

# **ROUTING IMPLEMENTATION BASED ON PROCESSING DELAY AND BANDWIDTH IN SOFTWARE DEFINED NETWORKING**

*A Dissertation submitted in partial fulfillment of the requirements for the  
degree*

*Of*

**Bachelor of Technology**

**In**

**Computer Science and Engineering**

*Submitted By*

**Mandeep Kumar Singh (CSB17039)**

**Gaurav Kumar (CSB17051)**

**Samir Sujan (CSB17067)**

*Under the supervision of*

**Dr. Sanjib KumarDeka**

Associate Professor



School of Engineering  
Department of Computer Science & Engineering  
Tezpur University  
Napaam- 784028, Assam, India

June 2021

## List of Figures

2.1 SDN Architecture.....	14
9.1 Example Toplogy.....	22
9.2 Dummy Flow Entry of the Switches.....	23
9.3 Structure of Dummy Flow Mod Packet.....	25
9.4 Switch Id along with Processing Delay.....	27
9.5 Utilized Bandwidth between the Links.....	29
10.1 Deployed Architecture in Mininet.....	30
10.2 Code Snippet for showing Deployed Architecture.....	30
10.3 Code Snippet for Processing the Topology.....	31
10.4 Topology Table showing the link between the Switches.....	31
11.1 Calculation of Routing Matrix.....	32
12.1 Code snippet used to find Next Hop.....	34
12.2 Output snippet showing the Next Hop.....	35
13.1 Example Topology.....	36
13.2 Snippet of the ARP Request by host of the Deployed Architecture.....	37
13.3 Snippet of the ARP Reply by the Switch to the Host of Deployed Architecture.....	37
13.4 Snippet of Status of Packets.....	38
13.5 Client Connected to Server.....	39



**Department of Computer Science and Engineering**

**Tezpur University**

**CERTIFICATE BY HEAD OF THE DEPARTMENT**

This is to certify that the project entitled “ **Routing Implementation based on Processing Delay and Bandwidth in Software Defined Networking** ” submitted by Mandeep Kumar Singh (CSB17039), Gaurav Kumar (CSB17051), Samir Sujan (CSB17067). The report is found worthy of acceptance for the award of Bachelor of Technology in Computer Science and Engineering. He has worked under the supervision of Dr. Sanjib K Deka.

Date:  
HOD, CSE



## Department of Computer Science and Engineering

### Tezpur University

---

#### Certificate By Examiner

This report titled “**Routing Implementation based on Processing Delay and Bandwidth in Software Defined Networking**” submitted by Mandeep Kumar Singh (CSB17039), Gaurav Kumar (CSB17051), Samir Sujana (CSB17067) in partial fulfilment of the requirements for the major project of Bachelor of Technology in Computer Science and Engineering has been examined.

**Examiner**

Date :

Place :Tezpur



**Department of Computer Science and Engineering**  
**Tezpur University**

---

**CERTIFICATE**

This is to certify that the report entitled “ **Routing Implementation based on Processing Delay and Bandwidth in Software Defined Networking**” is submitted by **Mandeep Kumar Singh** bearing Roll no: **CSB17039**, **Gaurav Kumar** bearing Roll no: **CSB17051**, **Samir Sujan** bearing Roll no: **CSB17067** is carried out by her under my supervision and guidance for partial fulfillment of the requirements and the regulations for the award of the degree of Bachelor of Technology in Computer Science & Engineering during the session 2017-2021 at Tezpur University. To the best of my knowledge, the matter embodied in the report has not been submitted to any other university/institute for the award of any Degree or Diploma.

Date :

Place :

Dr. Sanjib Kumar Deka  
Associate Professor  
Department of Computer Science and Engineering  
Tezpur University



## Department of Computer Science and Engineering

### Tezpur University

---

#### DECLARATION

I hereby declare that the report titled “ **Routing Implementation based on Processing Delay and Bandwidth in Software Defined Networking**” submitted to the Department of Computer Science and Engineering, Tezpur University is prepared by me and was not submitted to any other institution for award of any other degree.

Date :

Place :

Dr. Sanjib Kumar Deka  
Associate Professor  
Department of Computer Science and Engineering  
Tezpur University



## Department of Computer Science and Engineering

### Tezpur University

---

#### DECLARATION

We hereby declare that the project report titled “**Routing Implementation based on Processing Delay and Bandwidth in Software Defined Networking**” submitted to the Department of Computer Science and Engineering, Tezpur University is prepared by us and was not submitted to any other institution for the award of any other degree.

Date: 19/06/2021

Place: Tezpur

Mandeep Kumar Singh  
CSB1739  
Department of CSE  
Tezpur University

Gaurav Kumar  
CSB17051  
Department of CSE  
Tezpur University

Samir Sujan  
CSB17067  
Department of CSE  
Tezpur University

## **Acknowledgement**

I would like to extend heartfelt gratitude to my project guide Dr. Sanjib Kumar Deka, Associate Professor, Department of CSE, Tezpur University, for giving us the opportunity to work under him and providing me ample guidance and support through the course of the project. I am highly indebted to Dr. Nabojyoti Medhi, Assistant Professor, Department of CSE, Tezpur University for his helpful guidance as well as for providing necessary information regarding the project.

I would also like to thank Head of the Department of Computer Science and Engineering department and all the Faculty members of Department of CSE, Tezpur University for the valuable guidance and co-operation throughout the project.

My thanks and appreciations also go to all other people who have directly or indirectly helped me out with their abilities.



## **Abstract**

During our project we have learnt about the various methodologies and technologies of Software Defined Networks through which we have tried to implement a routing mechanism based on processing delay and utilized bandwidth between the switches. Every topic from crust to core is explained and this report is a result of what we have learned and implemented. Information in this report is gathered from different sources like project proposals, online websites and documentation.

This document gives reader an insight of Software Defined Networks. Also it provides some ideas about the technological advancement with Computer Network. Interactive visuals provide an ease to reader for understanding process and doesn't let reader to get bored as traditional technical text does. We have tried our best to eliminate all mistakes and misrepresentation of facts but since it is natural for humans to make mistakes, so we ask your pardon in advance for any such mistake.

---

## Contents

### List of Figures

### Problem Statement

<b>1. Introduction</b>	<b>12</b>
1.1 Control Plane .....	12
1.2 Data Plane.....	12
<b>2. Software Defined Networking</b>	<b>14</b>
<b>3. Advantages of SDN Approach</b>	<b>16</b>
<b>4. Infrastructural Requirements of SDN</b>	<b>17</b>
4.1 Hardware Domain.....	17
4.2 Software Domain.....	17
<b>5. Drawbacks in Traditional Network without SDN Applications</b>	<b>18</b>
<b>6. RYU Controller</b>	<b>19</b>
<b>7. Mininet</b>	<b>20</b>
<b>8. Disadvantages of Traditional Network Packet Loss</b>	<b>21</b>
8.1 Causes of Packet Loss.....	21
<b>9. Proposed Idea</b>	<b>22</b>
9.1 Calculating Processing Delay of the Switch(Delay).....	23
9.2 Calculating Utilized Bandwidth between the switch.....	28
<b>10. Topology Discovery</b>	<b>30</b>
<b>11. Routing Matrix Calculation</b>	<b>32</b>
<b>12. Optimum Routing Path</b>	<b>33</b>
12.1 Dijkstra's Shortest Path Algorithm.....	33
<b>13. Packet Flow</b>	<b>36</b>
13.1 Flow of packet from source to destination.....	36
13.2 Packet Flow from host1 to host4.....	36
<b>Bibliography.....</b>	<b>40</b>

## **Problem Statement:**

In traditional network, we use lot of parameters like bandwidth, RTT of link etc. to calculate the distance/weight between the links of two Layer 3 switches. In traditional network, the distance between two Layer 3 switches is calculated based on the status of the link without considering the status of the Layer 3 switch. As always the routing is decided based on the status of the link without considering into the fact about the congestion in the Layer 3 switches which may lead to a situation which may drop the incoming stream of packets and delay the forwarding of stream of packets.

### **Scope of our work:-**

Our main area of work is to:

1. Calculate the Processing Delay of all the switches present in the Deployed Architecture.
2. Calculate the utilized bandwidth between of the links between the all pair of switches.

The calculated processing delay and the utilized bandwidth are to be combined using a mathematical formula to determine the distance/weight of links between all the pairs of switches present in the deployed architecture for finding the optimized path from source to destination.

# Chapter 1

## **INTRODUCTION**

The conventional architecture of networks is created by the use of switches and routers which are in some sense autonomous in their working. When the switch and router connect to the network, the Network performs certain actions (like constructing a spanning tree) to avoid cycles in the network and to know about the neighboring devices. For this, they exchange certain messages which give them information about their neighbor devices and the actual topology to other devices. Thus, every device has an idea (or complete structure of topology) of the network topology in which devices are located. Every router has two functional planes:

1. Control Plane and
2. Data Plane.

### **1.1 Control Plane**

Control plane is the part of router architecture that is use for drawing the network topological and information. It's means to gather information of routing in a routing table that defines how router deal with incoming packet streams. Depending upon the router-design, there may be different entries in tables for unicast and broadcast flow mappings. Different entries have different sets of preferences for routing information, and these are not standardized among IP routers. It is fair to say that subnets on directly connected active interfaces are always preferred. There will be different Routing-protocols specify how routers collect information for the routing tables and flow streams to their neighbouring routers that enables them to select any two nodes on a computer network and send data from one node to other.

