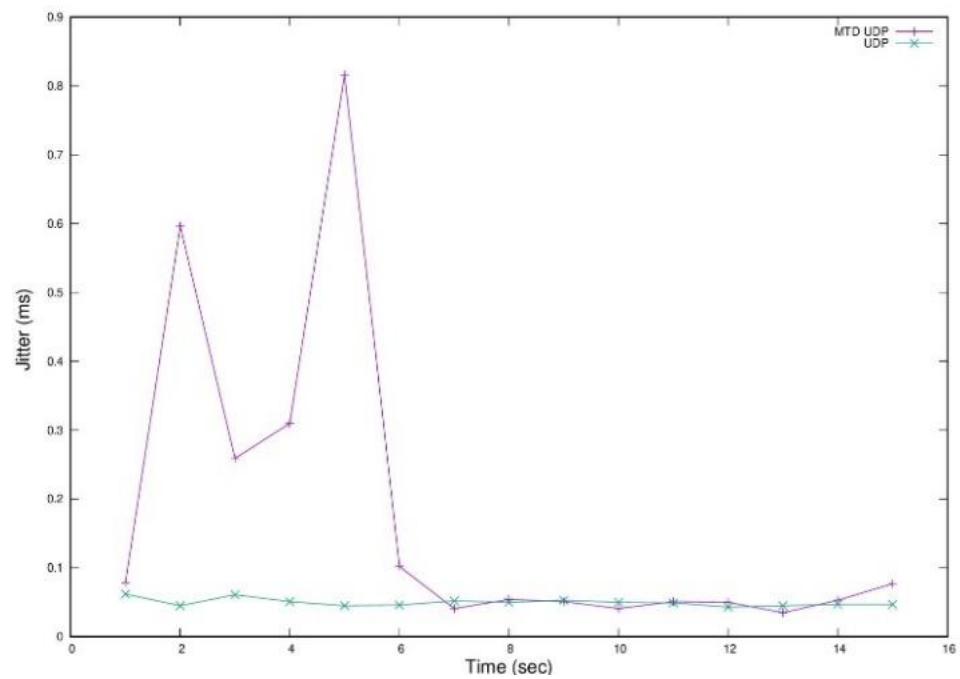


Figure 4.29 Jitter Measurement

The data transmitted in a period of time is known as bandwidth. Due to network disturbances, if the data packets are dropped in a TCP session, the bandwidth will be used more to re-transmit the dropped data. In a TCP session, the receiving endpoint will request the source to re-transmit the lost data using the sequence numbers. Since the IP address of the host is changed frequently, the data that is sent to the IP address before mutation will be dropped if the IP address is changed. The controller has to re-send the data packets dropped to the IP address after mutation.

The data from source travel in equal intervals to destination in a healthy network. The time interval between each data packet that is sent to the destination is called Jitter. The time interval between each packet will be disrupted if there are disturbances in the network. Jitter causes packet loss by causing congestion.

The bandwidth measurement is shown in Figures 4.24, 4.25, and 4.26, respectively, based on the aforesaid test findings. In traditional network topology, we can see that the bandwidth graph is normal. In the MTD network, as time increases, the bandwidth is decreasing. The mutation of the IP address is causing termination of the session at the same time, and the packets are getting dropped. In order to re-transmit the data to a new IP address, the bandwidth usage is increased, and new data transmission capacity is reduced. In both topologies, bidirectional traffic is generated, as seen in Figure 4.27. The traffic will be sent from both endpoints at the same time, and the bandwidths are measured for both sender and receiver. At the beginning and end of the test, the MTD architecture has somewhat less bandwidth than the traditional network.

The bandwidth analysis of the UDP session is shown in Figure 4.28. The data is transmitted normally in a traditional network without any packet loss, however, the MTD

topology bandwidth increased significantly at first and then decreased abruptly before returning to normal. The bandwidth usage is normal because UDP does not re-transmit the lost data packets.

The jitter analysis of UDP session traffic is shown in Figure 4.29. In typical network topology, jitter is expected. In a typical network, the time distribution between packets arriving at their destination is normal. If the jitter time interval fluctuates in a session, congestion occurs, resulting in packet loss. The IP mutation in the MTD network causes the controller to be burdened with the task of installing numerous flows in the switches in a short period of time. The network becomes unbalanced as a result of the configuration modifications, leading to congestion, packet loss, and other issues. Table 4.2 shows the resulting overheads created and Table 4.3 shows the packet loss.

Table 4.2 *Overheads*

Overhead	Description
Address-space	This overhead is created due to the frequent assignment of virtual IP addresses to the hosts. The controller requires additional space to map the sessions with IP addresses.
Flow-table size	As the IP addresses of the nodes are changed frequently, the number of flows in the table increases if the existing flows are not managed to delete on time.
Routing-update	As the new flow is added to the flow table, the updates should be sent to network devices.
Execution	The application running on the controller changes the IP address of the host frequently, resulting in execution overhead.

Table 4.3 *Packet Loss*

Test	Protocol	Topology	Packet Loss
1, 2, 3	TCP	Both	None
4	TCP	Both	None
5	UDP	Traditional	None
5	UDP	MTD	22%

4.3 Discrete Host Address Mutation and Analysis in SDN

In this research, we propose host address mutation [62] deployed as a novel MTD technique in the SDN environment, which aims to create high uncertainty in adversary scanning by changing the IP addresses of the host in the network based on individual mutation time intervals. The main objectives of this research are discussed in sequence. First, transparency is maintained in the mutation of the IP address of each host in the network. To provide transparency, the real IP (*rIP*) address of each host is unchanged, and a short-lived random virtual IP (*vIP*) address is assigned regularly to each real IP according to the mutation time interval.

Second, Once the session (S_{tcp} , S_{udp} , S_{icmp}) is established, the hosts are ready to transfer and receive the data from each other. The active session ($Act(h_i, h_k)$) mapping is created for each host, and it is monitored. Since the session time interval varies for each host, the mutation time interval also varies. Sometimes the session interval is long that adversaries can be successful in implementing a scanning attack. Even though the adversaries obtain the host's information, it is a virtual IP that will be changing rapidly. In this way, the mutation interval is different for every host, which does not need to change the address while in active session, preserving network performance and stability.

Third, we assume that every host in the network will not be given privileges to access sensitive data, modify network configuration, change firewall settings, and simply not even BIOS settings of the host itself in an enterprise network topology. In this situation, the hosts with administrative privileges are targeted more than the others. The scanning attacks on those specific host IP addresses will be high when compared to other host IP addresses. The host active in transferring and receiving data is targeted more than the host, which is less active; this attribute can also be added because the attackers are more likely to collect more useful data within a short period.

Fourth, to provide enough IP addresses to hosts in the network, the unused public address space range should be equivalent to the number of mutations in a host time interval. In a network, there will be hosts that can be reached publicly and need a public address range, and some hosts are internal to the network, which can be provided with a private address space range. The private address space range is always huge than the public address space. This mutation scheme using data stats is also effective when the available address space is less. Due to insufficient address space, some of the hosts cannot be moved, or the host address is repeated multiple times in a short interval of time. To avoid this problem, data stats are used to allocate the address space range for a host involved actively in the network and targeted by the attackers frequently.

To implement these techniques, traditional network implementation is costly and poses more challenges. We use software-defined Networking (SDN) infrastructure, which is quite flexible in developing and managing the network with minimal operational overhead. The network controller RYU is used to monitor and control the network using the OpenFlow 1.3 protocol. The network topology is built using Mininet. The experimental

results and analysis of the simulated network will show the significant rise in defending the network against reconnaissance attacks by increasing uncertainty in scanning, complexity in gathering the information about the network systems.

4.3.1 Proposed Methodology

In this moving target defense host address the mutation scheme. Consider a network topology with N_s number of hosts. Here we created the topology in Mininet, and the RYU controller is used. The controller implements the data flow between the network components. Figure 4.30 shows the topology with two hosts, and later we add two hosts.

$$N_s = \{h_1, h_2, h_3, h_4 \dots \dots h_n\} \quad (4.1)$$

The real Ip address (rIP) of each host in the network is replaced by a short-lived virtual IP address (vIP). The virtual IP addresses are assigned randomly from the pool of available unused address space. The mapping function f_{map} is used to assign a virtual IP address to a real IP address.

$$vIP = f_{map}(rIP) \quad (4.2)$$

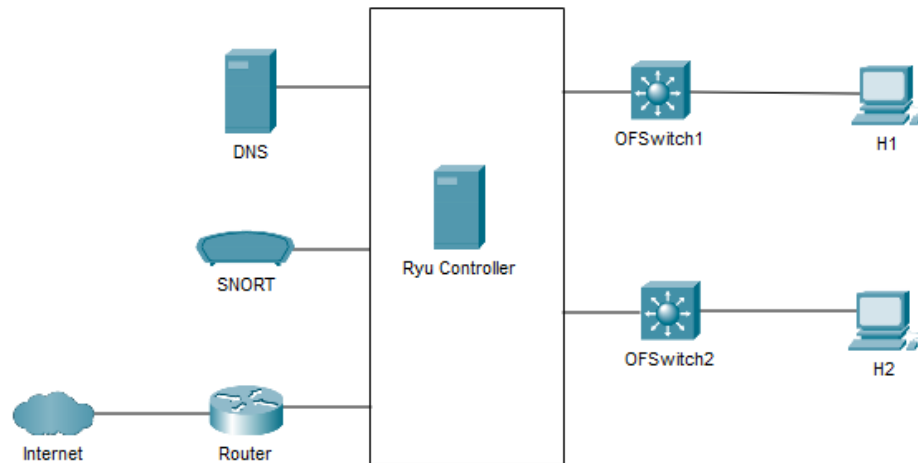


Figure 4.30 Network Topology with Two Hosts

When the host h_i initiates communication with host h_j , a DNS request will be sent to the server to resolve the domain name into the vIP address of the host h_j and a session $(S_{tcp}, S_{udp}, S_{icmp})$ will be established between host h_i and h_j by installing the flows in the required switches and active session mapping $Act(h_i, h_j)$ is created for monitoring the data stats. The controller will monitor the data stats for each session. Data stats include parameters like session interval, source IP, destination IP, source MAC, destination MAC, data packets sent, and received in that time interval, etc. Once the session is ended, it is removed from the active session table. If the host wants to communicate and establish a session again, the DNS request should be sent to the server to retrieve the vIP address of the host.

Since each session time interval (S_t) depends on the length of time taken by the hosts to complete the transfer of data between each other, the randomization is applied when the session is ended to provide network performance and stability. The mutation time interval M_t is the session time interval. If the mutation time interval is random, the session between the hosts will be interrupted, and the packets are dropped.

$$S_t = S_{(tcp, udp, icmp)}(Act(h_i, h_j)) \quad (4.3)$$

$$M_t = S_t \quad (4.4)$$

If the address space in network topology is N_s . The available active hosts in the network are N_a . The time taken for reconnaissance attack for each active host is $T_{r,h}$. The total average time $T_{a,r}$ that the adversary will spend on reconnaissance attack on all hosts will be:

$$T_{a,r} = N_a \times T_{r,h} \quad (4.5)$$

If $C_{r, h}$ is the cost to spend for reconnaissance attack on a single host. The total cost $C_{a, r}$ that the adversary needs to spend on reconnaissance on all active hosts will be:

$$C_{a, r} = N_a \times C_{r, h} \quad (4.6)$$

Fingerprinting is a technique used by the adversary to find the vulnerable host with high probability in the network by analyzing the data collected from the reconnaissance. Similar fingerprint operations will not be repeated on the host if it fails to find. The probability of identifying the vulnerable host in the network successfully with i steps will be:

$$P_i = 1 / N_a, 1 \leq i \leq N_a \quad (4.7)$$

If $T_{f, a}$ is the fingerprint time spent on the single host, then the average time taken by the adversary for reconnaissance and fingerprinting analyses to find the vulnerable host is defined as:

$$T_f = N_s \times T_{a, r} + ((N_a + 1) \times T_{f, a}) / 2 \quad (4.8)$$

From the above analysis if T_m is the mutation time interval, then:

$$T_m \leq (N_s \times T_{r, h} + T_{f, a}) \quad (4.9)$$

From the above analysis, for each T_m interval, the adversary cannot complete a reconnaissance attack. The adversaries will target the host, which is very active in transferring and receiving data. The adversaries tend to collect a large amount of traffic within a small time. The amount of available unused address space should be equivalent to the number of hosts in the network topology. If the address space in network topology is N_s and the available unused address space is $N_{u, s}$ then:

$$N_{u, s} \equiv N_s \quad (4.10)$$

If the available unused address space is less, then some of the hosts in the network cannot be moved or cannot be moved frequently according to the mutation time interval. To solve this problem, the hosts which are highly active in the network are identified. The adversaries try to gather the maximum amount of data within less time, which is possible when the host is highly active and can be targeted. The scanning attack stats and data packets stats can be monitored using SNORT, and it will alert the controller according to the rules written into it. Using SNORT rules, we can analyze the data packets and discover the hosts which are highly active in the network and targeted by the adversaries. In this way, the unused IP addresses can be used effectively. Ryu and SNORT can be configured on a single machine or different machines.

4.3.2 Architecture

The topology is implemented in the Mininet network using the Ryu controller. The topology can be seen in Fig. 2 in detail. Ryu controller acts as intermediate central software to manage network activities like IP mutations, DNS responses, Session establishment, IP address space management, Data stats analysis using SNORT.

Table 4.4 *Notations*

N_s	Network address space
N_a	Active hosts in the network
rIP	Real IP address
vIP	Short-lived virtual IP address
f_{map}	Real IP to virtual IP mapping function
S_{tcp}	TCP session between hosts
S_{udp}	UDP session between hosts
S_{icmp}	ICMP session between hosts
$Act(h_i, h_j)$	Active session mapping between host h_i and host h_j

Table 4.4 *Notations (continued)*

S_t	Session time interval
M_t	Mutation time interval
$T_{r,h}$	Time taken for reconnaissance attack on each active host in the network
$T_{a,r}$	Total average time spent on reconnaissance attack on all active hosts in the network
$C_{r,h}$	Cost to implement reconnaissance attack on a single host
$C_{a,r}$	The total cost of implementing reconnaissance attack on all active hosts
P_i	The probability of identifying vulnerable host in the network
$T_{f,a}$	Time spent for fingerprinting on a single host
T_f	Average fingerprinting time spent on all hosts in the network
$N_{u,s}$	Available unused address space
<i>SNORT</i>	Data packet sniffer, IDS, IPS

The unmatched packets in the OpenFlow switches will be encapsulated and sent to the Ryu controller. The controller discovers the type of packets, and required actions are taken. If the packet is DNS request to resolve the host, then the controller will follow series of steps to authenticate the host and install the necessary flows in necessary OpenFlow switches to establish the session between the hosts. The active session will be mapped into the table $Act(h_i, h_j)$ to track the number of active sessions in the network. The mutation interval is discrete for each host, and the session will be terminated when the hosts stop communicating with each other. Every time the hosts need to communicate, the name should be resolved and establish a session. SNORT will monitor all the data packets and alert the controller if anomalies are detected according to the rules. SNORT will collect data stats about each host, which can be used to discover highly active hosts in the network. This data can also be used when the unused address space is limited.

