Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CSU33031 Computer Networks

Assignment #2: Flow Forwarding

Student name: Yingzheng Pan **Student number: 19336862**

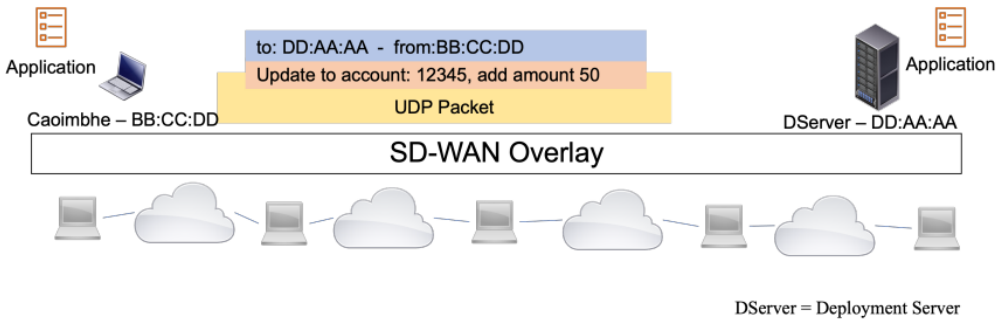
Contents

1 Introduction

The focus of this assignment is to learn about decisions to forward flows of packets and the information that is kept at network devices that make forwarding decisions. The design of forwarding mechanisms aims to reduce the processing required by network elements that forward traffic while providing scalability and flexibility.

2 Theory of Topic

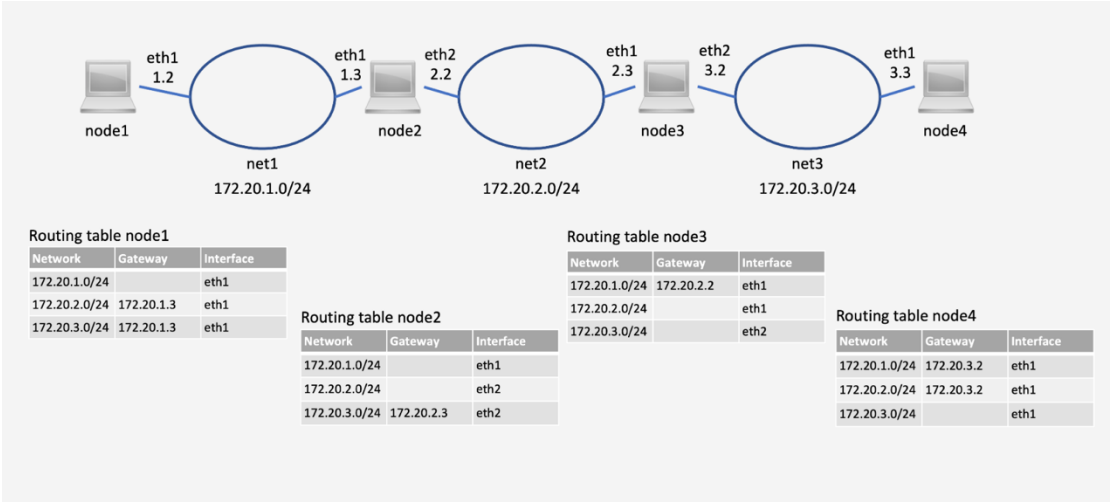
Due to the COVID pandemic, an increased number of employees in a company are working from home. If we are as development team of a network provider, we are being asked to look at the concept of Software-Defined Wide-Area Networks and extend the concept to ADSL routers at the homes of individual employees. Our task is to develop an overlay that links implementations of forwarding mechanisms in ADSL routers to network elements of the provider to forwarding services at a cloud provider. The forwarding information in the forwarding tables of the implementation at the ADSL routers, the network elements at the provider and the cloud provider will be controlled from a central controller at the network provider.