There are three major classes of routing-protocol that are widely use on IP based networks:

Link-state routing protocols (OSPF and IS-IS).

Distance-vector routing protocols (Routing Information Protocol, RIPv2, IGRP).

Exterior gateway protocols are routing protocols which is used on Internet for exchanging routing information between Autonomous Systems (Border Gateway Protocol(BGP)).

### **1.2 Data Plane**

Data plane mainly decide what to do with packets which is arrive on an inbound interface. Mostly, It refers to table where router store information about destination address of the incoming packet and retrieves the information required to determine the path of receiver so that stream can be forward through the internal routers and outgoing interface(s). Control plane and Data plane work together to define the complete working of a router. But with such complexity, this approach draws certain drawbacks which are as follows:

1. Since the routers need to discover the topology when they are connected (determine neighbouring devices, spanning-tree construction etc.), for this they need certain processing capability. Thus, they need to be high processing power. Because of this routers and switches become expensive. For detecting the loop free topology and the neighbouring nodes certain setup is time required.

2. If a certain path on spanning-tree goes down, the whole routine of identifying spanning tree will be done again. This again will require certain time expenditure. Hence some time is spent in convergence.
3. The optimization of route in conventional networks is limited. Because they require loop free topologies means they have to detect the paths in network graph which can lead to cycles.
4. For removing these shortcomings, concept of Software Defined Networks(SDN) was introduced. The current network uses the conventional networking approach. Now we would see how SDN (Software Defined Networking) can help to remove the above shortcomings.

## Chapter 2

### SOFTWARE DEFINED NETWORKING

The basic idea behind Software Defined Networking is to separate the Control Plane and Data (forwarding) Plane i.e decoupling the two planes. When the functioning of the two planes are kept independent, then we can optimize each plane independently, results the increase in efficiency. So, Let us look at the following figure:

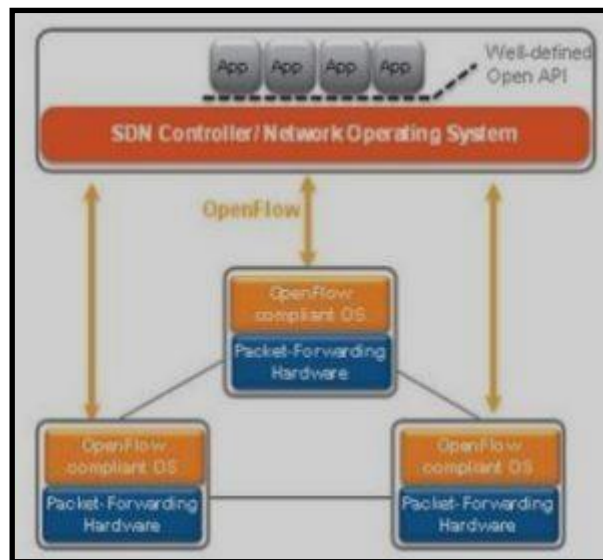


Fig 2.1: SDN Architecture

The above figure describes the functional architecture of SDN with respect to hardware and software. In other words, it describes how the control plane and data plane are arranged in the scheme. The control plane now resides at a high level and is part of a software called **controller (SDN controller)**. The hardware (switches and routers) have only the data plane. The controller is also called Southbound APIs (OpenFlow APIs) which is used to communicate with the SDN-switches and manages their data forwarding plane. Controller has also Northbound APIs (application APIs), which are used for communicating with controller and application running over the network.

We can also say that controller acts like an interface between the switches and applications. There are certain type of OpenFlow messages which are used for the interaction between switch and controller. Some of which are as follows:

**a) Controller-to-switch messages:** Handshake, Switch Configuration, Flow Table Configuration, Modify State Messages, Queue Configuration Messages, Read State Messages, Packet Messages.

***b) Asynchronous messages:*** Packet-In Message, Flow Removed Message, Port Status Message, Error Message.

***Symmetric messages:*** Hello, Echo Request, Echo Reply, Experimenter.

## Chapter 3

### **ADVANTAGES OF SDN APPROACH**

Software Defined Networking offers the following advantages over the conventional networking approach:

- As we know the control plane is decoupled from the data plane, the switches/routers become very simple in terms of functionality. They need to forward a packet to a particular port based upon the flow installed in the flow table by the controller inside the switch.
- The Switches/Routers needs a trivial multiplexing circuit to carry out the packet forwarding. There is no requirement for significant computation power as in Conventional Networks.
- Since, the controller needs to know about the topology, significant amount of time can be saved, as the nodes in the network isn't required to be to discovered all the other nodes inside the network except their neighbours.
- Optimization of both planes possible independently.
- The paths between two nodes need not be pruned to avoid the cycles. In fact, they can also be used to increase the throughput of the network.
- This approach is easy to scalable and complicated configuration is not required. Most importantly, centralizing the control plane which allows forwarding decisions to be made globally across the SDN domain rather than at each hop.



## Chapter 4

### **INFRASTRUCTURAL REQUIREMENTS OF SDN**

In this section, we cover the necessary changes which are the basic need for deployment of Software based networking approach. With reference to architecture of SDN approach (see figure 1), we infer that we need to identify the changes in two domains which are as follows:

#### **4.1 Hardware Domain:-**

In this domain, we need to identify switches which gives access to their forwarding tables by means of well-defined APIs, as normal switches do not provide interface to access and configure their tables. This need is met by the use of OpenFlow compliant switches (in other words OpenFlow enabled switches). These switches are different from the normal switches, as they allow access as well as manipulation of entries in their data forwarding tables via OpenFlow APIs. This is same as defining flow rules in the switch, based on which, the switch multiplexes the packets. Baremetal switches are a new breed of highly customizable switches which offer flexibility at a very low cost.

#### **4.2 Software Domain:-**

The Hardware-Software domain interacts, via OpenFlow communication protocol. SDN controllers or controllers are softwares that are used to manage and control the flow and data entries in the forwarding tables of the switches. Increasingly, these softwares are being developed as network operating systems which perform multiple tasks of managing and monitoring the status of hardware and applications running on the network.

## Chapter 5

### **DRAWBACKS IN TRADITIONAL NETWORK WITHOUT SDN APPLICATIONS**

1. Traditional networks can not at all have these two kinds of planes i.e. (Control Plane and Data Plane).
2. The applications in traditional network are indirectly represent by the application request rather to the network layer and while processing the request it may take too much of time which needs much of the multiple human processing steps.
3. Traditional networks don't provide the dynamic features and that means these networks don't provide the dynamic requirements of user can't be scaled up and down in the Network.
4. Traditional networks don't give better throughput, delay, jitter and delay variation in the the transmission of the data.
5. Non SDN networks don't provide any network information and the state of application using such information. SDN application can monitor promptly the devices of the network and their resource usages.

## Chapter 6

### **RYU CONTROLLER**

Ryu Controller is an Open, Software-defined networking (SDN) Controller designed to increase the agility of the network which makes it easy to manage and adapt how traffic is handled. Generally, the SDN Controller acts the brain of the SDN environment, which helps in communicating information down to the switches with southbound APIs, and up to the applications and business logic with northbound APIs.

The Ryu Controller provides software components, with well-defined application program interfaces (APIs), that makes it easy for developers to create new network management and controlling applications. It's approach also helps the organizations to customize deployments to meet their specific needs, developers can quickly and easily modify existing components and can implement their own underlying network which can meet the changing demands of their applications.

There are various inbuilt applications in RYU and RYU Manager listens on Openflow ports (6653) and are in listening states. RYU applications are in python script and they can run in multiples on RYU in a single initiation.

Some RYU controller command lines are :-

- RYU-manager—help # To know all the available options
- RYU-manager--verbose # To enable the debug logs
- RYU-manager--ofp-tcp-listen-port6634 # To use custom openflow port number
- RYU-manager--observe-links # To use Topology Discovery

The RYU Controller interacts with the forwarding plane in order to modify how the network will handle traffic flows using protocols such as OpenFlow.

## Chapter 7

### MININET

Mininet is totally an open source software package that can be used to simulate a software defined network (SDN) and its compatible controllers, layer 3 switches and the hosts. By default, Mininet provides OVS switches and controllers. It also supports other SDN controllers and switches instead of the default present already in it. It also supports the Openflow protocol which provides an interface between the two planes i.e. (control plane and the data forwarding plane).

#### Commands :-

- a) mininet>nodes # helps to show the nodes present inside the network
- b) mininet>net # helps to show and define the links present in the network
- c) mininet>dump # helps to show the IP addresses and the PIDs of the nodes
- d) mininet> h1pingh2 # used to ping a from a particular host to a targeted host
- e) mininet> h1ifconfig-a # helps to show the address info of the nodes
- f) mininet>pingall # it's a command to test the connectivity among hosts
- g) mininet> link s1h1 down # command to down a link

We can start Mininet using the “**sudo mn command**” with no CLI arguments, creates a simple network of two hosts and one switch, h1–s1–h2, and starts the Mininet command-line interface (CLI). In Mininet by convention the host name begin with ‘h’ and the switch name begin with ‘s’.