4.3.3 Traffic Generation and Reconnaissance

To generate the traffic between the hosts, IPerf tool is used, and using the TCPDump command-line tool, and traffic is captured and saved into a PCAP file for further analysis. The PCAP file is analyzed using Snort IDS. The scanning attack on the network is made using Nmap from Kali Linux OS, as shown in Figure 4.31. SNORT details the number of TCP and UDP sessions between the hosts, events that needed to be reviewed, etc.

Using the SNORT statistics, the scanned IP addresses can be known and the number of times each IP is scanned. Using Nmap scanning, the attacker can gather whether the host is up or down, open ports available, services running on the open ports and their versions etc. Even though the attacker gathers all the details, the IP address will be changed frequently, creating complex situations to understand the network.

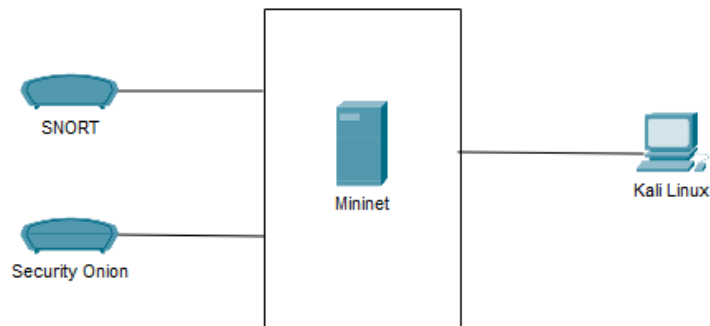


Figure 4.31 Reconnaissance Using Kali

Using security onion as shown in Figure 4.31, further analysis can be done on the PCAP file. If the adversary succeeds reconnaissance stage and try to infect the PC, as a

security analyst, detailed packet analysis should be done. The following Figure 4.32 shows the series of steps involved in the analysis.

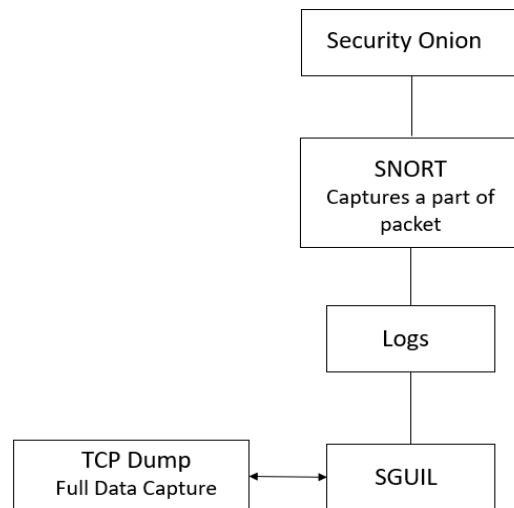


Figure 4.32 SGUIL for Packet Analysis

4.3.4 Results and Analysis

```
Nmap scan report for 198.51.100.88 [host down]
Nmap scan report for 198.51.100.89 [host down]
Nmap scan report for 198.51.100.90 [host down]
Nmap scan report for 198.51.100.91 [host down]
Nmap scan report for 198.51.100.92 [host down]
Nmap scan report for 198.51.100.93 [host down]
Nmap scan report for 198.51.100.94 [host down]
Nmap scan report for 198.51.100.95 [host down]
Nmap scan report for 198.51.100.96 [host down]
Nmap scan report for 198.51.100.97 [host down]
Nmap scan report for 198.51.100.98 [host down]
Nmap scan report for 198.51.100.99 [host down]
Initiating Parallel DNS resolution of 1 host. at 14:25
Completed Parallel DNS resolution of 1 host. at 14:25, 13.00s elapsed
Initiating SYN Stealth Scan at 14:25
Scanning 2 hosts [1000 ports/host]
Discovered open port 5900/tcp on 198.51.100.100
Discovered open port 23/tcp on 198.51.100.100
Discovered open port 111/tcp on 198.51.100.100
Discovered open port 22/tcp on 198.51.100.100
Discovered open port 25/tcp on 198.51.100.100
Discovered open port 80/tcp on 198.51.100.100
Discovered open port 3306/tcp on 198.51.100.100
Discovered open port 445/tcp on 198.51.100.100
Discovered open port 21/tcp on 198.51.100.100
Discovered open port 53/tcp on 198.51.100.100
Discovered open port 139/tcp on 198.51.100.100
Discovered open port 6667/tcp on 198.51.100.100
Discovered open port 2049/tcp on 198.51.100.100
Discovered open port 1099/tcp on 198.51.100.100
Discovered open port 2121/tcp on 198.51.100.100
Discovered open port 1524/tcp on 198.51.100.100
Discovered open port 513/tcp on 198.51.100.100
Discovered open port 8009/tcp on 198.51.100.100
Discovered open port 514/tcp on 198.51.100.100
Discovered open port 512/tcp on 198.51.100.100
Discovered open port 6000/tcp on 198.51.100.100
Discovered open port 8180/tcp on 198.51.100.100
Discovered open port 5432/tcp on 198.51.100.100
Completed SYN Stealth Scan against 198.51.100.100 in 0.34s (1 host left)
```

Figure 4.33 Nmap Scan for Live Hosts

From Figure 4.33, you can see that the adversary can find the active host and identify the open ports available by using stealth scan. Figure 4.34 shows the services running on the open ports and its versions. In Figure 4.35, SNORT collects the individual host's total sessions for a certain period for further analysis.

```

not shown: 577 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    I open  ftp          vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp    I open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
|_ssh-hostkey:
|_ 1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_ 2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp    I open  telnet       Linux telnetd
25/tcp    I open  smtp         Postfix smtpd
|_smtp_commands: metasploitable.localdomain, PIPELINING, SIZE 10240000,
SN,
53/tcp    I open  domain       ISC BIND 9.4.2
|_dns-nsid:
|_ bind.version: 9.4.2
80/tcp    I open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
|_http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
|_http-title: Metasploitable2 - Linux

```

Figure 4.34 Nmap Results

```

Stream statistics:
    Total sessions: 2384
    TCP sessions: 2243
    UDP sessions: 141
    ICMP sessions: 0
    IP sessions: 0
    TCP Prunes: 0
    UDP Prunes: 0
    ICMP Prunes: 0
    IP Prunes: 0
TCP StreamTrackers Created: 2243
TCP StreamTrackers Deleted: 2243

```

Figure 4.35 SNORT Session Results

```

[**] [1:257:9] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/04-19:21:41.257402 198.51.100.50:35385 -> 198.51.100.100:53
TCP TTL:64 TOS:0x0 ID:14648 IpLen:20 DgmLen:84 DF
***AP*** Seq: 0x408EE00 Ack: 0xBE91605B Win: 0x1D TcpLen: 32
TCP Options (3) => NOP NOP TS: 383655 138588
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=10028][Xref => http://www.wh
itehats.com/info/IDS278]

[**] [1:2113:3] RSERVICES rexec username overflow attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
12/04-19:21:41.257415 198.51.100.50:57858 -> 198.51.100.100:512
TCP TTL:64 TOS:0x0 ID:46757 IpLen:20 DgmLen:84 DF
***AP*** Seq: 0xEAEA626B Ack: 0xBE9DBC0D Win: 0x1D TcpLen: 32
TCP Options (3) => NOP NOP TS: 383655 138588

```

Figure 4.36 SNORT Alerts

Figure 4.36 shows the alerts generated by SNORT IDS. The attacker scanned the network for possible information leak, which includes a username overflow attempt classified under attempted administrator privilege gain. The alerts generated by SNORT shows the number of hosts scanned and the information gathered. The mutation technique thwarts the scanning by changing the IP address of the host. The attacker cannot know the details of which host the information is gathered. Some other types of events filtered by SNORT are shown below in Table 4.5 and only few are listed.

Table 4.5 Snort Filtered Events

Events filtered by SNORT
DNS named version attempt
RSERVICES rexec username overflow attempt
SCAN nmap XMAS
POLICY FTP anonymous login attempt
FTP PORT bounce attempt
CHAT IRC nick change
SNMP request tcp
SNMP AgentX/tcp request

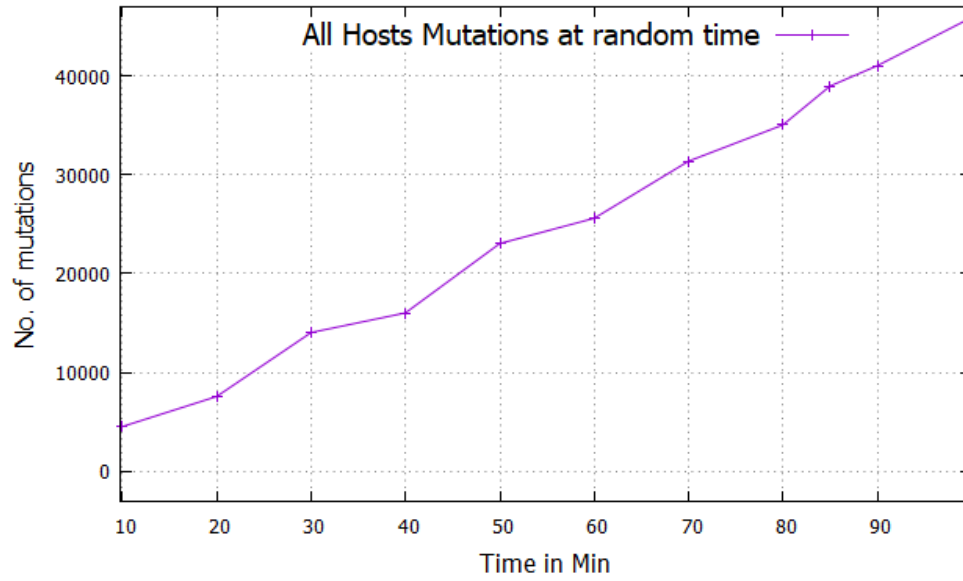


Figure 4.37 Hosts Mutations at the Same Time

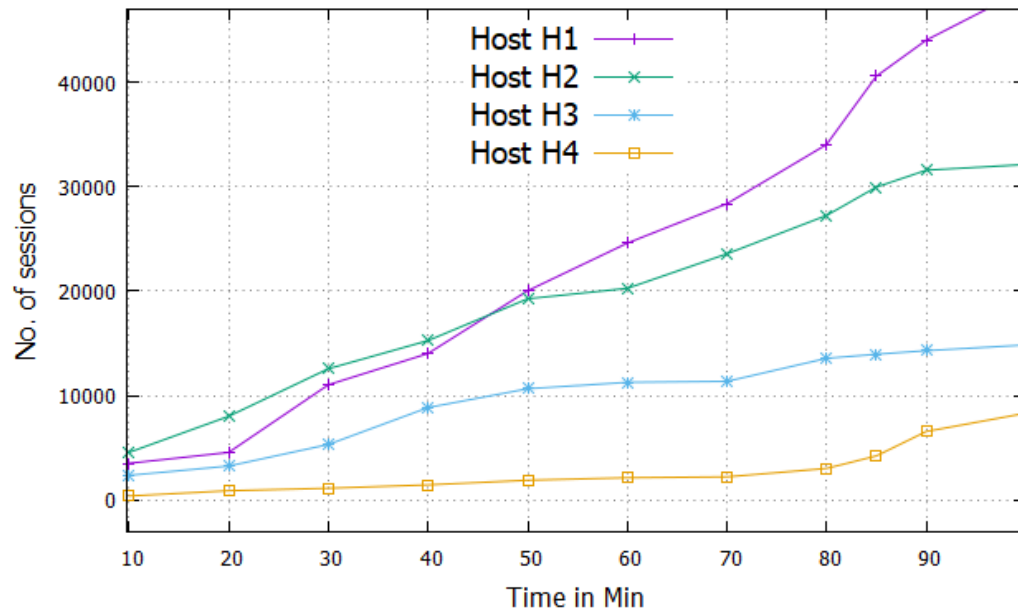


Figure 4.38 Hosts Sessions

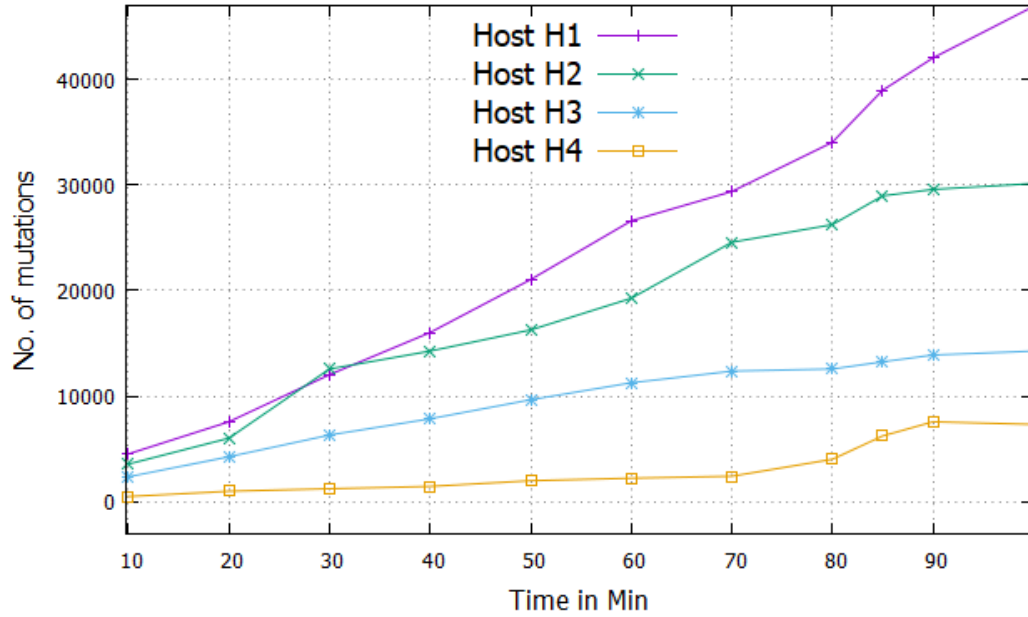


Figure 4.39 individualized Hosts Mutations

The number of mutations of all hosts for a certain period is shown in Figure 4.37. Since the mutation time interval is random for all hosts, the host addresses are changed at the same time for all hosts. If a host is in a TCP/UDP session with other host or server, it will be interrupted due to random mutation. This will create instability in the network, and the network performance will be degraded. Figure 4.38 shows the number of sessions established by each host. The session establishment number of each host varies with one another. Figure 4.39 shows the number of mutations of each host after the application of a discrete mutation interval. The mutation technique is applied when the host terminates the session without interrupting. The stability of the network can be preserved with this technique. The host, which is active in the network, established more sessions than the other hosts. By analyzing this data, the address space range for each host can be assigned. The host which is active is assigned more address space range than other hosts which are less active. In this way, when the availability of host addresses is less, it can be managed

with this technique. When the address space range is less, the same IP address is assigned to the host multiple times in a short interval. This gives the adversary to gather more information with the same IP address.