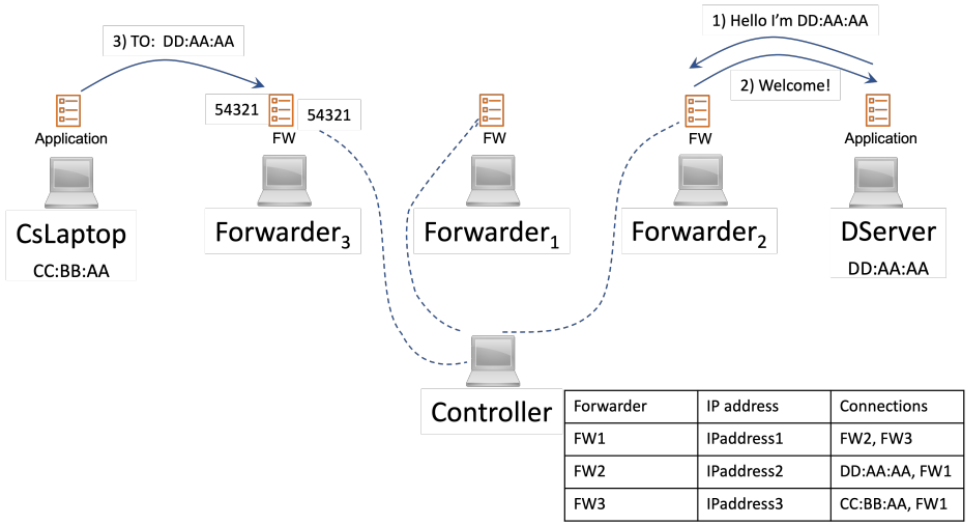


DServer = Deployment Server

As the figure shown above, the application will send UDP datagrams to the forwarding service on the local host. The forwarding service will refer to a table to determine where to forward the datagram from the protocol's header information. This table will provide it with the IP address of the network element as the next hop, and your implementation should forward it to another instance of the forwarding service at that IP address. Therefore, the basic functionality that a forwarding service has to provide is to accept incoming packets, inspect the header information, consult the forwarding table and forward the header and payload information to the destination. In the first place, if a destination is

not known, a forwarding service would drop an incoming packet; once a controller has been implemented, the forwarding service needs to contact the controller to enquire about a forwarding information to the unknown destination and if it receives forwarding information, integrate this into its forwarding table.

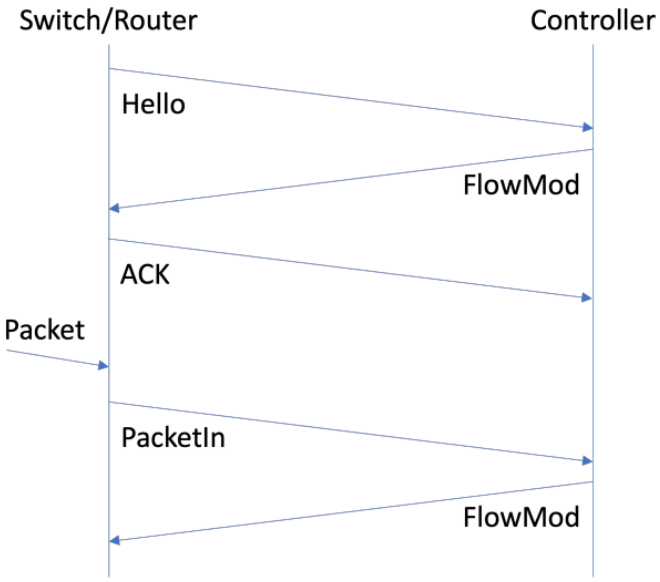




As the above figure shown, for the first above figure an example of 4 nodes connected to 3 networks using docker. The two nodes in the middle act as routers between the networks i.e. the two nodes at the end can only see the network that they are directly connected two and need to send all other traffic to the other node on the network which acts as a gateway. This setup is a simplistic example for the forwarding of IP packets, encapsulated in Ethernet frames. The implementation of the forwarding services for the overlay will use a UDP socket bound to port 54321 at every network element. An application at employees devices will be using your overlay by creating a header you design, attach the payload of the application to be transmitted, and send this data as payload to your forwarding service, for example bound to port 54321 on the default gateway, The forwarding service will examine the header and decide

where to forward the information to, based on the header information and a forwarding table. The forwarding table will indicate instances of the service on other network elements.