Some advantages of Mininet include faster loading time, large scalability and providing more bandwidth. Even then, this large giant hub of a tool is not without it's limitations. Mininet based networks can't exceed the CPU or bandwidth available yet. Also Mininet requires the Linux OS, without which it will be unable to run OpenFlow switches or applications.

## Chapter 8

### **DISADVANTAGES OF TRADITIONAL NETWORK PACKET LOSS**

While accessing the network chunks of data units called packets are being received and sent. When any of the packets fails to reach its defined destination this term is called packet loss. For users, packet loss displays itself in the form of network disruption also the slow service and even the total loss of the network connectivity. Any application can be easily disrupted by “packet loss”, but mostly the applications that rely on real-time packet processing, online gaming, online transaction etc.

#### **8.1 Causes of Packet Loss :-**

##### **a) Buffer Congestion**

Network congestion occurs when a particular network becomes totally congested with traffic and hit the maximum capacity. Packets need to wait for their turn to be delivered, but if anyhow the connection falls and it cannot store any more packets, they will simply be discarded.

##### **b) Software Bugs**

It is another common cause of packet loss. If the rigorous testing is not carried out or bugs have been introduced due to the following software updates, this could result in an unexpected network behavior. Rebooting can sometimes resolve these issues, until the software is updated or some new patch is added.

##### **c) Problems with Network Hardware**

The faulty or sometimes the outdated network hardware such as network switches, firewalls and routers can easily slow down network traffic. As a company grows, it starts to experience packet loss, lag and sometimes total connectivity drops. The hardware needs to be updated and revised so that it can manage the growing network traffic or throughput.

##### **d) Security Threats**

Packet loss can also be easily caused by the security breach. The attack that has become popular within recent years is the packet drop attack. During this attack, a unnamed or malicious user takes total control of the router and sends various types of commands that easily drop packets into the stream of data. If the rate of packet loss across the network is very high, it could be a cyber- attack in progress.

## Chapter 9

### PROPOSED IDEA

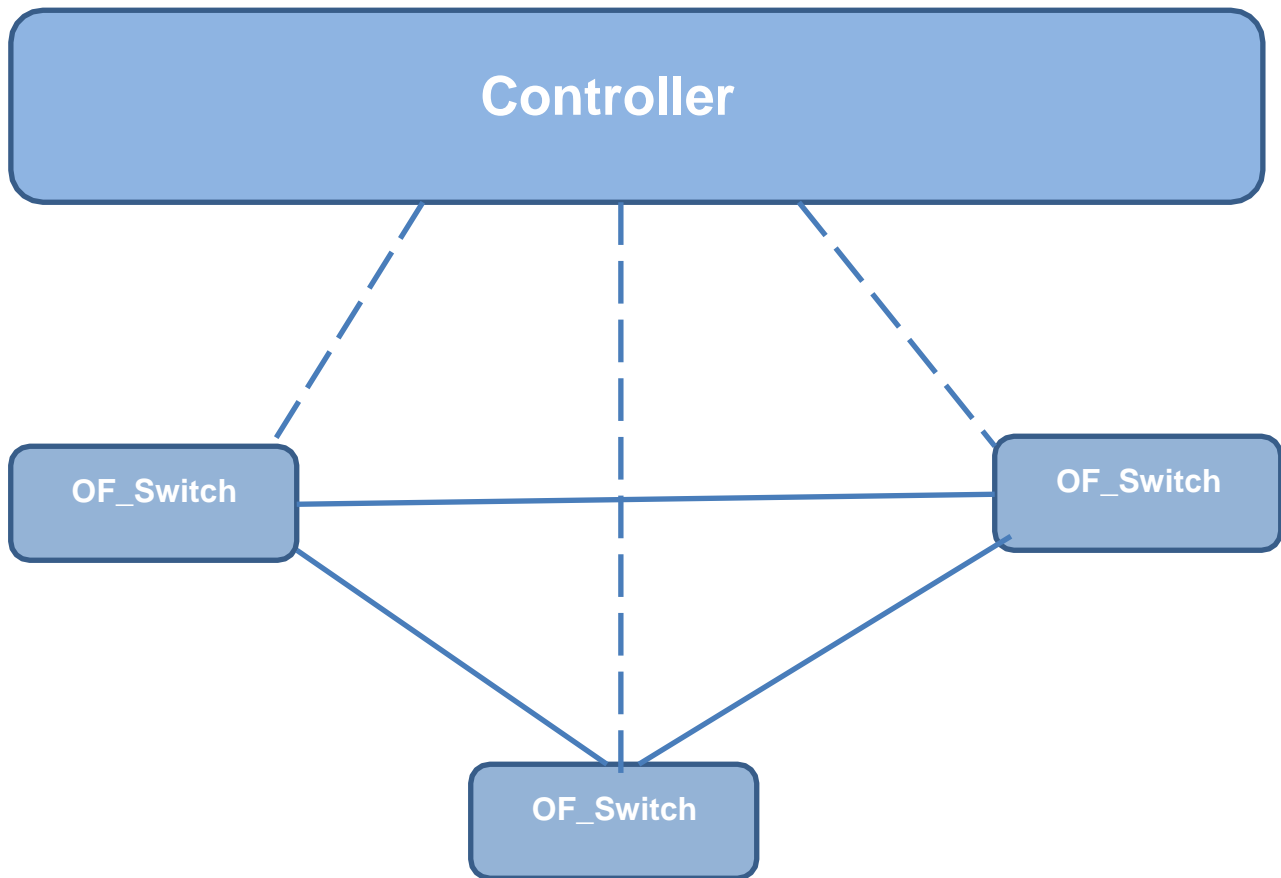


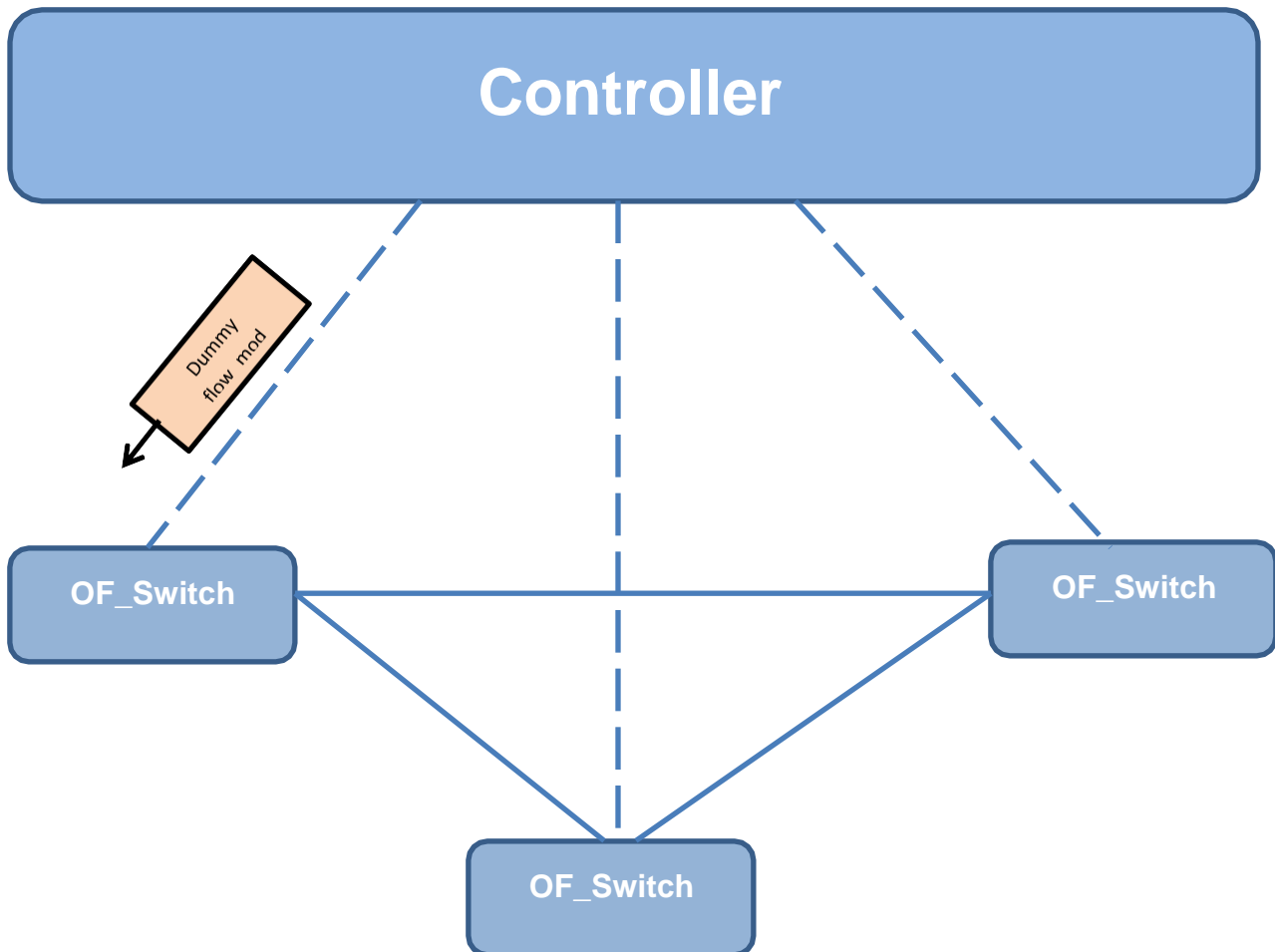
Fig 9.1: Example Topology

The parameters involved are:

- a) Congestion of the switch (DELAY)
- b) Remaining Bandwidth between the Switches

## 9.1 Calculating Processing Delay of the Switch(DELAY):

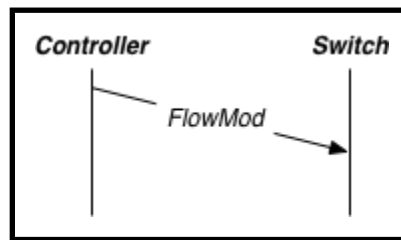
Step 1:



```
FLAG 1 BAND_TABLE updated for switch 1 port 5 recved byte 49589 at time 1624111520.74
Dummy flow entry removed from switch 2 at time 1624111520.74
Dummy Flow entry Added at switch 2 at time 1624111520.75
{'1': 4, '3': 4, '2': 0, '5': 4, '4': 4, '7': 4, '6': 4, '9': 4, '8': 4}
Dummy flow entry removed from switch 5 at time 1624111520.77
Dummy Flow entry Added at switch 5 at time 1624111520.78
{'1': 4, '3': 4, '2': 0, '5': 0, '4': 4, '7': 4, '6': 4, '9': 4, '8': 4}
Dummy flow entry removed from switch 1 at time 1624111520.79
Dummy Flow entry Added at switch 1 at time 1624111520.8
{'1': 0, '3': 4, '2': 0, '5': 0, '4': 4, '7': 4, '6': 4, '9': 4, '8': 4}
Dummy flow entry removed from switch 7 at time 1624111520.81
Dummy Flow entry Added at switch 7 at time 1624111520.82
{'1': 0, '3': 4, '2': 0, '5': 0, '4': 4, '7': 0, '6': 4, '9': 4, '8': 4}
Dummy flow entry removed from switch 6 at time 1624111520.84
Dummy Flow entry Added at switch 6 at time 1624111520.85
{'1': 0, '3': 4, '2': 0, '5': 0, '4': 4, '7': 0, '6': 0, '9': 4, '8': 4}
Dummy flow entry removed from switch 3 at time 1624111520.86
Dummy Flow entry Added at switch 3 at time 1624111520.88
{'1': 0, '3': 0, '2': 0, '5': 0, '4': 4, '7': 0, '6': 0, '9': 4, '8': 4}
FLAG 1 BAND_TABLE updated for switch 2 port 1 recved byte 49589 at time 1624111520.89
```

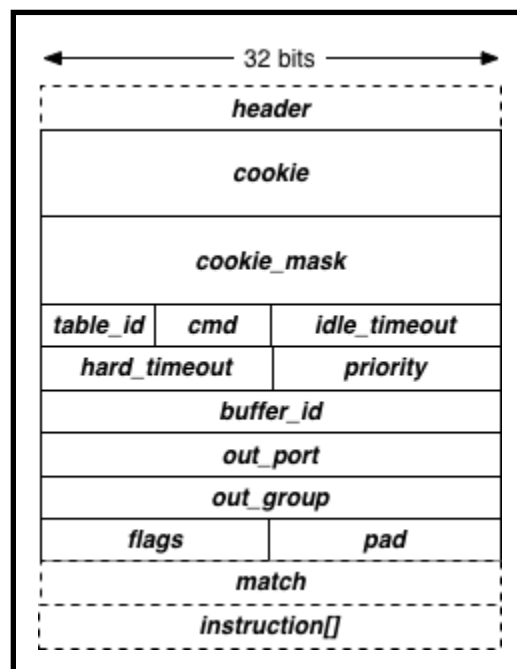
Fig 9.2: Dummy Flow Entry of the Switch of the Deployed Architecture

### A) Flow Mod:



This message allows the controller (RYU) to easily modify the state of an OpenFlow switch. The Flow\_Mod message begins with the defined standard OpenFlow header which contains the version of OpenFlow and type values which is followed by the FlowMod structure.

### B) Structure of Flow Mod:



Flow\_Mod message starts with the standard header followed by cookie, which is being set by the controller, its table, command and mask which specifies the type of flow table modification which needs to be done. This is followed by priority, hard\_timeout and idle\_timeout. The priority field defines an order for matches that overlaps with higher numbers i.e. representing higher priorities. Idle\_timeout and hard\_timeout represent number of seconds since packet inactivity and creation since expiration, respectively. Next the FlowMod contains buffer\_id, out\_port, out\_group and flags. Buffer\_id is the id of the buffered packets that created a packet\_in inside the switch, make FlowMod and then reprocess the packet. Finally, 2 byte padding, match and instructions are also added at the end.



### C) Structure of Dummy Flow Mod Packet

The following snippets shows how the header is being modified according to our requirement which shows how the flow\_mod\_entry is being installed in the switch:

```
def add_flow_2(self, datapath, priority, match, actions, buffer_id=None):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                         actions)]

    if buffer_id:
        mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id,
                                priority=priority, match=match,
                                instructions=inst, idle_timeout=0x20, flags=0x0001)
    else:
        mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                                match=match, instructions=inst, idle_timeout=0x20, flags=0x0001)

    datapath.send_msg(mod)

SimpleSwitch13
@set_ev_cls(ofp_event.EventOFPacketIn, MAIN_DISPATCHER)
def _dummy_packet_in(self, ev):
    if ev.msg.msg_len < ev.msg.total_len:
        self.logger.debug("packet truncated: only %s of %s bytes",
                          ev.msg.msg_len, ev.msg.total_len)

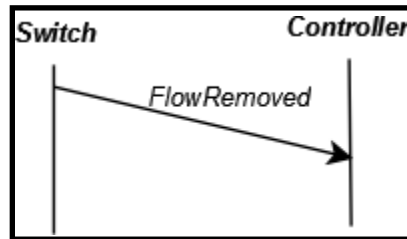
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
    # self.logger.info("\n----| DATA PATH |---- %s", datapath)
    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocols(ethernet.ethernet)[0]

    #if eth.ethertype == ether_types.ETH_TYPE_LLDP:
    #    # ignore lldp packet
    #self.logger.info("packet in %s \n", msg)
    for i in (10000):

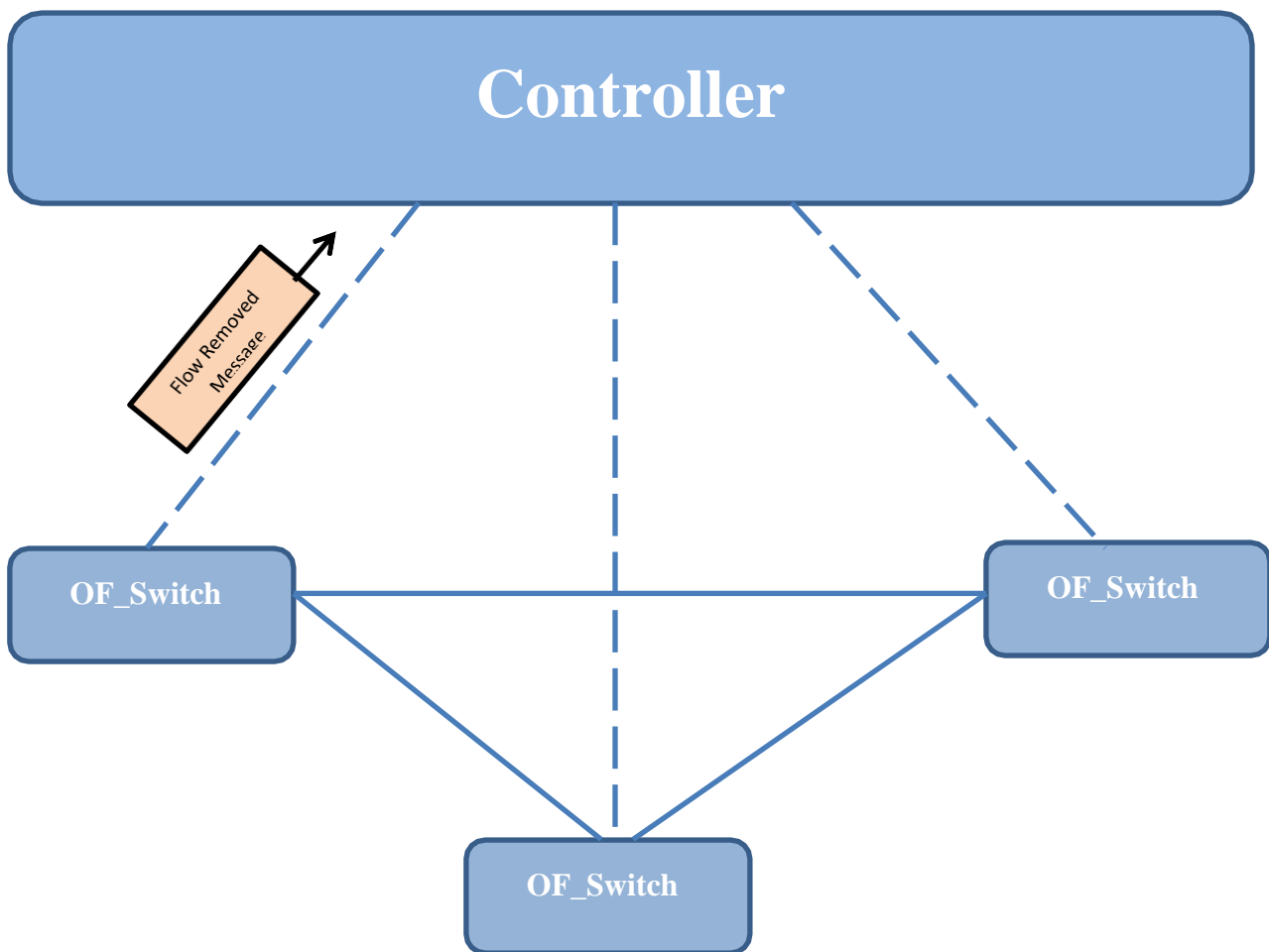
        dst = eth.dst
        src = eth.src
        if dst == '33:33:00:00:00:16':
            dst='66:35:34:35:38:39'
            src='66:35:34:35:38:40'
            actions = []
            match = parser.OFPMatch(eth_dst=dst, eth_src=src)
        # self.logger.info("\n----| DUMMY FLOW ENTRY |---- \nSOURCE - %s DESTINATION - %s", src, dst)
        self.add_flow_2(datapath, 1, match, actions)
```