4.4 Discrete Host Address Mutation with Subnet Game Strategy and Benchmarking

Discrete Host Address Mutation is introduced in section 4.3. In this chapter, subnet game strategy has been included in our discrete MTD algorithm to improve the complexity in understanding the network structure by the attacker.

4.4.1 Subnet Game Strategy

The private IP addresses in a network are not routed on the internet. They are used within the local network. The range of private IP addresses in each class is shown in Figure 4.40.

Class	Private IP address range	Subnet mask	No. of hosts
A	10.0.0.0 – 10.255.255.255	255.0.0.0	16,777,212
B	172.16.0.0 – 172.16.31.255	255.255.0.0	8190
C	192.168.0.0 – 192.168.255.255	255.255.255.0	65,534

Private IP Addresses

Figure 4.40 Private IP Addresses

Consider a network topology as shown in Figure 4.41. The network subnets are ranging from 192.168.1.0/24 to 192.168.7.0/24. In each subnet, you have more than 200 unused IP addresses are left. We can use the unused IP addresses and assign them to the hosts as virtual IP addresses by creating multiple IP address pools. Each pool is a different subnet, as shown in the Figure 4.41.

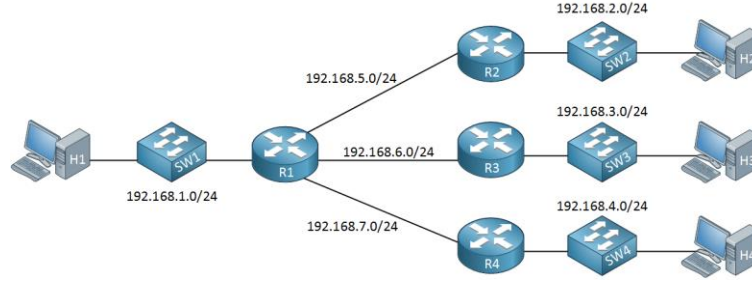


Figure 4.41 Example Subnet Topology

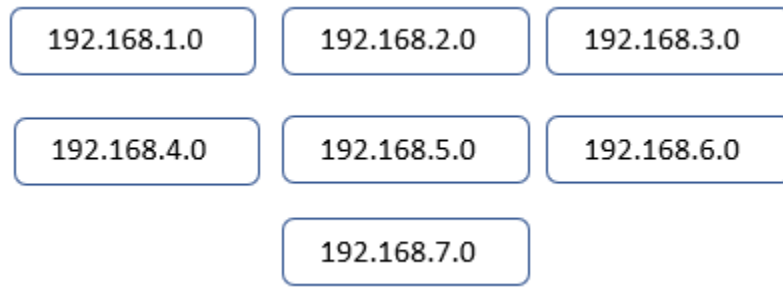


Figure 4.42 Subnet Pools

The virtual IP addresses are selected randomly from each different subnet pool, as shown in Figure 4.42. Implementing this technique increases complexity in understanding the network topology by the attacker. For example, if the initial IP address of the host is 192.168.2.3 and after mutation interval, let us assume the IP address changes to 192.168.4.6. The attacker will get confused to determine the host's subnet and network topology diagram.

4.4.2 Benchmarking and Analysis

4.4.2.1 Benchmarking Topology

This section introduces benchmarking [61] analysis to measure SDN controller network stability in terms of performance, scalability, and reliability. The popular Leaf-Spine network topology is adopted and emulated in Mininet. IPerf traffic generators TP1

and TP2 are initiated at required nodes to generate the network traffic. The Leaf-Spine topology and parameters are shown below in Figure 4.43 and Table 4.6. We used Wireshark to capture the network traffic data and do further analysis. We increased the number of switches and hosts from topology 1 to topology 5. Adding more resources to the network topology increases the load on the controller. As the load is increased from T1 to T5, we measured the network's stability and the controller's performance. The benchmarking analysis is performed on random host address mutation technique and discrete host address mutation technique against traditional software-defined networking, and results are shown. We can observe from the results that the proposed MTD network outperformed the random MTD network. Each test in the topology is repeated for 10 times, and the average value is recorded.

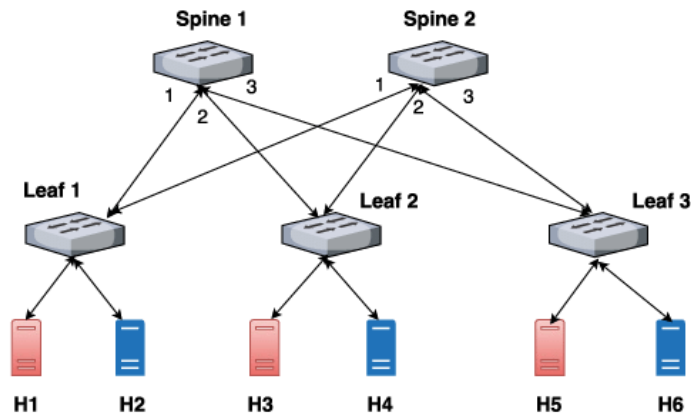


Figure 4.43 Leaf-Spine Topology

Table 4.6 Topologies

Topologies	OVS Switches	Nodes
T1	16	200
T2	32	400
T3	48	600
T4	64	800
T5	80	1000

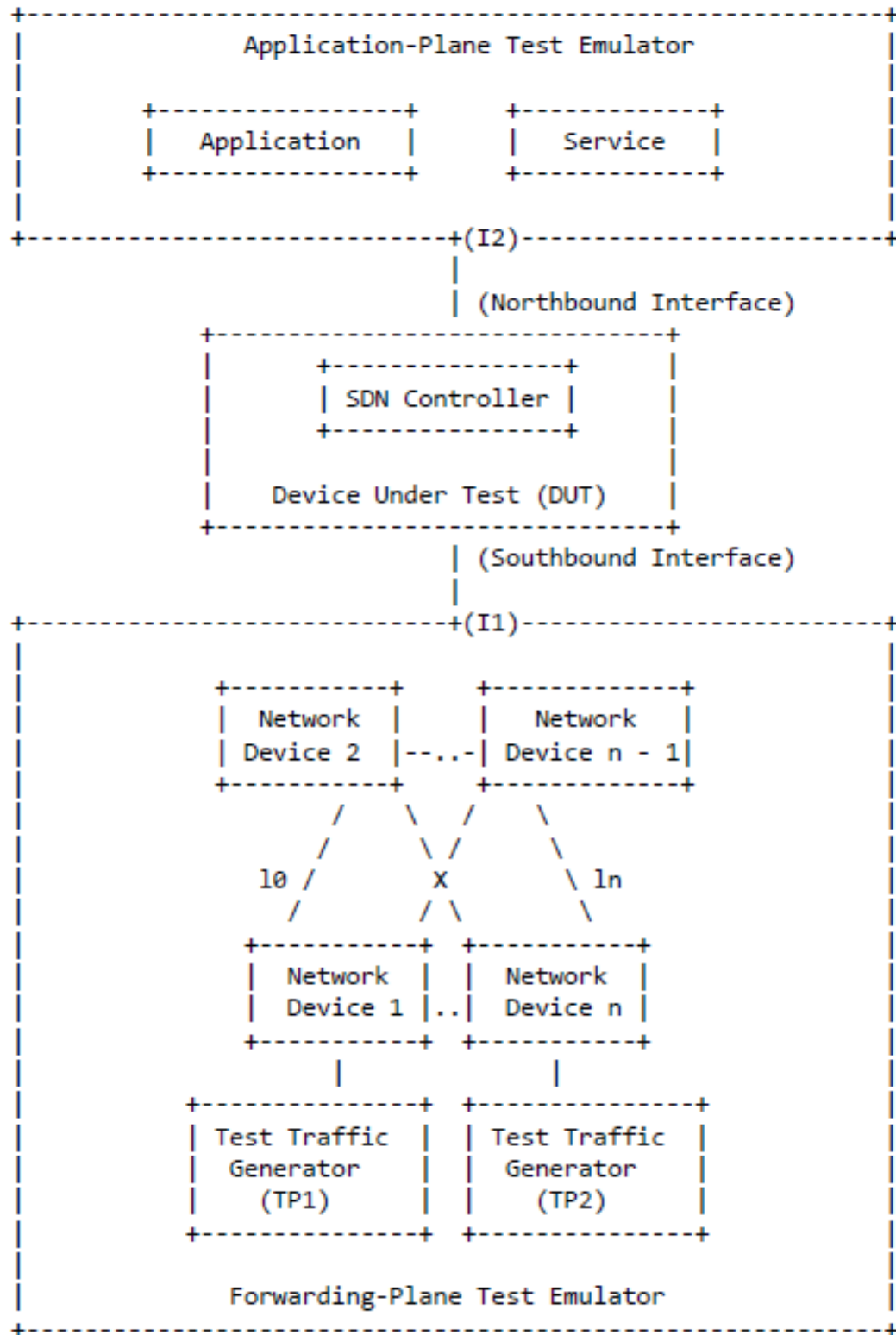


Figure 4.44 Leaf-Spine Topology with Controller

4.4.2.2 Benchmarking Performance

Benchmarking Performance contain the following tests:

1. Network Topology Discovery Time
2. Asynchronous Message Processing Time
3. Asynchronous Message Processing Rate
4. Reactive Path Provisioning Time
5. Proactive Path Provisioning Time
6. Reactive Path Provisioning Rate
7. Proactive Path Provisioning Rate
8. Network Topology Change Detection Time

4.4.2.2.1 Network Topology Discovery Time

The time taken to discover the network devices and determine the complete topology of the network. Link layer discovery protocol is used to determine the discovery time, as shown in Figure 4.45.

Tm1 is the timestamp of the initial discovery message sent by the controller.

Tmn is the final discovery message sent by the controller.

The time for the last discovery message = Tmn

Topology Discovery Time (DT1) = Tmn - Tm1

The average of the topology discovery time is as follows:

$$(TDm) = \frac{DT1 + DT2 + DT3 .. DTn}{Total\ Trials}$$

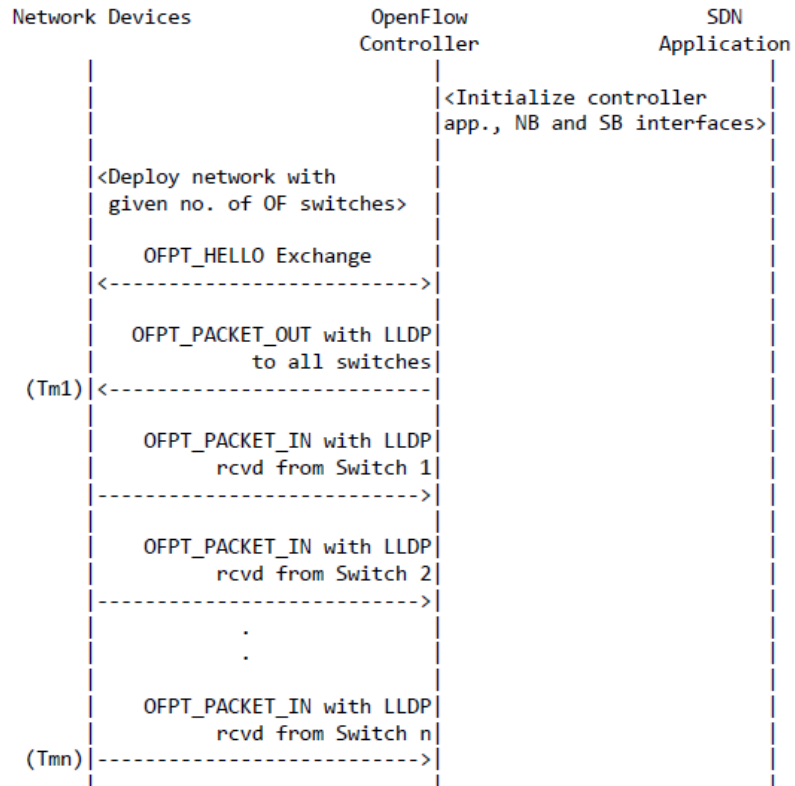


Figure 4.45 Network Topology Discovery Time

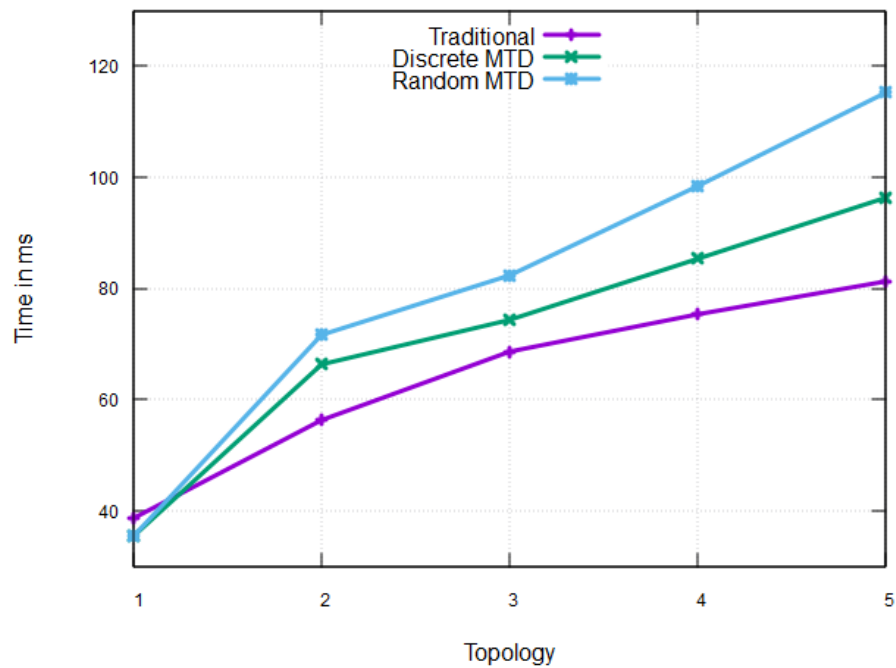


Figure 4.46 Network Topology Discovery Time Benchmark

The network topology discovery time is measured for both random and discrete MTD techniques enabled networks against the traditional network. The Figure 4.46 shows that the traditional network took less time to discover the topology, and the random MTD network took greater time to discover the topology. The discrete MTD network discovery time is closer to the traditional network, as shown in Figure 4.46.