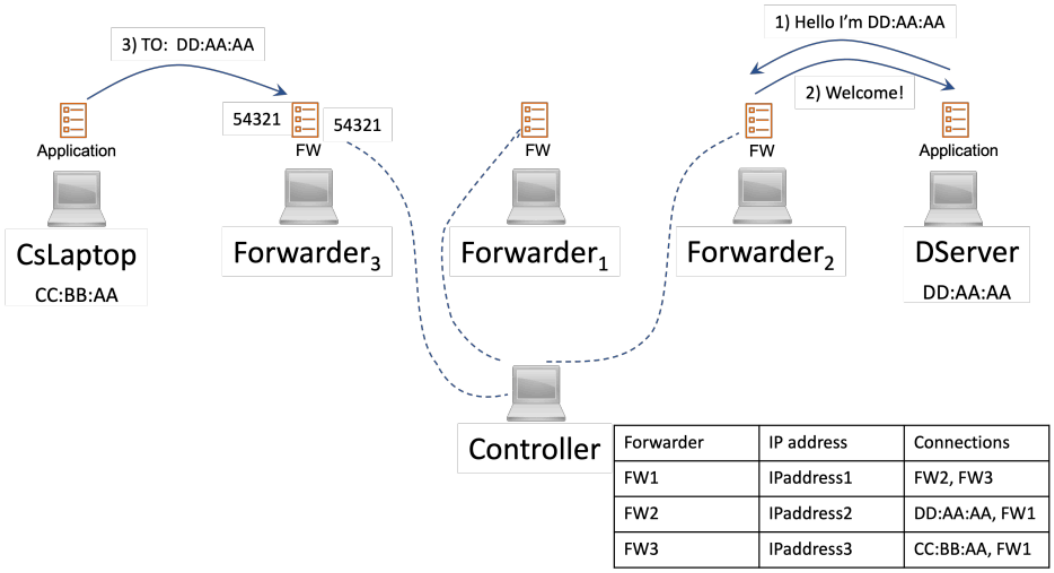
For now we can combine the information of first above figure and second above figure, we can know that There are some different network and different IP address between each different nodes, each node have their own small routing table, which only have the route beside the node, however the node still doesn't know the route to the unknown destination. That's the reason why we need controller. The forwarding service needs to contact the controller to enquire about a forwarding information to the unknown destination, because the small routing table in each forward they only have the route information of the beside node, they don't know how to go to the unknown destination, the forwarding services have the route information about beside node, but they don't have the route and information about the destination node therefore they need to send an request packet to ask controller for how to go to the destination node and the routing to the unknown destination, the controller will use the routing table in controller to send back the routing of how to go to destination and routing information to the forwarding services, therefore the forwarding service will send an request udp datagram packet to ask controller to provide them the routing so that they can know how to go to the unknown destination they want and if it receives forwarding information, they will integrate this into its forwarding table. In general, the network elements should be controlled by a central controller that initializes the flow tables of the network elements and would inform them about routes to destinations as the network changes. The implementation of forwarding services are able to handle to a number of concurrent flows originating from networks of end users and direct these flows to and from a number of services. The following flow figures is an example of visualizations of network traffic between network elements. It shows a sequence of messages exchanged between the elements.



The diagram shows a router contacting the controller with an initial 'Hello' message and receives a modification message to its flow table. It replies to this with an acknowledgement. When a router receives a packet with an unknown destination, it will contact the controller and receive an additional modification to its flow table.

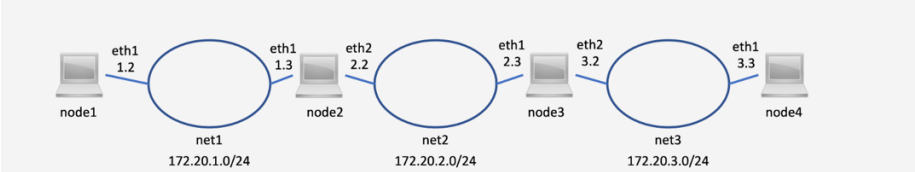
2.1 First Component: Client

Firstly, I used Docker, which can provide a mechanism to create a subnetwork on a local machine. Setup of the containers from creating the image, then the containers and networks and connecting the containers to the networks. I create some bridged network called net1, net2, net3 and then create a container image, named netimg, based on a Dockerfile. The first component container I created is Client.