Fig 9.3: Structure Of Dummy Flow Mod Packet

Step 2:



FlowRemoved is a message which is sent to the controller by the switch when a particular flow entry in a flow table gets removed/deleted. It happens after a particular timeout occurs, it maybe due to inactivity or hard timeout. The switch sends a FlowRemoved message after a particular timeout(idle timeout/hard timeout) is specified by the FlowMod. The removal of flow entry with a FlowMod message from the controller can lead to a FlowRemoved message as shown below:



## D) PROCESSING DELAY CALCULATION

a. Installing dummy flow\_mod\_packet i.e. a modified packet structure in the switch which will be sent from Controller.

The modified fields are as follows:

- Configuring timeout according to our need
- Flag bit as 0x0001 is set (helps in receiving flow\_removed message).  
In datapath field the Destination MAC Address is set. It is set so that we can understand the packet sent is dummy\_flow\_mod packet.

b. The moment a controller sends a dummy\_flow\_mod packet TIMER is started and is stored using some data structure w.r.t. to dpid of the switch.

c. After the TIMEOUT the flow gets deleted from the flow\_table and a FLOW\_REMOVE message is sent.

d. After receiving FLOW\_REMOVE message the TIMER gets stopped and delay calculation will be done as shown below:

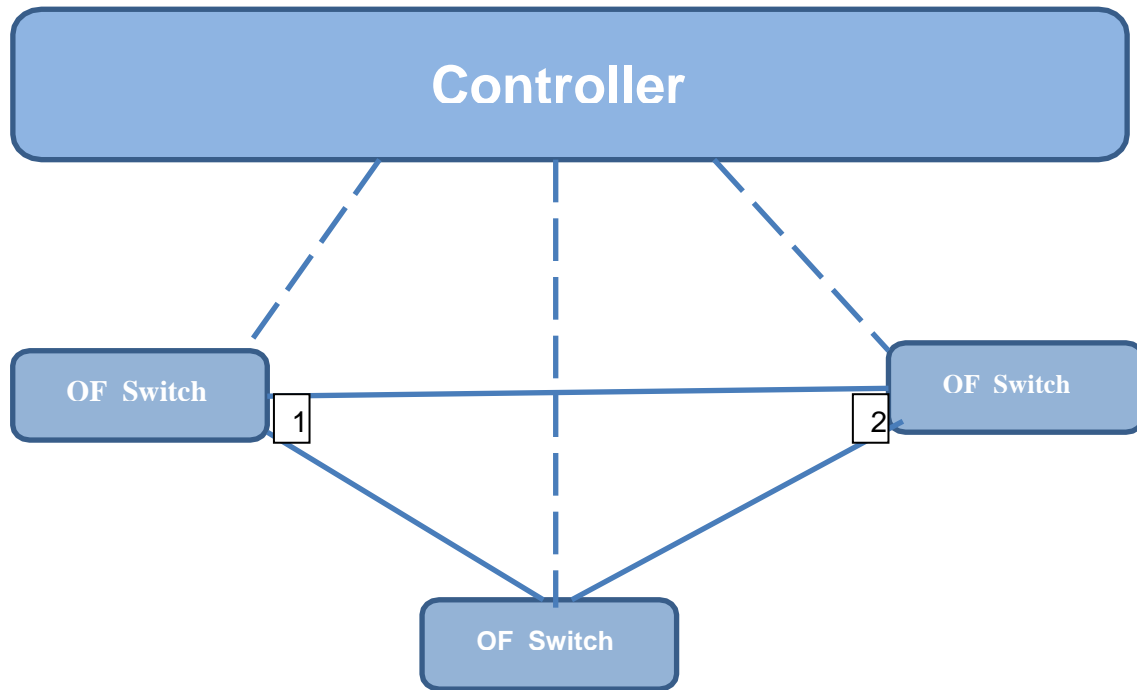
$$\text{DELAY} = \text{SEND\_TIME}(\text{Dummy\_Flow\_Mod}) - \text{ARRIVAL\_TIME}(\text{Flow\_Remove}) - \text{TIMEOUT}$$

The above steps will be repeated in some interval for every switch and average will be taken as final delay for every switch.

id	delay
switch6	18.160184025764465
switch7	18.174781441688538
switch4	13.642531991004944
switch5	13.625614941120148
switch3	13.641210734844208
switch2	18.19724988937378
switch8	18.16545569896698
switch1	13.637711346149445

Fig9.4: Switch id along with the Processing delay

## 9.2 Calculating Utilized Bandwidth between the Switches:



Step 1:

In this we will be storing the transmitted bytes at all the ports and time at which transmitted bytes are recorded of the Switch and will be stored it in the relational table of the database.

Step 2:

After a particular time interval we will be again storing the transmitted bytes at all the ports and time at which transmitted bytes are recorded of the Switch and will be stored it in the relational table of the database.

Step 3:

In this step we will be calculating the time difference of the updation of transmitted bytes and it will be stored in the different field of the same relational table of the database.

Step 4:

In this step we will be calculating the transmitted byte difference between that particular interval by subtracting from the fixed bandwidth of the link and will be stored in the different field known as byte/time of the same relational table of the database.

sid	tx1	tx1_entry_time	tx2	tx2_entry_time	port	total	diff_tx
1	5976	1624031264.4215934	5976	1624031147.6768363	1	31.905338525772095	5976
1	4140	1624031264.4265296	4140	1624031147.7066407	2	31.935142993927002	4140
1	0	1624031264.4307694	0	1624031147.7301452	4294967294	31.95864748954773	0
3	5836	1624031115.7388363	5836	1624031179.68908	1	0.006766319274902344	5836
3	4210	1624031115.7633002	4210	1624031179.6931424	2	0.010828733444213867	4210
3	0	1624031115.7714977	0	1624031179.7153196	4294967294	0.03300595283508301	0

Fig 9.5: Utilized bandwidth between the links

## Chapter 10

### Topology Discovery

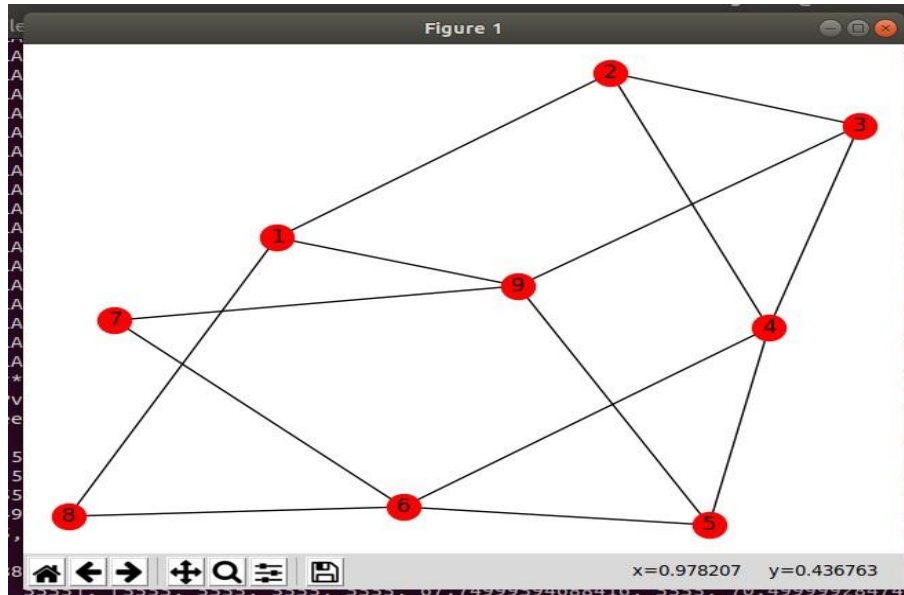


Fig 10.1: Deployed Architecture in Mininet

```
import networkx as nx
import mysql.connector
import matplotlib.pyplot as plt
G=nx.Graph()

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="SDN"
)
mycursor=mydb.cursor()
sql1="SELECT S1,S2 FROM topotable;"
mycursor.execute(sql1)
records= mycursor.fetchall()
print(records)
g=[]

G=nx.Graph()
G.add_edges_from(records)
print(G.nodes())
nx.draw(G,with_labels=True)
plt.show()
```