4.4.2.2.2 Asynchronous Message Processing Time

The network devices in the southbound interface often generate notifications for the controller. The total time taken by the controller to process the event is measured as shown in Figure 4.47. The process is repeated for ten trials, and the average value is recorded in each topology. The results for each network type are recorded.

Asynchronous Message Processing Time (APT1) =

$$\frac{SUM\{Ri\} - SUM\{Ti\}}{Nrx}$$

T1 is the event transmission timestamp, and R1 is the response received timestamp.

Nrx is the total number of messages exchanged successfully.

Average Asynchronous Message Processing Time =

$$\frac{APT1 + APT2 + APT3 .. APTn}{Total\ Trials}$$

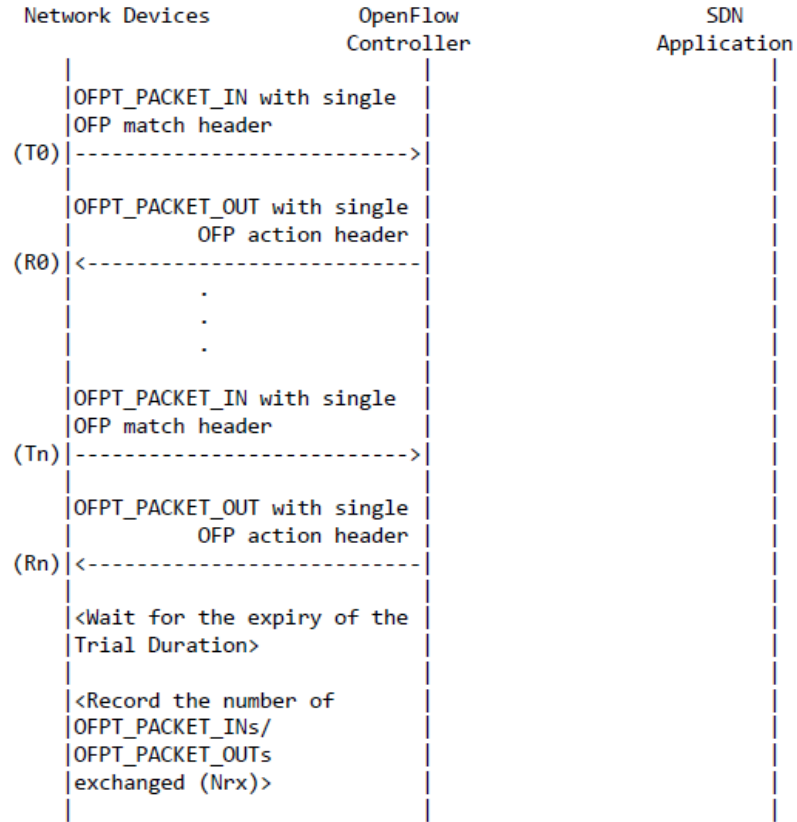


Figure 4.47 Asynchronous Message Processing Time

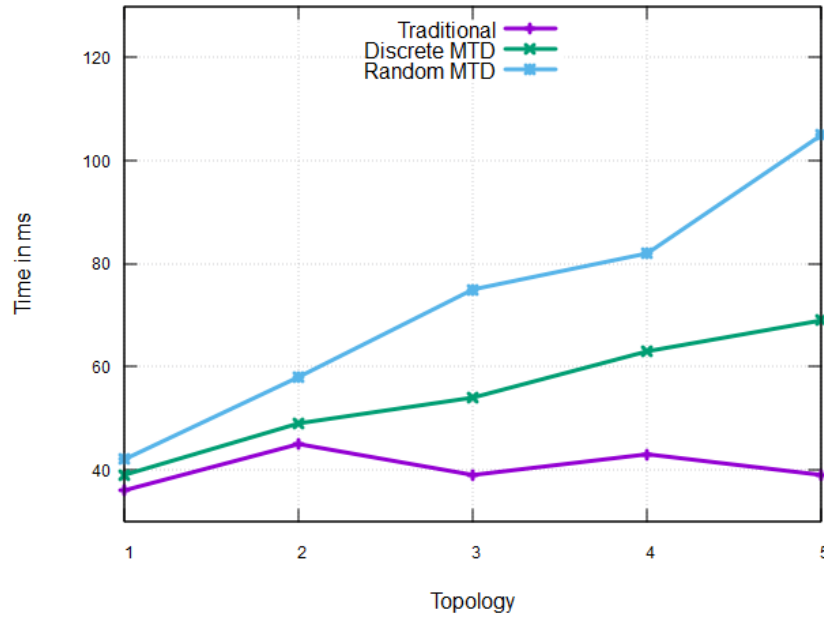


Figure 4.48 Asynchronous Message Processing Time Benchmark

Figure 4.48 shows the time taken to process the message by the controller in each network. The traditional network took a very short time to process, and the random MTD network took a long time when compared to traditional and Discrete MTD networks. The discrete MTD network performed better than the Random MTD network, and it is somewhat closer to the traditional network performance.

4.4.2.2.3 Asynchronous Message Processing Rate

The total number of event messages processed by the controller per second.

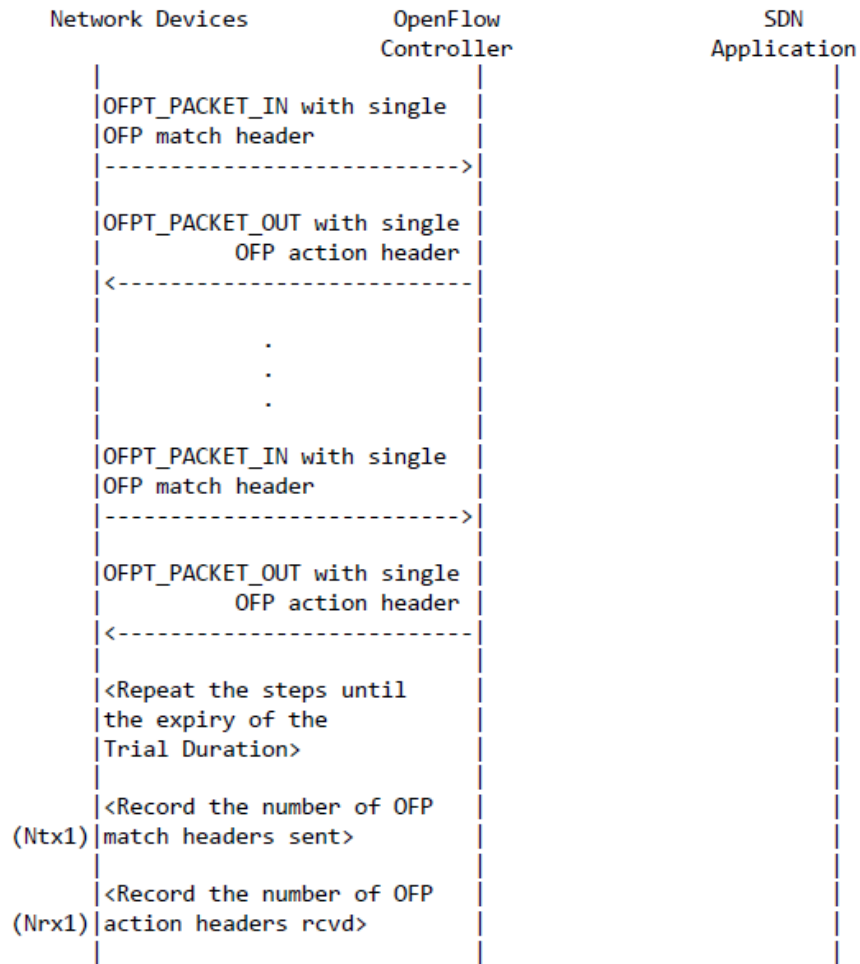


Figure 4.49 Asynchronous Message Processing Rate

It is measured by sending the messages from network devices to the controller for a short time and calculate the number of messages successfully processed by the controller and received by the network devices.

N_{rxn} is the number of successful messages received from the controller, as shown in Figure 4.49.

Asynchronous Message Processing Rate (APR_n) =

$$\frac{N_{rxn}}{T_d}$$

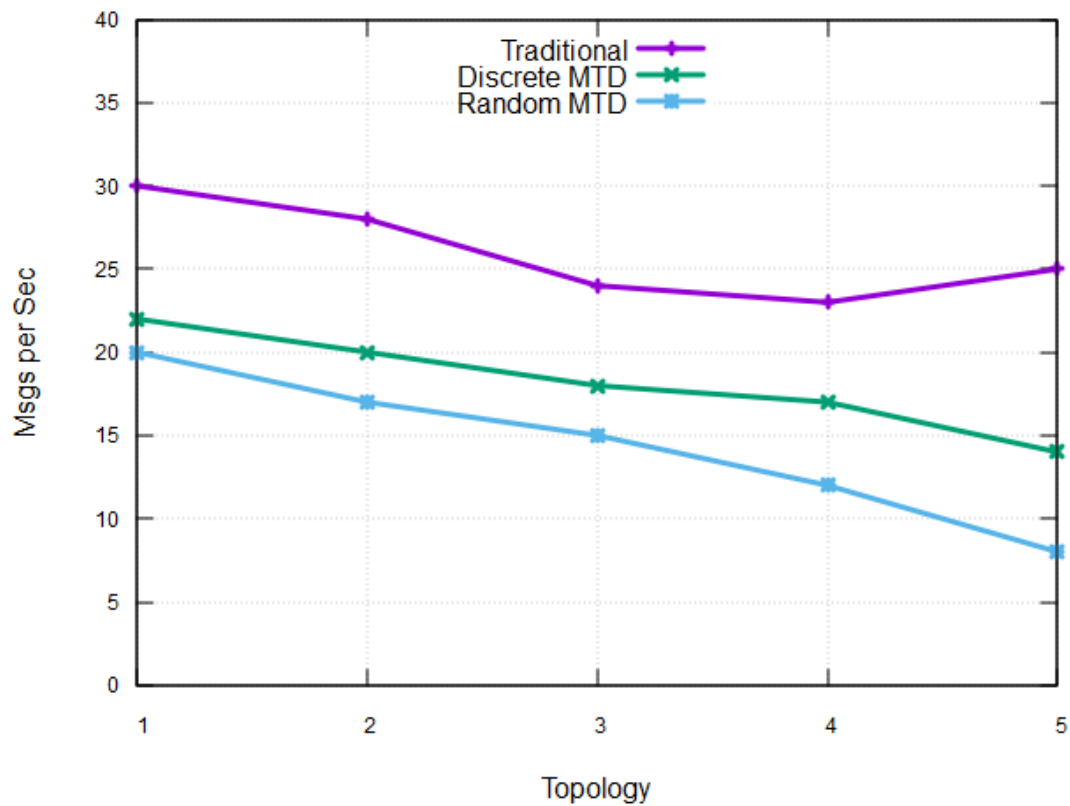


Figure 4.50 Asynchronous Message Processing Rate Benchmark

Figure 4.50 shows the number of messages processed by each network at different topologies. The random MTD network performed worse when compared to others. The discrete MTD network performed well, and its results are closer to the traditional network.

4.4.2.2.4 Reactive Path Provisioning Time

The amount of time taken by the controller to create a reactive path between source and destination nodes is measured. It is critical to track how quickly the controller creates an end-to-end data plane flow. The time period begins with the controller(s) receiving the initial flow provisioning request message and ends with the controller(s) sending the last flow provisioning response message at its southbound interface, as shown in Figure 4.51.

Tsf1 is the timestamp of the provisioning request received by the controller.

Tdf1 is the timestamp of the provisioning response received from the controller.

Reactive Path Provisioning Time (RPT1) = Tdf1 - Tsf1

Average Reactive Path Provisioning Time =

$$\frac{RPT1 + RPT2 + RPT3 .. RPTn}{Total Trials}$$

Figure 4.52 shows the provisioning time taken by all the networks. As the topology load increases, the time taken by the random MTD network is increased at a higher rate. On the other hand, the discrete MTD network performed well, and it is closer to the traditional network.