Forwarder	IP address	Connections
FW1	IPaddress1	FW2, FW3
FW2	IPaddress2	DD:AA:AA, FW1
FW3	IPaddress3	CC:BB:AA, FW1

As the figure shown above, CsLaptop the first node is my Client node. Client node will send UDP datagrams to a forwarding service on the local host. Client will first ask user which node do you want to transmit and then when user input the node they want to transmit, and then it will ask user input an message for transmitting. The forwarding service will consult a table to determine from the header information of your protocol where to forward the datagram to. The table will provide it with the IP address of the network element that is the next hop.



Routing table node1

Network	Gateway	Interface
172.20.1.0/24		eth1
172.20.2.0/24	172.20.1.3	eth1
172.20.3.0/24	172.20.1.3	eth1

Routing table node2

Network	Gateway	Interface
172.20.1.0/24		eth1
172.20.2.0/24		eth2
172.20.3.0/24	172.20.2.3	eth2

Routing table node3

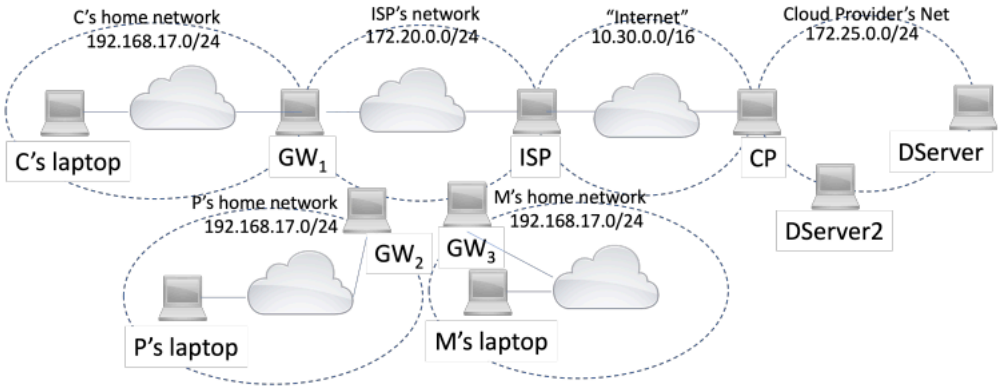
Network	Gateway	Interface
172.20.1.0/24	172.20.2.2	eth1
172.20.2.0/24		eth1
172.20.3.0/24		eth2

Routing table node4

Network	Gateway	Interface
172.20.1.0/24	172.20.3.2	eth1
172.20.2.0/24	172.20.3.2	eth1
172.20.3.0/24		eth1

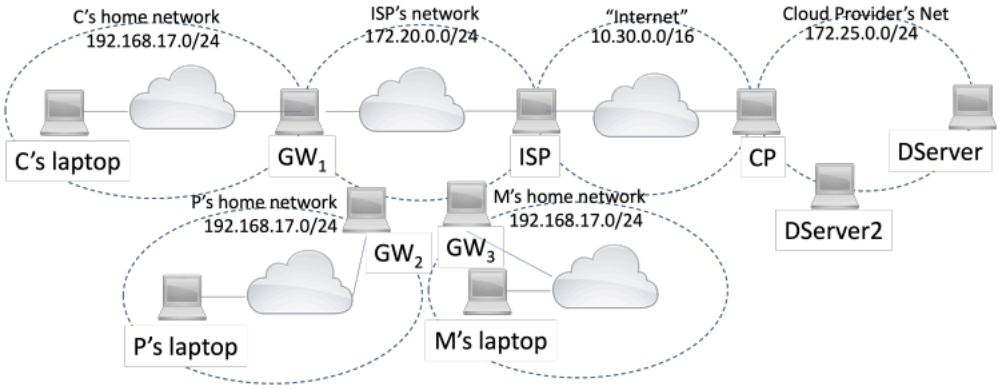
However as the second above figure shown, each node only have a small routing table and only have the beside node route and beside node routing information. In my client part, my client part will ask user to enter a message they want to transmit, and then after when the Client ask for the user input message, it will use udp datagram to set up a datagram packet and then put the user input message into the udp datagram packet and then send it to the forwarding services, However forward only have the routing table which only have the beside node routing, they doesn't know how to transmit the packet, and then they will ask controller to tell them the routing about how to go to the unknown destination by using the routing table in controller part. And then controller send back the routing to destination, what's more, the client message which is user input message will successfully send the udp packet to the last node Server node(destination).