Fig 10.2: Code Snippet for showing the Deployed Architecture

```

# Switches topology information stored in database toptotable(Table Name)

@set_ev_cls(event.EventSwitchEnter)
def handler_switch_enter(self, ev):
    self.topo_raw_links = copy.copy(get_link(self, None))

    """
    Now you have saved the links and switches of the topo. So you could do all sort of stuff with them.
    """

    #self.logger.info(topo_raw_switches)
    #print("\t" + "Current Links:")
    mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="SDN"
    )

    mycursor=mydb.cursor()

    for l in self.topo_raw_links:
        #self.logger.info("%s",l)
        temp = re.findall(r'\d+', str(l))
        res = list(map(int, temp))
        # print("Switch with dpid "+str(res[0])+ " having port no "+str(res[1])+ " connected to switch dpid "+str(res[2])+ " having port no "+str(res[3]) )

        sql="INSERT INTO toptotable(s1,s1_p,s2,s2_p) VALUES ('%s','%s','%s','%s');"%(str(res[0]),str(res[1]),str(res[2]),str(res[3]))
        mycursor.execute(sql)
        mydb.commit()

```

Fig 10.3: Code snippet for processing the topology

+ Options			
s1	s1_p	s2	s2_p
7	1	6	3
4	4	6	1
5	2	6	2
5	1	4	3
6	3	7	1
6	1	4	4
6	2	5	2
4	3	5	1
7	1	6	3
4	4	6	1
5	2	6	2
5	1	4	3
6	3	7	1
6	1	4	4
6	2	5	2
4	3	5	1
7	1	6	3
4	4	6	1
5	2	6	2
5	1	4	3
6	3	7	1

Fig 10.4: Topology table showing the link between the switches

# Chapter 11

## Routing Matrix Calculation

As far we have calculated the Processing delay of each and every switch of the topology and the remaining bandwidth of the links between the switches. The next part is to calculate the weight of each of the links using the processing delay and the remaining bandwidth of the link which will help to form the weighted routing matrix.

The routing matrix will be calculated with the help of processing delay of each and every switch and the remaining bandwidth of the link between the switches according to the mathematical formula.

Mathematical Formula:

$$\text{Distance} = \text{Switch Delay} * 1000 + \text{Utilized Bandwidth}$$

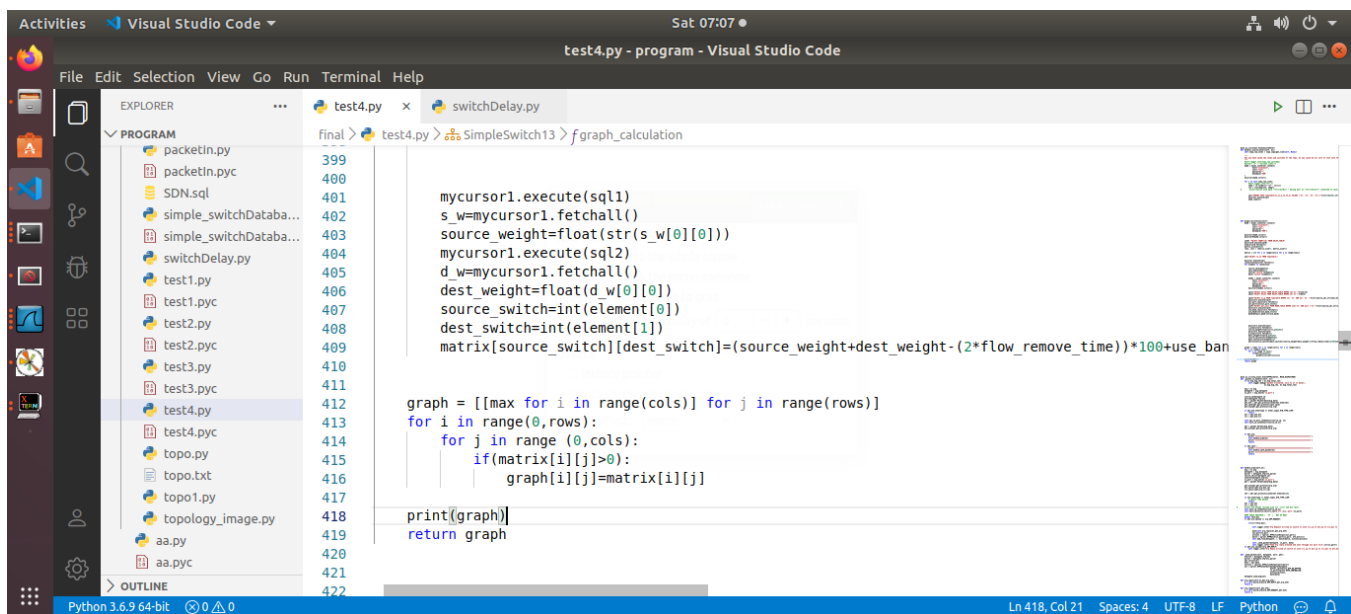
The image is a screenshot of the Visual Studio Code editor. The top status bar shows 'Sat 07:07'. The title bar indicates 'test4.py - program - Visual Studio Code'. The Explorer sidebar on the left shows a project structure with files like 'packetin.py', 'SDN.sql', 'simple\_switchData...', 'switchDelay.py', and 'test4.py'. The main editor area displays a Python script. The script uses a database cursor to fetch source and destination weights and delays, then calculates a weighted distance for each link. It constructs a graph matrix and returns it. The bottom status bar shows 'Python 3.6.9 64-bit', 'Ln 418, Col 21', 'Spaces: 4', 'UTF-8', 'LF', and 'Python'.

Fig 11.1: Code Snippet showing the Calculation of Routing Matrix



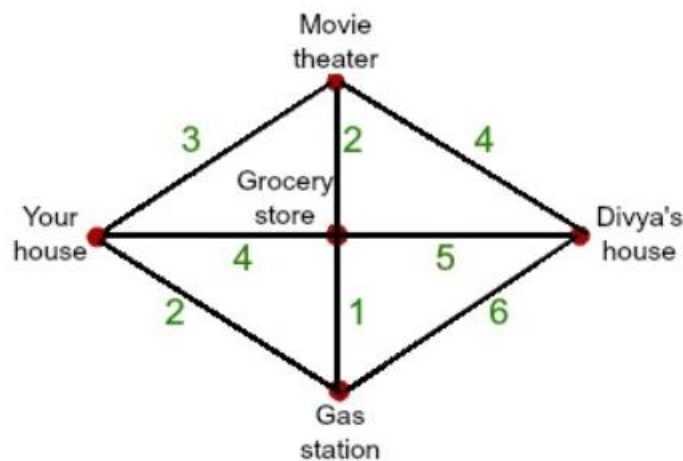
## Chapter 12

### Optimum Routing Path

As we have calculated the routing matrix for the provided custom topology the next part is to find the optimum route to transfer a packet from a particular source to a destination. The shortest path algorithm which we used to find the optimum route is Dijkstra's Algorithm.

#### **12.1 Dijkstra's Shortest Path Algorithm**

Dijkstra's algorithm is an algorithm we can use to find shortest distances or minimum costs depending on what is represented in a graph. We're basically working backwards from the end to the beginning, finding the shortest leg each time. The steps to this algorithm are as follows:



Step 1:

We are starting at the ending vertex by marking it with a distance of 0, because it is 0 units from the end. Then, we will be calling this vertex as our current vertex.

Step 2:

We are identifying all of the vertices that are connected to the current vertex with an edge. Then, we will calculate their distance to the end by adding the weight of the edge to the on the current vertex. Each of the vertices will have their corresponding distance, but we will only change the vertex's mark if it's less than a previous vertex mark. Each time we are marking the starting vertex, and keeping track of the path that resulted in that mark.

Step 3:

Here, we labeled the current vertex as visited by putting an X over it and once a vertex gets visited, we won't visit it again.

Step 4:

Then, we are finding the vertex with the smallest mark, and making it as current vertex. Now, we can start this again from step 2.

Step 5:

Once we have labeled the beginning vertex of the topology as visited - stop. The distance of the shortest path will be the mark of the starting vertex, and the path will be resulted as the shortest path.