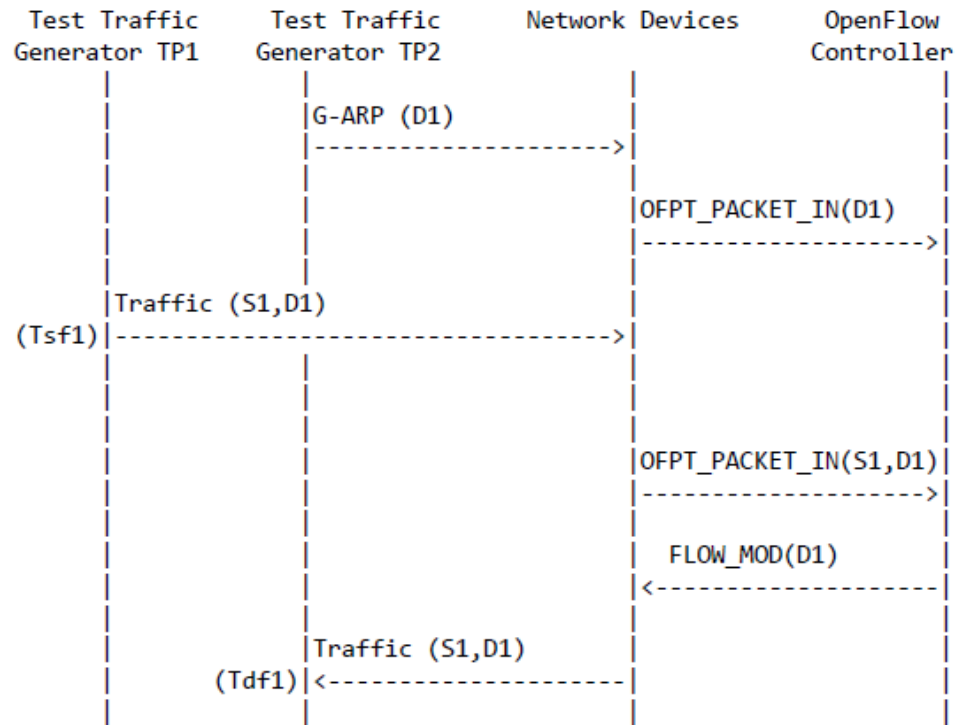


Figure 4.51 Reactive Path Provisioning Time

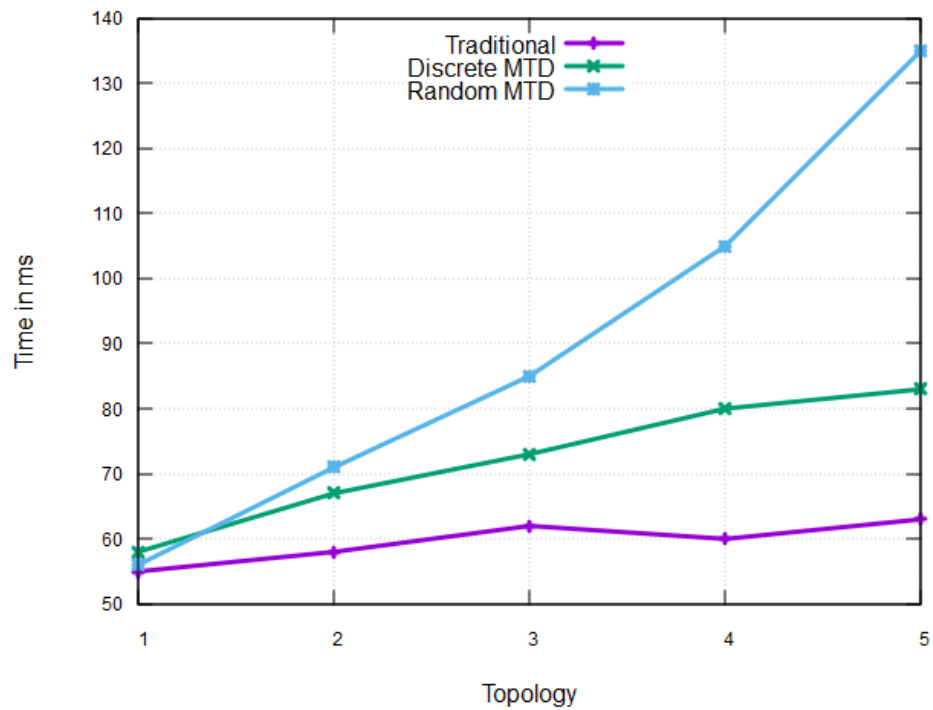


Figure 4.52 Reactive Path Provisioning Time Benchmark

4.4.2.2.5 Reactive Path Provisioning Rate

The maximum number of independent pathways between source and destination nodes that a controller can establish in a single second. The controller's ability to set up as many end-to-end flows in the data plane must be measured. If TP1 is the traffic generator and TP2 is the receiver, then the total number of frames received by the TP2 in a time interval as shown in Figure 4.53.

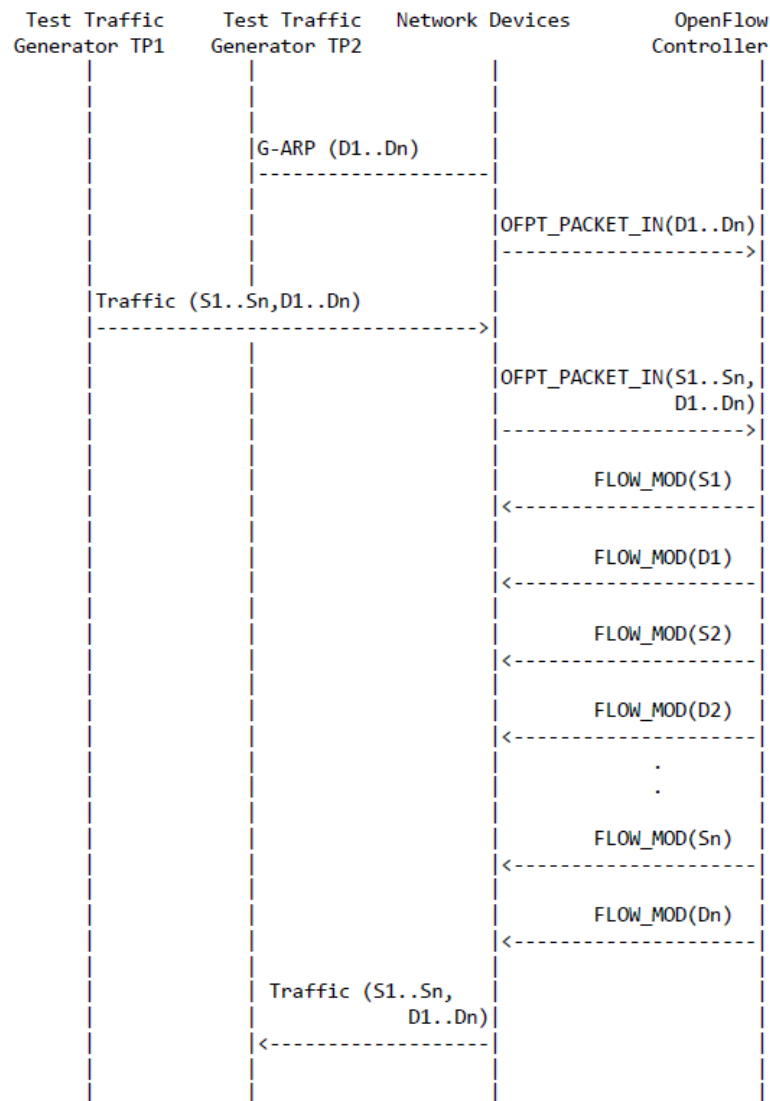


Figure 4.53 Reactive Path Provisioning Rate

$$\text{Reactive Path Provisioning Rate (RPR1)} = \frac{Ndf}{Td}$$

Ndf is the total number of successful traffic frames received at the destination.

Trial Duration (Td)

Average Reactive Path Provisioning Rate =

$$\frac{RPT1 + RPT2 + RPT3 .. RPTn}{Total Trials}$$

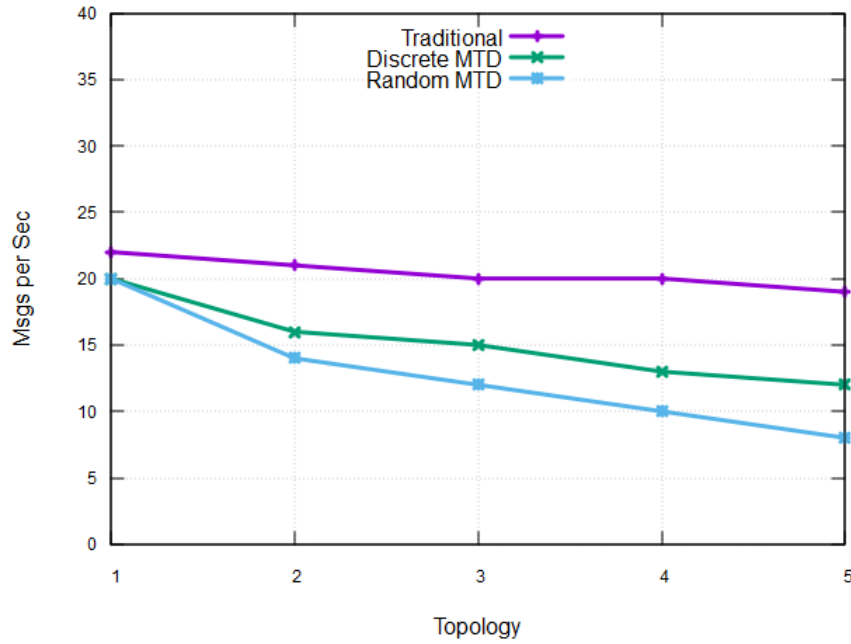


Figure 4.54 Reactive Path Provisioning Rate Benchmark

Figure 4.54 shows the number of unique frames received or the number of paths established per second between the hosts. The random MTD network underperformed as the topology weight is increased. The discrete MTD network performed better, and it is closer to the traditional network.

4.4.2.2.6 Proactive Path Provisioning Time

The time it takes the controller to create a path between the source and destination nodes proactively. It is similar to reactive path provisioning except that the flow is added manually, making it proactive. Here we measure the time taken to provision the proactive path between the hosts, as shown in Figure 4.55.

Tsf1 is the timestamp of the provisioning request received by the controller.

Tdf1 is the timestamp of the provisioning response received from the controller.

Proactive Flow Provisioning Time (PPT1) = Tdf1 - Tsf1

Average Proactive Path Provisioning Time =

$$\frac{PPT1 + PPT2 + PPT3 .. PPTn}{Total\ Trials}$$

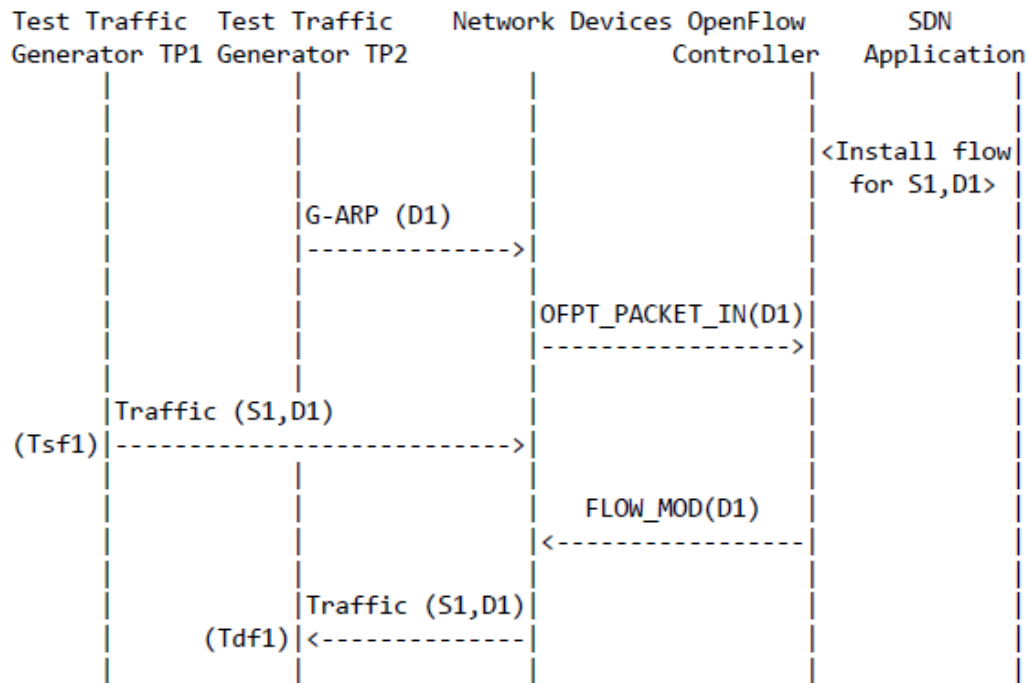


Figure 4.55 Proactive Path Provisioning Time

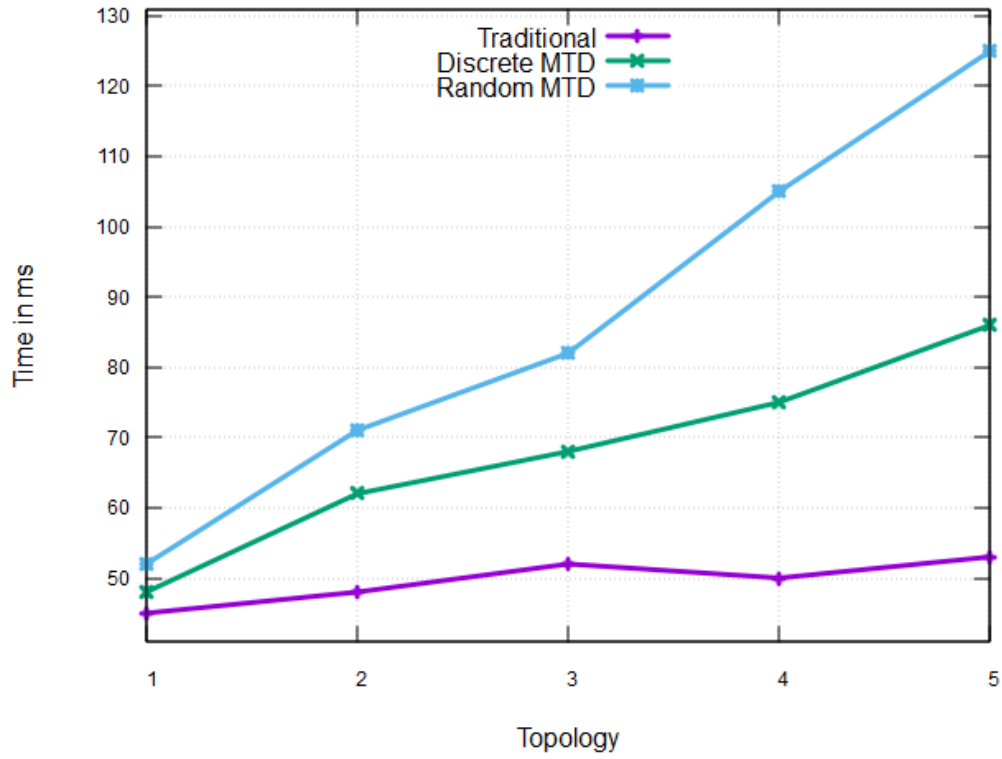


Figure 4.56 Proactive Path Provisioning Time Benchmark

Figure 4.56 shows the time taken by the discrete MTD network is better when compared to a random MTD network. As the topology weight increases, the time taken to provision the path increase very high in a random MTD network, and the discrete MTD network is somewhat closer to the traditional network.

4.4.2.2.7 Proactive Path Provisioning Rate

The maximum number of independent pathways between source and destination nodes that a controller can establish in a single second is shown in Figure 4.57.

$$\text{Proactive Path Provisioning Rate (PPR1)} = \frac{Ndf}{Td}$$

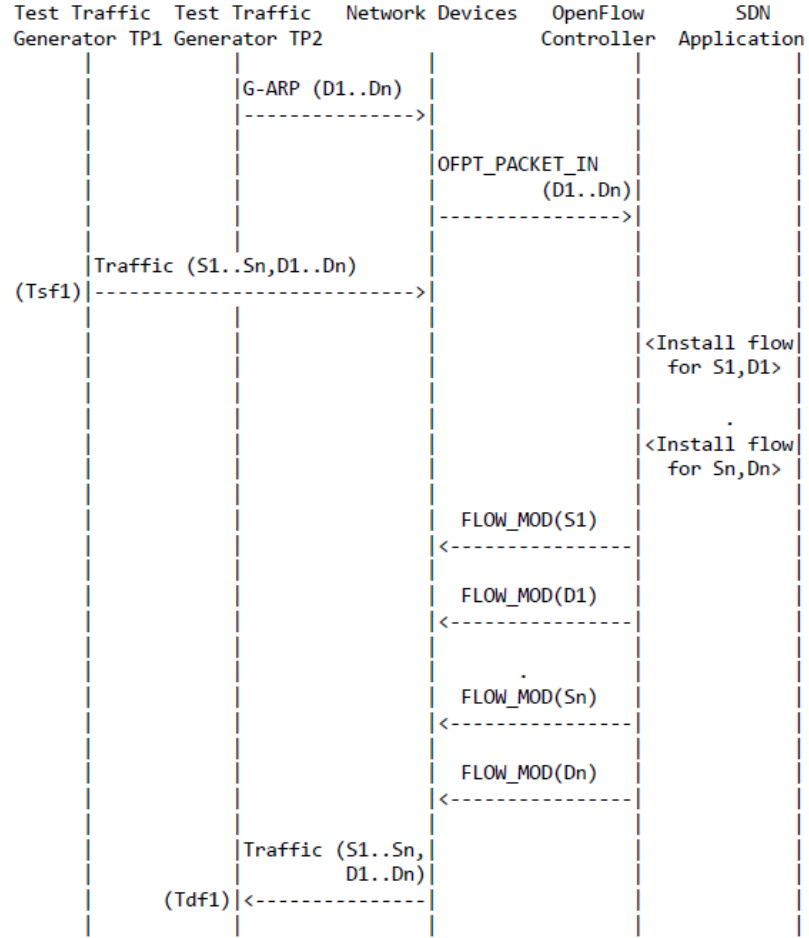


Figure 4.57 Proactive Path Provisioning Rate

Ndf is the total number of successful traffic frames received at the destination.

Trial Duration (Td)

Average Proactive Path Provisioning Rate =

$$\frac{PPR1 + PPR2 + PPR3 .. PPRn}{Total\ Trials}$$

Figure 4.58 shows the random MTD network underperformed in the proactive provisioning rate. The discrete MTD network performed fairly and is closer to the traditional network.

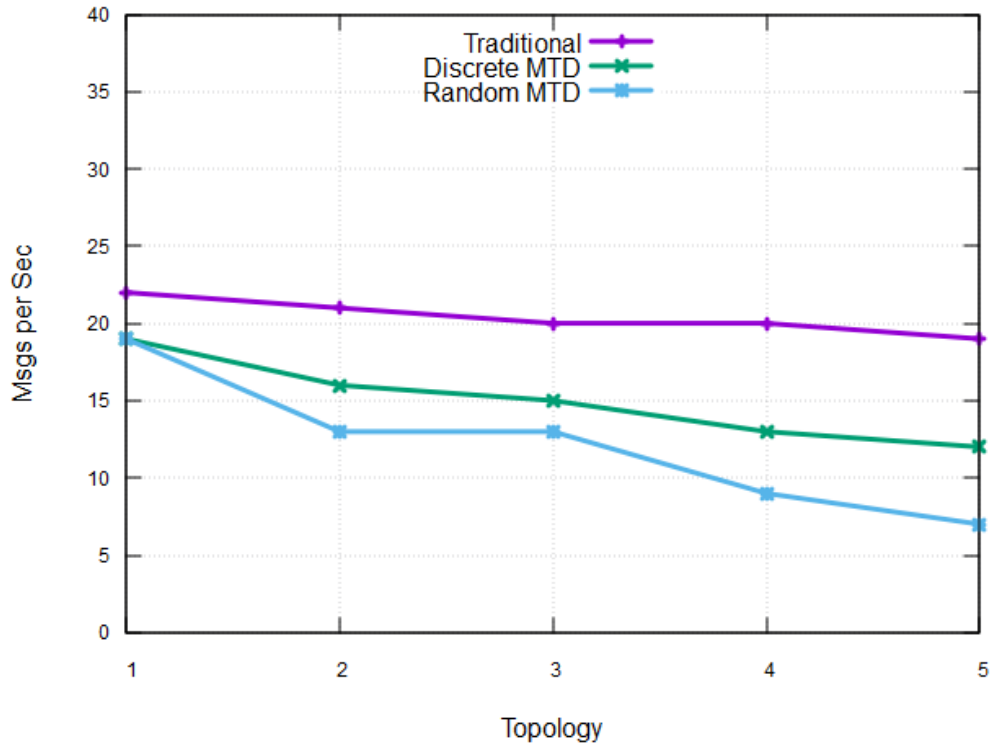


Figure 4.58 Proactive Path Provisioning Rate Benchmark

4.4.2.2.8 Network Topology Change Detection Time

The time it takes for the controller to notice changes in the network topology. It is vital to test how quickly the controller can identify any network-state change events to provide fast network failure recovery.

Tcn is the time when the controller receives the first topology change notification.

Tcd is the time when the controller sends the initial topology rediscovery message.

Network Topology Change Detection Time (TDT1) = Tcd - Tcn

Average Network Topology Change Detection Time =

$$\frac{TDT1 + TDT2 + TDT3 \dots TDTn}{Total\ Trials}$$

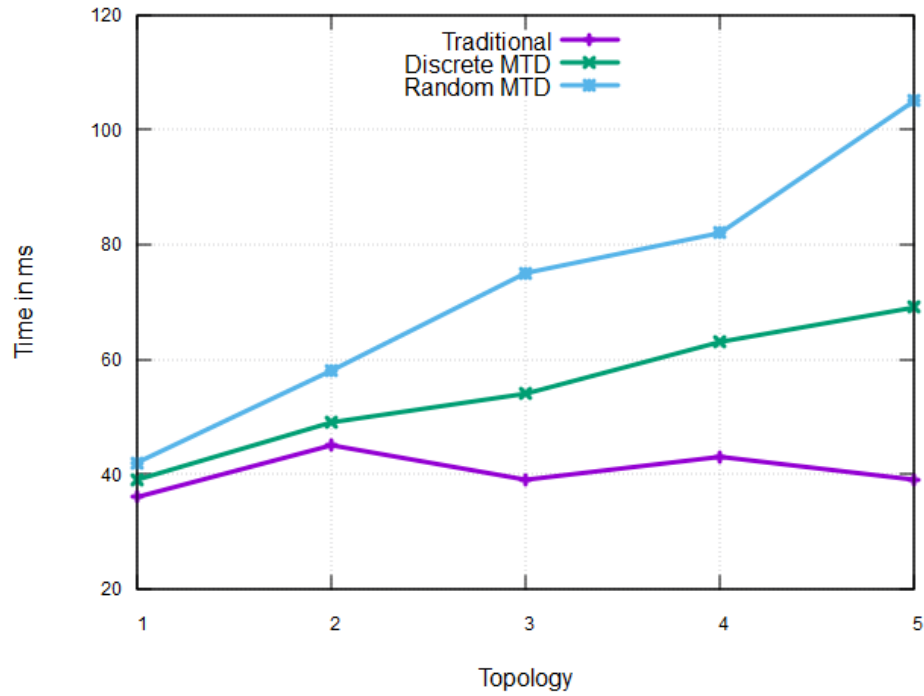


Figure 4.59 Network Topology Change Detection Time Benchmark

Figure 4.59 shows the discrete MTD network performed better than the random MTD network. The controller does not have additional operational cost in the traditional network to frequently mutate the host's IP address since the network is static. The random MTD network has more operational cost than the discrete MTD network, reducing the controller processing speed.

4.4.2.3 Benchmarking Scalability

4.4.2.3.1 Control Sessions Capacity

At a given time, the number of control sessions that the controller can monitor simultaneously. Control Sessions Capacity must be measured to determine the controller's system and bandwidth resource requirements. The control session capacity is measured by increasing the number of switches in the network and record the number of hello exchange

messages between the controller and the switch. If the hello messages exchanged become constant even though the switches are added to the topology, then that number is the capacity of the controller to handle those many switches simultaneously. The following Figure 4.60 shows control sessions capacity measurement.

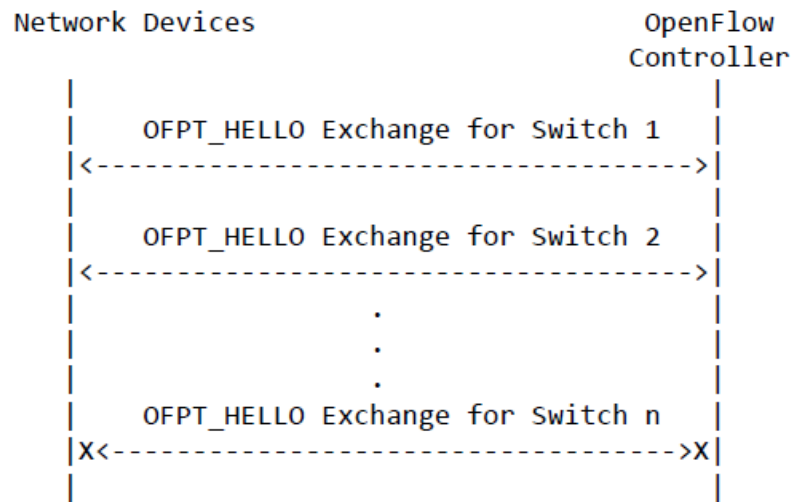


Figure 4.60 Control Sessions Capacity

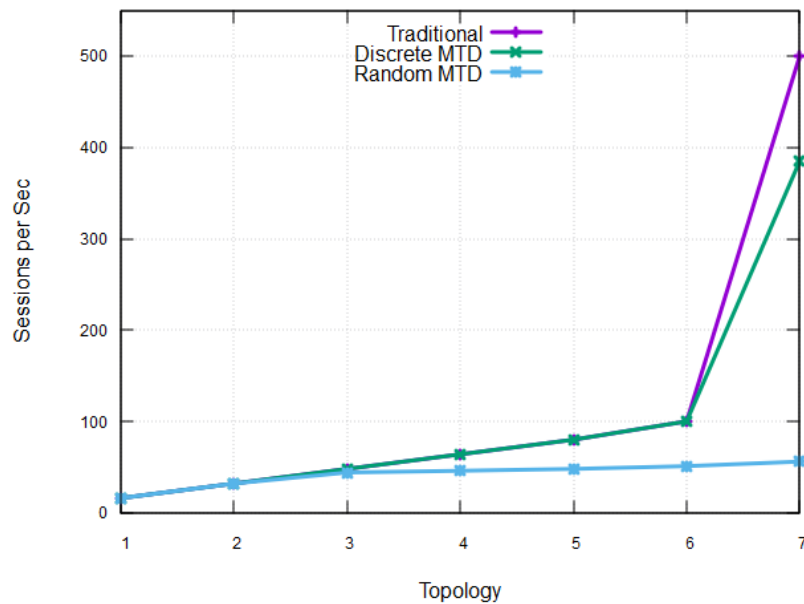


Figure 4.61 Control Sessions Capacity Benchmark

The Figure 4.61 shows the control sessions capacity of the controller. We have added T6 and T7 topology with high number of switches to find the session capacity of the controller. We can observe that the random MTD network underperformed, and it can handle the switches of less than a hundred simultaneously. The traditional network can handle up to five hundred switches, and the discrete MTD network performed well and the results are close to the traditional network.

4.4.2.3.2 Forwarding Table Capacity

The capacity of the forwarding table in the controller. It is critical to determine the number of flows the controller can handle and forward traffic without dropping.

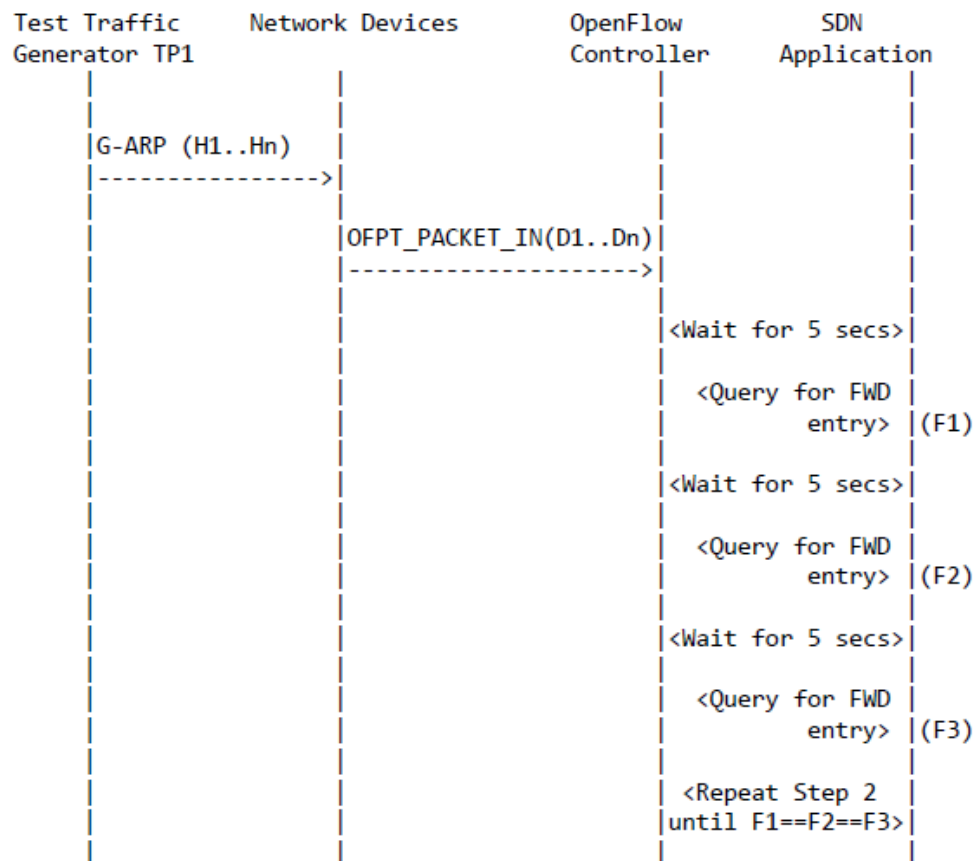


Figure 4.62 Forwarding Table Capacity

It is measured by sending traffic from multiple hosts to multiple destinations. When the controller receives the traffic, it will create an entry into the forwarding table for the traffic flow. The traffic is sent to the multiple destinations until the query to the forwarding table returns the same for 3 to 5 times as shown in the Figure 4.62.