Using Dijkstra's Algorithm we try to find the optimum route as well as in the meantime we also discover the next switch i.e. next hop along with its switch id and the port number through which the next hop can be reached.

```
def find_next_hop_out_port(self,src_ip,dst_ip,source_id):

    graph=self.graph_calculation()
    mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="SDN")
    crsr = mydb.cursor()

    sql="SELECT switch FROM HostConnection WHERE ip='%s';"%(dst_ip)
    crsr.execute(sql)
    switch_dst_id=crsr.fetchall()
    dest_id=int(switch_dst_id[0][0])
    #source_id=1
    #dest_id=2

    next_hop=self.dijkstra(graph,int(source_id),dest_id)
    print("next hop is %s ",next_hop)
    if(next_hop!=-1):
        sql1="SELECT s1_p FROM toptable WHERE s1= '%s' AND s2= '%s' ;"%(str(source_id),str(next_hop))
        crsr.execute(sql1)
        out_port_=crsr.fetchall()
        out_port=int(out_port_[0][0])
        return [out_port,next_hop]
    else:
        return [-1,-1]
```

Fig 12.1: Code snippet used to find the next hop

```
Activities Terminal Sat 07:05 gaurav@ubuntu: ~/program/final
File Edit View Search Terminal Help
*****
Arp Request arrived at switch 1 with src_ip 10.0.0.1 dst_ip 10.0.0.7 src_mac 00:00:00:00:00:01 dst_mac fe:ee:ee:ee:ef
Fake Arp reply Created and sent through out port 4
*****
Arp Request arrived at switch 1 with src_ip 10.0.0.1 dst_ip 10.0.0.7 src_mac 00:00:00:00:00:01 dst_mac fe:ee:ee:ee:ef
Fake Arp reply Created and sent through out port 4
*****
Arp Request arrived at switch 1 with src_ip 10.0.0.1 dst_ip 10.0.0.7 src_mac 00:00:00:00:00:01 dst_mac fe:ee:ee:ee:ef
Fake Arp reply Created and sent through out port 4
*****
IPv4 and TCP packet arrived at switch 9 with identification_no 0 src_ip 10.0.0.1 dst_ip 10.0.0.7 eth_src 00:00:00:00:00:01 eth_dst fe:ee:ee:ee:ef
('next hop is %s ', 3)
Tcp packet enter at switch 9 and destination switch is 3 next switch 3 forwarded to port 2:
Tcp packet enter at switch 9 and destination switch is 3 and next switch 3 forwarded to port 2:
flow entry added in switch id 9
*****
Dummy flow entry removed from switch 1 at time 1624110716.3
Dummy Flow entry Added at switch 1 at time 1624110716.31
{'1': 1, '3': 0, '2': 0, '5': 0, '4': 0, '7': 0, '6': 0, '9': 0, '8': 0}
```

**Next hop from the current Switch**

Fig 12.2: Output snippet showing the next hop

## Chapter 13

### Packet Flow

#### 13.1 Flow of packet from source to destination

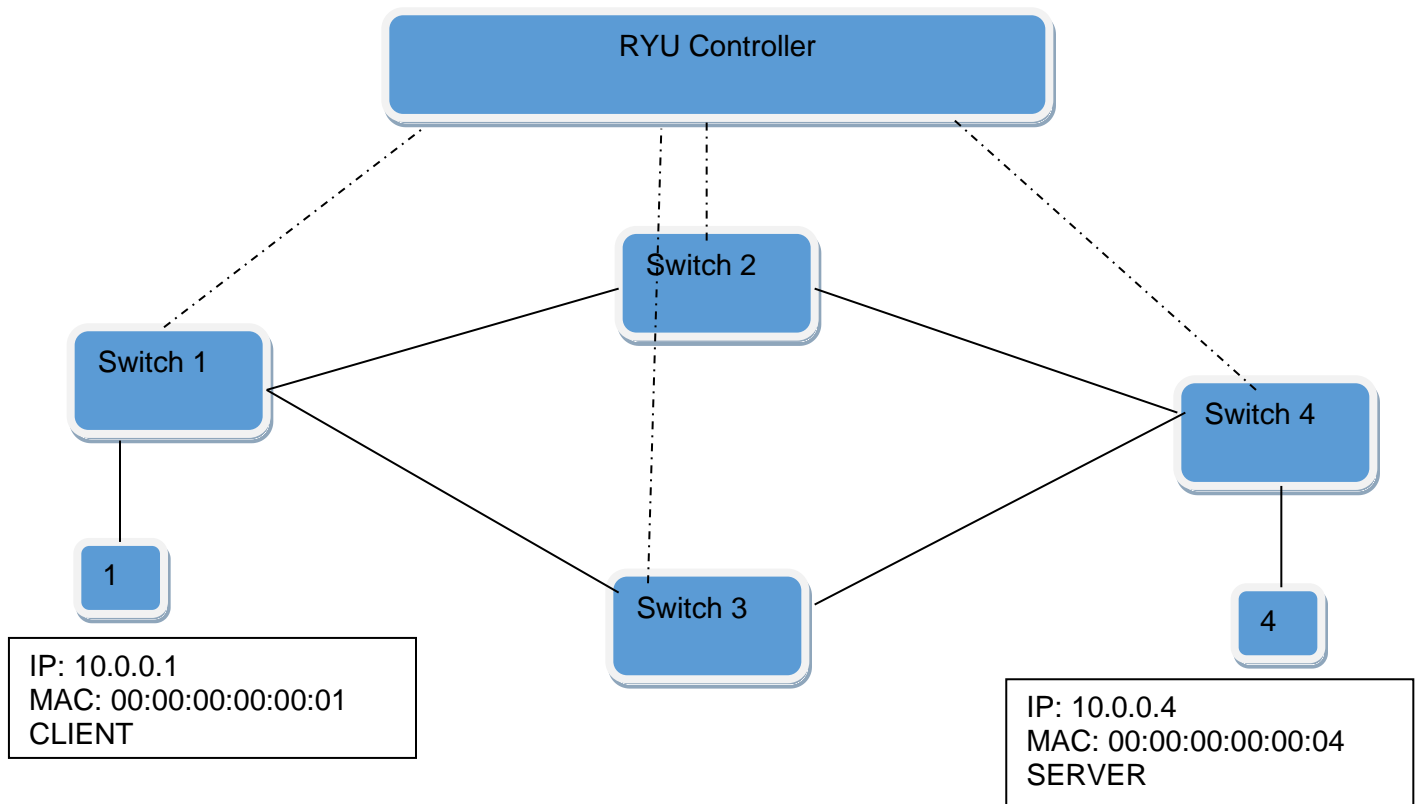


Fig 13.1: Example Topology

#### 13.2 Packet Flow from host1 to host4

We have taken an example topology as shown above to explain the flow of packet between the two host. As per the example one of the host will be acting as a SERVER and the other as a CLIENT

##### **Steps Involved:**

Step 1:

In the above example, the CLIENT IP :10.0.0.1 and the SERVER IP: 10.0.0.4. For connecting client to the server the host h1 will send ARP (Address Resolution Protocol) request having the destination set to IP Address: 10.0.0.4



Step 3:

After that Host1 will be sending a TCP packet to Switch1 (impersonating as Host4). After receiving the TCP packet Switch1 will be handling the packet.

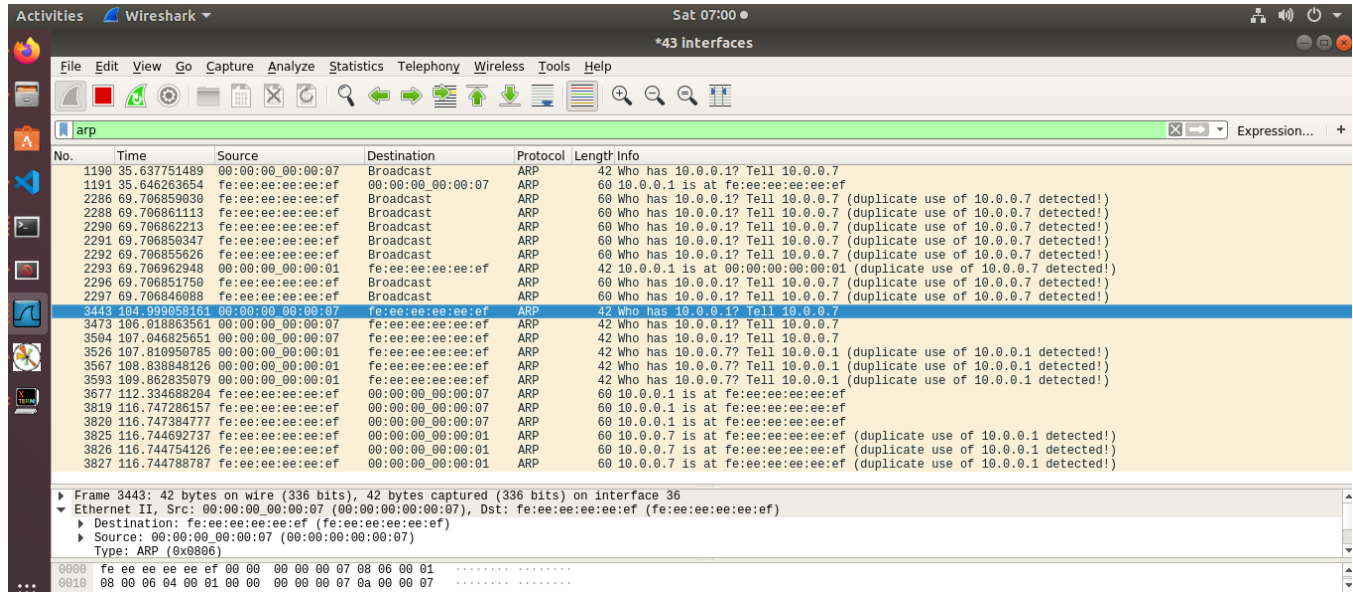


Fig 13.4: Snippet of Status of Packets

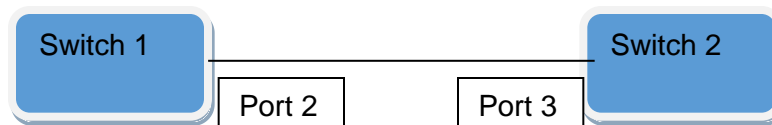
Step 4:

In this step the Switch1 will be finding the next hop and out\_port using Dijkstra's Algorithm by analyzing the topology (as shown in section 12.1).