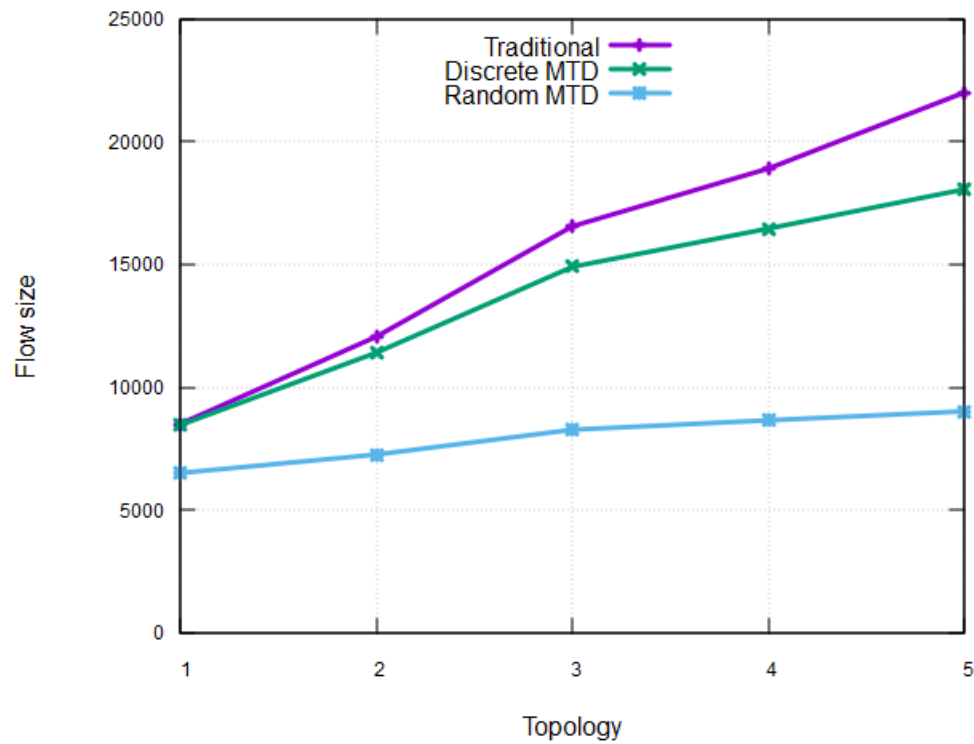


Figure 4.63 Forwarding Table Capacity Benchmark

From the Figure 4.63, the flow table capacity of traditional network and discrete MTD network is increased as the topology weight is increased. The random MTD network handles less flows when compared to others.

4.4.2.4 Benchmarking Reliability

4.4.2.4.1 Controller Failover Time

When the controllers are in redundancy mode, and one of the active controllers fails, the time it takes to transition from the active controller to the backup controller. The time period begins when the active controller is turned off and ends when the new controller's southbound interface receives the first rediscovery message. When two controllers are paired together, and one of them fails, this benchmark assesses the impact of provisioning new flows. The Controller Failover Time is calculated as the difference between the final valid frame received before the traffic loss and the first valid frame received after the traffic loss.

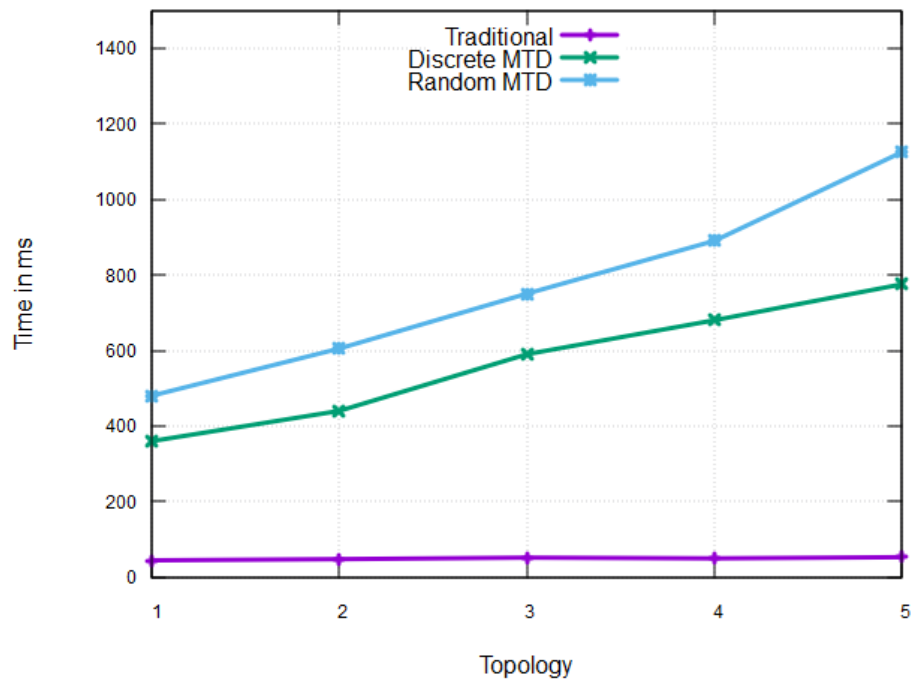


Figure 4.64 Controller Failover Time Benchmark

Figure 4.64 shows the controller failover time. The time taken by the backup controller to become active in a traditional network is very small when compared to the

random and discrete MTD networks. The traditional network is static, and there is no IP address mutation. On the other hand, random and discrete MTD networks continuously change the IP address of the host. In dynamic networks, the controller has to synchronize data with the backup controller to avoid data loss. Data Synchronization is a limitation in this research and synchronizing the data across the controllers is a costly operation and is considered future work. Since the dynamic networks change the IP addresses frequently, when the controller failed, it took more time to become active for the backup controller. However, the discrete MTD network performed better than the random MTD network.

4.4.2.4.2 Network Re-provisioning Time

When existing traffic paths fail, the time it takes for the controller to reroute traffic.

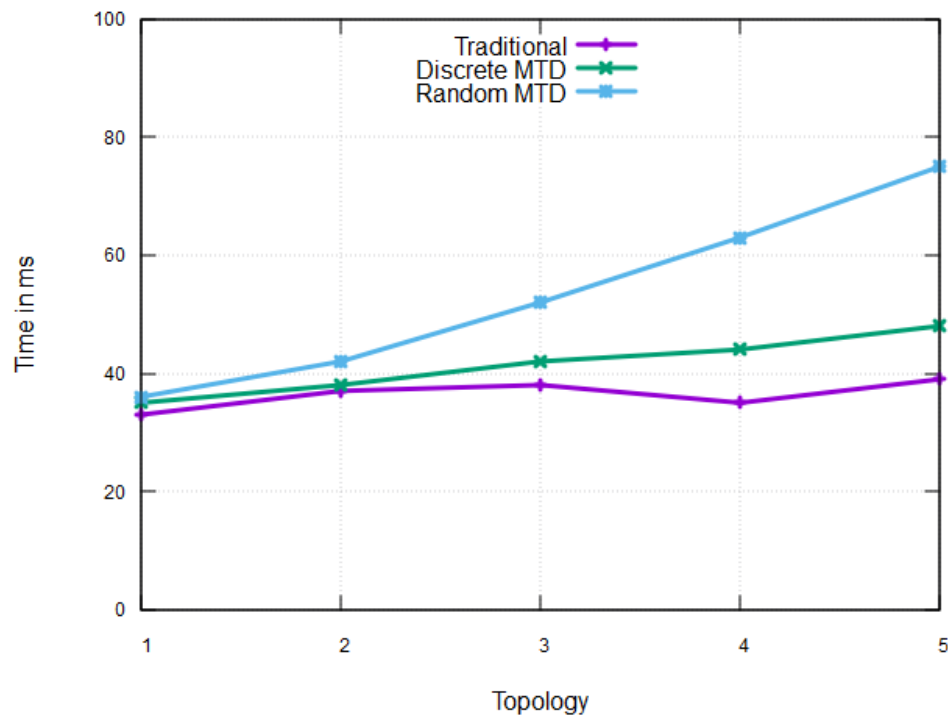


Figure 4.65 Network Re-provisioning Time Benchmark

To measure network re-provisioning time, record the timestamp of the last frame received by the host before the loss of traffic and the timestamp recorded when the host receives the first frame after the loss of traffic. The difference of these two timestamps gives the time required to re-provision the network.

In Figure 4.65, the time taken by the traditional network controller is less when compared to dynamic networks. The discrete MTD network performed better than the random MTD network, and it is close to the traditional network performance.

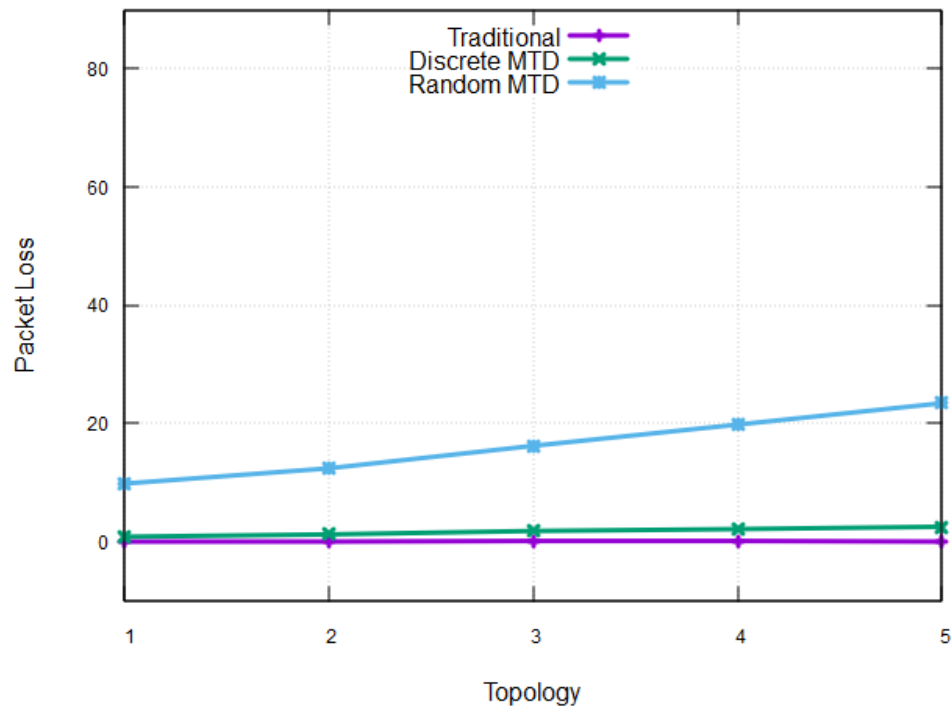


Figure 4.66 Packet Loss

Figure 4.66 shows the packet loss in each topology T1 to T5. As the topology weight is increasing on the network, the controller's load will also increase to process high amount of data. In a random MTD network, since all host IP addresses are changed simultaneously, the session termination is made, and data packets are lost in the

transmission. The re-transmission of lost data packets occupies the network bandwidth, resulting in slower processing of new data by the controller, and also packet loss is high. The discrete MTD network has lower amounts of packet loss due to the individualization of mutation intervals. The sessions are not interrupted in the network; as a result, the packet loss is less and closer to the traditional network.

CHAPTER V – CONCLUSION AND FUTURE WORK

This chapter summarizes the contributions of this research, and the future works are illustrated at the end. We have introduced four studies related to moving target defense techniques application and analysis. In the first study, MTD technique is implemented on drone's wireless network. We observed that the latency in the network is increasing as the number of mutations increase. To observe the network performance after the application of MTD technique, we have implemented random host address mutation technique in software-defined networking and network performance is analyzed. We observed that the MTD algorithms is creating disturbances in the network that we have discussed. In the third study, we have proposed a new discrete MTD technique to improve network performance and reduce the disturbances in the network. In the fourth study, we have added a game strategy improve the security of the network and the discrete MTD technique, random MTD techniques are benchmarked against the traditional network.

In the first study, a MAC address dynamic mutation technique is developed and implemented on the drone wireless network, including intrusion detection system and enhanced security with wireless network encryption. The MAC address of the drone is used to launch a cyber-attack. In the reconnaissance stage of the cyber kill chain, the hacker collects the configuration of the drone network. Once the information is gathered, the hacker does not need to re-scan since the wireless network of the drone is static. We developed an MTD technique to change the static MAC address of the drone frequently. Since the MAC address is changing from time to time, the attacker will be unsuccessful in launching an attack on the network using the network configuration information collected previously.

In the second study, A random host address mutation technique is developed in software-defined networking, and network analysis is shown. The mutation of the host IP addresses of all hosts is implemented at a random time in the network. The IP address mutation is a widely researched technique, and it provides network security by assigning a virtual IP address to the host at a time interval. Thus, the mutation technique not only provides security to the network but also creates an overhead for changing the host's configuration by the SDN controller. We implemented the random host address mutation technique, and the performance of the network is shown by benchmarking against the traditional network.

The third study, we proposed a novel discrete host address mutation technique in software-defined networking to individualize the mutation interval of each host in the network. In random host address mutation, the IP addresses of all hosts are changed at the same interval, terminating the established session between the hosts deteriorating the network stability. To overcome this backlog, a discrete host address mutation is developed to individualize the mutation interval of each host. However, individualizing the mutation interval of each host makes it complex for the attacker to understand the mutation interval. The mutation interval of each host is based on the flow statistics of the host monitored by the controller. The controller changes the IP address of the host when there is no exchange of data.

In fourth study we added an IP subnet game strategy to our MTD technique to enhance the security and we benchmarked the discrete and random MTD networks against the traditional network. The virtual IP addresses are selected randomly from the pool of different IP subnets to make it complex for the attacker to understand the network topology.

A benchmarking framework is developed to measure the stability of the network in terms of performance, scalability, and reliability.

The random host address mutation has limitations such as the mutation of host IP address is implemented at random time. If the host is in a session of exchanging data with other hosts, it is forced to terminate the session and change the IP address resulting in packet loss and additional overhead on the controller to transmit the lost data packets. The operational cost on the controller increases as the topology weight is increased on the network resulting in underperformance of the network controller. We have proposed a discrete moving target defense technique that individualize each host's mutation interval to solve the above problem. The mutation interval of each host on the network is based on its flow stats. We used SDN controller Ryu northbound Rest API service to collect the flow stats of each host for each time interval. When the host is not exchanging any data, the IP address is changed. In this way, the packet loss is reduced in the network and also operational cost on the controller.

Finally, the IP subnet strategy is developed to increase in complexity for the attacker to understand the network topology. In the reconnaissance phase, the attacker scans the network using multiple stealth mode tools to gather the network's data and create a network topology diagram. When the host's IP address is assigned a virtual IP address from the pool of different subnets each time, the attacker finds it complex to create a network topology diagram. The benchmarking framework is developed to measure the stability of the network in terms of performance, scalability, and reliability. The discrete MTD network outperformed in all tests compared to random MTD network showing that discrete host address mutation is a better technique to apply on the network for security

and stability. The drawback found in benchmarking is that the controller's failover time. When the controller fails, the backup controller should become active and take over the network control. Since the IP address mutation is implemented, there is a need for the controller to synchronize the data, and further discussed in future works.

Future Works. In the conclusion of this research, we highlighted a critical topic for future work on controller failover time. Since the static network does not need to change the IP address of each, the failover time is very small, and the IP addresses remain the same. In the case of a dynamic network, the controller has to change the IP address of each host after a time interval. When the controller fails, the tracking data of each host's current virtual IP address will be lost. There will be a data loss before the new controller takes over. Even though the new controller takes over, there is no data synchronized from the failed controller to the backup controller, so that the hosts need to establish new sessions to transmit the data. Synchronizing controller data will create a high operational cost, and if only the important data is synchronized, the cost can be reduced. So that future work will be focused on backup controller synchronization, and reduction of controller fail over time for better stability of the network. The future work also includes exploring other categories of MTD techniques and further improving the performance of discrete MTD networks.

REFERENCES

- [1] "CSD-MTD," *Department of Homeland Security*, 21-Sep-2018. [Online]. Available: <https://www.dhs.gov/science-and-technology/csd-mtd>. [Accessed: 20-Jun-2021].
- [2] Ward, Bryan C., Steven R. Gomez, Richard Skowrya, David Bigelow, Jason Martin, James Landry, and Hamed Okhravi. "Survey of Cyber Moving Targets Second Edition." (2018).
- [3] "Software-Defined Networking (SDN) Definition," Open Networking Foundation, 03-Jun-2020. [Online]. Available: <https://opennetworking.org/sdn-definition/>. [Accessed: 20-Jun-2021].
- [4] "List of SDN controller software," Wikipedia, 06-Apr-2021. [Online]. Available: https://en.wikipedia.org/wiki/List_of_SDN_controller_software. [Accessed: 20-Jun-2021].
- [5] Gudla, Charan, Md Shohel Rana, and Andrew H. Sung. "Defense techniques against cyber attacks on unmanned aerial vehicles." Proceedings of the International Conference on Embedded Systems, Cyber-physical Systems, and Applications (ESCS). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2018, pp. 110-116.
- [6] Gudla, C. and Sung, A.H., 2020, November. Moving Target Defense Application and Analysis in Software-Defined Networking. In 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) (pp. 0641-0646). IEEE.
- [7] C. Gudla and A. H. Sung, " Moving Target Defense Discrete Host Address Mutation and Analysis in SDN," 2020 International Conference on Computational Science and Computational Intelligence (CSCI), 2020. IEEE, pp. 55-61.
- [8] S. I. S. A.- P. S. Specialists, "Reconnaissance – the Eagle's Eye of Cyber Security - SISA Blog," SISA Information Security, 15-May-2020. [Online]. Available: <https://www.sisainfosec.com/blogs/reconnaissance-the-eagles-eye-of-cyber-security/>. [Accessed: 20-Jun-2021].
- [9] Yuan Shi, Huanguo Zhang, Juan Wang, Feng Xiao, Jianwei Huang, Daochen Zha, Hongxin Hu, Fei Yan, Bo Zhao, "CHAOS: An SDN-Based Moving Target Defense System", *Security and Communication Networks*, vol. 2017, Article ID 3659167, 11 pages, 2017. <https://doi.org/10.1155/2017/3659167>.

- [10] W. Xia, Y. Wen, C. H. Foh, D. Niyato and H. Xie, "A Survey on Software-Defined Networking," in *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27-51, Firstquarter 2015, doi: 10.1109/COMST.2014.2330903.
- [11] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.
- [12] Braun W, Menth M. Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices. *Future Internet*. 2014; 6(2):302-336. <https://doi.org/10.3390/fi6020302>.
- [13] K. Bakshi, "Considerations for Software Defined Networking (SDN): Approaches and use cases," 2013 IEEE Aerospace Conference, 2013, pp. 1-9, doi: 10.1109/AERO.2013.6496914.
- [14] V. Pashkov, A. Shalimov and R. Smeliansky, "Controller failover for SDN enterprise networks," 2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), 2014, pp. 1-6, doi: 10.1109/MoNeTeC.2014.6995594.
- [15] Berde, P., Snow, W., Parulkar, G., Gerola, M., Hart, J., Higuchi, Y., Radoslavov, P. (2014). ONOS. Proceedings of the Third Workshop on Hot Topics in Software Defined Networking - HotSDN '14. doi:10.1145/2620728.2620744.
- [16] Goransson, P., Black, C., & Culver, T. (2016). Software defined networks: a comprehensive approach. Morgan Kaufmann.
- [17] Ryu SDN Framework—Open-source SDN Platform Software | NTT Technical Review. (n.d.). NTT. Retrieved June 20, 2021, from <https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201408fa4.html>.
- [18] S. Asadollahi, B. Goswami and M. Sameer, "Ryu controller's scalability experiment on software defined networks," 2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), 2018, pp. 1-5, doi: 10.1109/ICCTAC.2018.8370397.
- [19] Contributors, M. P. (n.d.). Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet. Mininet.Org. Retrieved June 20, 2021, from <http://mininet.org/>.

- [20] R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda and Ligia Rodrigues Prete, "Using Mininet for emulation and prototyping Software-Defined Networks," 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), 2014, pp. 1-6, doi: 10.1109/ColComCon.2014.6860404.
- [21] Jajodia, S., Ghosh, A. K., Swarup, V., Wang, C., & Wang, X. S. (Eds.). (2011). *Moving target defense: creating asymmetric uncertainty for cyber threats* (Vol. 54). Springer Science & Business Media.
- [22] Lichtman, A., & Nair, M. (2015). Humanitarian uses of drones and satellite imagery analysis: the promises and perils. *AMA journal of ethics*, 17(10), pp. 931-937.
- [23] Udeanu, Gheorghe, et al. "Unmanned Aerial Vehicle in Military Operations." Scientific Research and Education in the Air Force, vol. 18, no. 1, 2016, pp. 199-206., doi:10.19062/2247-3173.2016.18.1.26.
- [24] Kafi, Mohamed Amine, et al. "A Study of Wireless Sensor Networks for Urban Traffic Monitoring: Applications and Architectures." *Procedia Computer Science*, vol. 19, 2013, pp. 617–626., doi: 10.1016/j.procs.2013.06.082.
- [25] Alvear, Oscar, et al. "Using UAV-Based Systems to Monitor Air Pollution in Areas with Poor Accessibility." *Journal of Advanced Transportation*, vol. 2017, 2017, pp. 1–14., doi:10.1155/2017/8204353.
- [26] Debusk, Wesley. "Unmanned Aerial Vehicle Systems for Disaster Relief: Tornado Alley." *AIAA Infotech@Aerospace 2010*, 2010, doi:10.2514/6.2010-3506.
- [27] Waharte, Sonia, and Niki Trigoni. "Supporting Search and Rescue Operations with UAVs." 2010 International Conference on Emerging Security Technologies, 2010, doi:10.1109/est.2010.31.
- [28] Guillen-Perez, Antonio, et al. "Wi-Fi Networks on Drones". 2016 ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT), 2016, doi:10.1109/itu-wt.2016.7805730.
- [29] Amazon prime air delivery using drones to deliver the ordered packages, <https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011> [Accessed: 20-Jun-2021].
- [30] Iraqi insurgents hacked predator drone feeds, <http://www.cnn.com/2009/US/12/17/drone.video.hacked/index.html> [Accessed: 20-Jun-2021].

- [31] The Computer virus infects drone plane command centre US, <https://www.theguardian.com/technology/2011/oct/09/virus-infects-drone-plane-command> [Accessed: 20-Jun-2021].
- [32] Wikipedia contributors. (2021, April 15). Iran–U.S. RQ-170 incident. Wikipedia. https://en.wikipedia.org/wiki/Iran%E2%80%93U.S._RQ-170_incident. [Accessed: 20-Jun-2021].
- [33] The Predator drone live video feeds exposed online, <https://www.bleepingcomputer.com/news/government/us-government-leaves-predator-drone-video-feeds-exposed-online/> [Accessed: 20-Jun-2021].
- [34] N. M. Rodday, R. D. O. Schmidt, A. Pras. "Exploring security vulnerabilities of unmanned aerial vehicles", NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, pp. 993-994, Apr 2016.
- [35] J. S. Pleban, R. Band, R. Creutzburg, R. Creutzburg, D. Akopian, "Hacking and securing the AR.Drone 2.0 quadcopter: Investigations for improving the security of a toy", International Society for Optics and Photonics, pp. 90300L, feb 2014.
- [36] Rani C, Modares H, Sriram R, Mikulski D, Lewis FL (2016): Security of unmanned aerial vehicle systems against cyber-physical attacks. *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* 2016, Vol. 13(3) pp. 331–342 The Author(s) 2015 DOI: 10.1177/1548512915617252.
- [37] K. Hartmann, C. Steup, "The vulnerability of UAVs to cyber-attacks an approach to the risk assessment", *Cyber Conflict (CyCon) 2013 5th International Conference on*, pp. 1-23, 2013.
- [38] Goppert, James, et al. "Numerical Analysis of Cyberattacks on Unmanned Aerial Systems." *Infotech@Aerospace 2012*, 2012, doi:10.2514/6.2012-2437.
- [39] R. Mitchell, I.-R. Chen, "Adaptive intrusion detection of malicious unmanned air vehicles using behavior rule specifications", *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 44, no. 5, pp. 593-604, May 2014.
- [40] "Wi-Fi.", Microchip Developer Help, <http://microchipdeveloper.com/wifi:start> (accessed 12 May 2018) [Accessed: 20-Jun-2021].
- [41] Compton, Stuart: 802.11 Denial of Service Attacks and Mitigation, SANS Institute InfoSec Reading Room. [Accessed: 20-Jun-2021].
- [42] Aircrack-ng, <https://www.aircrack-ng.org/> [Accessed: 20-Jun-2021].

- [43] RPI-Wireless-Hotspot for raspberry pi to convert into router, <https://github.com/harryallerston/RPI-Wireless-Hotspot> [Accessed: 20-Jun-2021].
- [44] Robot Operating System, <http://www.ros.org/about-ros/> (accessed 16 June 2018) [Accessed: 20-Jun-2021].
- [45] WPA2 encryption, <https://github.com/daraosn/ardrone-wpa2> [Accessed: 20-Jun-2021].
- [46] Kismet wireless intrusion detection system for drone, <https://raw.githubusercontent.com/kismetwireless/kismet/master/README> [Accessed: 20-Jun-2021].
- [47] D. Kewley, R. Fink, J. Lowry, and M. Dean, "Dynamic approaches to thwart adversary intelligence gathering," in DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings, IEEE (2001), vol. 1, pp. 176–185.
- [48] J. Michalski, C. Price, E. Stanton, E. Lee, K. Chua, Y. Wong, and C. Tan, "Network security mechanisms utilizing dynamic network address translation" (2002).
- [49] J. Li, P.L. Reiher, and G.J. Popek, "Resilient self-organizing overlay networks for security update delivery," IEEE Journal on Selected Areas in Communications 22(1), pp. 189–202 (2004).
- [50] H. Moniz, N.F. Neves, M. Correia, and P. Verissimo, "Randomized intrusion-tolerant asynchronous services," in Dependable Systems and Networks, 2006. DSN 2006. International Conference on, IEEE (2006), pp. 568–577.
- [51] S. Antonatos, P. Akritidis, E.P. Markatos, and K.G. Anagnostakis, "Defending against hitlist worms using network address space randomization," Computer Networks 51(12), pp. 3471–3490 (2007).
- [52] E. Al-Shaer, "Toward network configuration randomization for moving target defense," in Moving Target Defense, Springer, pp. 153–159 (2011).
- [53] J.D. Touch, G.G. Finn, Y.S. Wang, and L. Eggert, "DynaBone: Dynamic defense using multilayer internet overlays," in DARPA Information Survivability Conference and Exposition, 2003. Proceedings, IEEE (2003), vol. 2, pp. 271–276.
- [54] "AFRL resources," Personal communication.
- [55] E. Al-Shaer, Q. Duan, and J. Jafarian, Random Host Mutation for Moving Target Defense, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 310–327 (2013).

- [56] Y.B. Luo, B.S. Wang, X.F. Wang, X.F. Hu, and G.L. Cai, "TPAH: A universal and multiplatform deployable port and address hopping mechanism," in 2015 International Conference on Information and Communications Technologies, IET (2015), pp. 1–6.
- [57] J. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: Transparent moving target defense using software-defined networking," in Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM (2012), pp. 127–132.
- [58] J. Jafarian, E. Al-Shaer, and Q. Duan, "Spatio-temporal address mutation for proactive cyber agility against sophisticated attackers," in Proceedings of the First ACM Workshop on Moving Target Defense, ACM (2014), pp. 69–78.
- [59] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in Proceedings of the 2013 ACM Conference on Computer and Communications Security, ACM (2013), pp. 413–424.
- [60] R. Skowrya and D. Bigelow, "Dynamic flow isolation: Adaptive access control to protect networks," Cyber Security Division Transition to Practice Technology Guide (2016).
- [61] Vengainathan, B., Basil, A., Tassinari, M., Manral, V., & Banks, S. (2018). Benchmarking Methodology for Software-Defined Networking (SDN) Controller Performance. *RFC*, 8456, 1-64.
- [62] C. Gudla and A. H. Sung, "Moving Target Defense Discrete Host Address Mutation and Analysis in SDN," 2020 International Conference on Computational Science and Computational Intelligence (CSCI), 2020, pp. 55-61, doi: 10.1109/CSCI51800.2020.00017.