Step 5:

After that the Switch1 will be inserting a flow entry in the flow table of itself so that in future the next TCP packets referring to same destination can be transferred accordingly.

Let's assume the next hop is Switch2 then Switch1 will be sending the TCP packet to Switch2 without modifying the TCP packet further.



In the same way Switch2 will be sending the TCP packet to Switch4.

Step 6:

In this step as the TCP packet reaches the Destination switch i.e. Switch4 and when the Switch4 tries to find the next hop and out port using Dijkstra's Algorithm it will report that it's the final destination. After reaching the final destination the TCP packet tries to find the Destination Host i.e. Host4. It will be done by searching the host\_dictionary containing the information of all the host's connected with that particular switch i.e. Switch4.

If the entry is not present in the host\_dictionary then the switch i.e. Switch4 will broadcast ARP request with

SOURCE = FAKE MAC Address  
DESTINATION = BROADCAST\_MAC

Step 7:

If host is connected with the Switch, ARP reply comes to switch and the host\_dictionary gets modified.

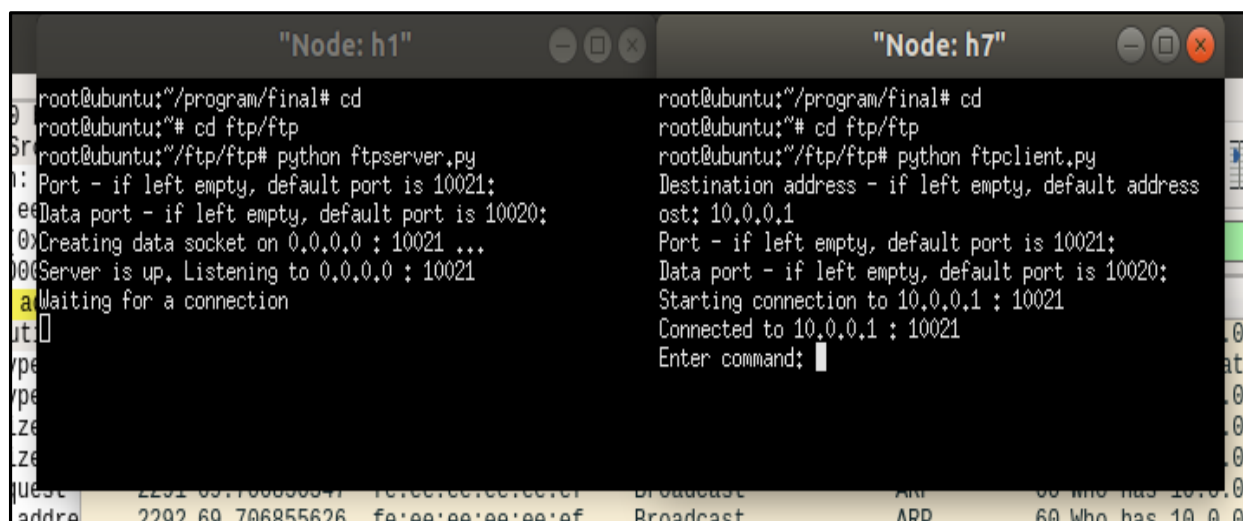
Step 8:

After the modification of the host\_dictionary, we will modify the TCP\_ethernet packet.

Modification Done:

SOURCE\_MAC = FAKE\_MAC Address  
DESTINATION\_MAC = DESTINATION\_HOST\_MAC Address

Also we insert a FLOW\_MOD i.e. entry will be inserted in the Flow\_Table accordingly and then the TCP packet will be forwarded to the out\_port and the TCP packet reaches the destination i.e. Host4 successfully. So, that's how the client gets connected with the server



```
root@ubuntu:~/program/final# cd
root@ubuntu:~/program/final# cd ftp/ftp
root@ubuntu:~/program/final/ftp# python ftpserver.py
Port - if left empty, default port is 10021:
Data port - if left empty, default port is 10020:
Creating data socket on 0.0.0.0 : 10021 ...
Server is up. Listening on 0.0.0.0 : 10021
Waiting for a connection

root@ubuntu:~/program/final# cd
root@ubuntu:~/program/final# cd ftp/ftp
root@ubuntu:~/program/final/ftp# python ftpclient.py
Destination address - if left empty, default address is: 10.0.0.1
Port - if left empty, default port is 10021:
Data port - if left empty, default port is 10020:
Starting connection to 10.0.0.1 : 10021
Connected to 10.0.0.1 : 10021
Enter command:

2251 05:10:00:00:00:00 08:00:00:00:00:00 Broadcast ARP 60 Who has 10.0.0.1
```

Fig 13.5: Client connected to Server

## **Bibliography**

<http://mininet.org/>

<https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec- v1.0.0.pdf>

<https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec- v1.3.0.pdf>

[https://www.researchgate.net/publication/319550243\\_A\\_Review\\_Paper\\_on\\_Software\\_Defined\\_Networking](https://www.researchgate.net/publication/319550243_A_Review_Paper_on_Software_Defined_Networking)

<https://www.hindawi.com/journals/ijrc/2019/7363901/>

[https://e-tarjome.com/storage/panel/fileuploads/2019-07-10/1562749634\\_E11438-e-tarjome.pdf](https://e-tarjome.com/storage/panel/fileuploads/2019-07-10/1562749634_E11438-e-tarjome.pdf)

<https://www.ijert.org/research/intelligent-network-bandwidth-allocation-using-sdn-software-defined-networking-IJERTV4IS070088.pdf>

[http://article.nadiapub.com/IJGDC/vol9\\_no1/3.pdf](http://article.nadiapub.com/IJGDC/vol9_no1/3.pdf)

<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6834762>

<https://osrg.github.io/ryu-book/en/Ryubook.pdf>

<https://ieeexplore.ieee.org/document/7495433>

[https://ryu.readthedocs.io/en/latest/getting\\_started.html](https://ryu.readthedocs.io/en/latest/getting_started.html)

<https://ieeexplore.ieee.org/document/8913471>

<https://inside-openflow.com/2016/07/21/ryu-api-dissecting-simple-switch/>

<https://www.semanticscholar.org/paper/Efficient-Routing-Approach-In-Software-Defined-Tejaswini-Divyavani/23699f063bb0d99fc386c79a5f2a30b96ed64497>



## CSB17039\_51\_67\_ROUTING IMPLEMENTATION BASED ON PROCESSING DELAY AND BANDWIDTH IN SOFTWARE DEFINED NETWORKING

### ORIGINALITY REPORT

<b>35%</b>	<b>24%</b>	<b>10%</b>	<b>24%</b>
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

### PRIMARY SOURCES

<b>1</b>	<b>Submitted to Royal Business College</b> Student Paper	<b>6%</b>
<b>2</b>	<b>www.forcepoint.com</b> Internet Source	<b>5%</b>
<b>3</b>	<b>Priyesh Kumar, Rajnarayan Dutta, Rakesh Dagdi, Kavitha Sooda, Archana Naik. "A Programmable and Managed Software Defined Network", International Journal of Computer Network and Information Security, 2017</b> Publication	<b>3%</b>
<b>4</b>	<b>github.com</b> Internet Source	<b>3%</b>
<b>5</b>	<b>academiccollegeprojects.com</b> Internet Source	<b>2%</b>
<b>6</b>	<b>Submitted to Istanbul Aydin University</b> Student Paper	<b>2%</b>

flowgrammable.org

7	Internet Source	2%
8	Submitted to Indian Institute of Science, Bangalore Student Paper	2%
9	upcommons.upc.edu Internet Source	1%
10	Submitted to University of South Australia Student Paper	1%
11	Submitted to Victorian Institute of Technology Student Paper	1%
12	www.irjet.net Internet Source	1%
13	intronetworks.cs.luc.edu Internet Source	1%
14	archive.org Internet Source	1%
15	Submitted to Institute of Technology, Tallaght Student Paper	1%
16	www.litcharts.com Internet Source	1%
17	en.wikipedia.org Internet Source	1%
18	uknowledge.uky.edu Internet Source	

---

		<1 %
19	osrg.github.io Internet Source	<1 %
20	www.freepatentsonline.com Internet Source	<1 %
21	Submitted to Symbiosis International University Student Paper	<1 %
22	Submitted to Swinburne University of Technology Student Paper	<1 %
23	docplayer.net Internet Source	<1 %
24	"Handbook of Computer Networks and Cyber Security", Springer Science and Business Media LLC, 2020 Publication	<1 %
25	Submitted to DeVry University Online Student Paper	<1 %
26	Submitted to North Down and Ards Institute of Further and Higher Education Student Paper	<1 %
27	epdf.pub Internet Source	<1 %