2.2 Second Component: Forwards



As the figure shown above, C's laptop is my Client, P's laptop and M's laptop is my serverA and serverB, Dserver is my server node. As we can see in the figure, I have implemented nine node and one controller in my project. And I have six network connect between each node, my network systems are both as the same as the figure shown. I have five forwards in my project, which are forwardA(GW1),forwardB(ISP),forwardC(CP),forwardD(GW2),forwardE(GW3). At the beginning, forwards will receive the packet from Client, In Client user will enter message they want to transmit and also choose the destination node they want to transmit to. When forwards receive the packet from Client, Client will also show an acknowledgement message to show forwards receive the packet from Client. Each forwards have their own small routing table in it, which only have the beside node route and route information but it doesn't have the destination route so that those forwards will ask controller to tell them how to go to the destination node. Forwards will send an request packet which is use to ask controller to provide them a routing which can go to the destination node. When forwards receive the routing from controller, they will send an acknowledgement response message to controller so that to show they receive the packet successfully. When the forwarding services receive the routing to destination they will integrate this into its forwarding table, and then follow the routing that controller provide them to send the packet to the destination node.

2.3 Third Component: Controller



As the above figure shown, that's the basic structure of my project. I have a controller which control all the forwarding services. When Client send the user input message to forwards, however forward have the small routing table which only have the beside node routing so that they need to send an request packet to ask controller the routing to the destination. In Controller, I implemented the Dijkstra's shortest path algorithm to draw and create each path between each node. My controller I have a central big routing table which have all the routing between start node and all destination nodes. When controller receive the request packet from forwards it will send the routing to destination back to forwards, and then it will show the acknowledgement message to show which forwards receive the packet from controller which contains the route to destination. When the forwarding service receive the routing to destination node from controller, the forwards will send the user input packet follow the routing that controller provide them to the destination node(Server node).

2.4 Fourth Component: Server

Finally, my last component is Server node. My server node is my destination node, which is the node that receive all the user input message from client. Client will send the udp packet to forward, and then forward will send an request to ask controller for the routing for how to go to the destination node by using the routing table in the controller. And then controller will send the routing to forwards, and then the forwarding services will send the client user input message to the Server(destination node), my server node receive the user input message packet and then will send back an acknowledgement response packet to forwarding services, when we use Wireshark to capture the last node(Server node) traffic, which will show user input message packet that Server received and also user message packet content in pcap file.

2.5Communication and Packet Description(Encoding)

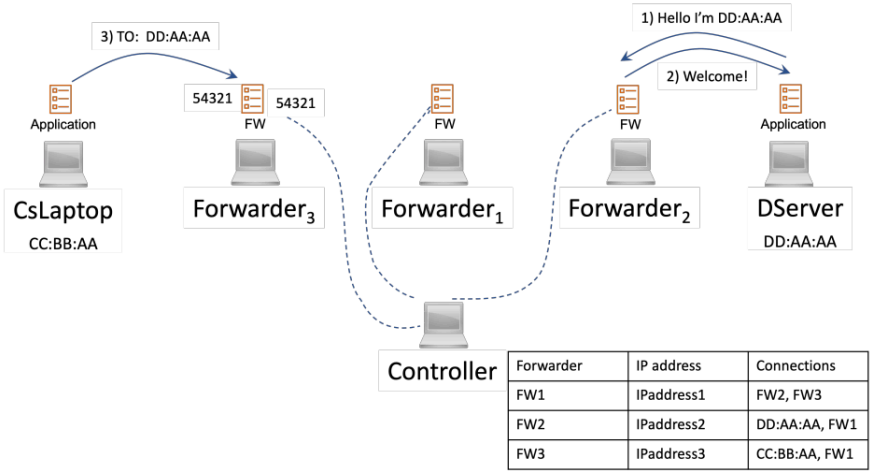
Understanding of UDP: For UDP (User Datagram Protocol), which provides bidirectional message transport between the server and one or more clients. UDP is not connection based, and each packet transmission is a separate event. Provides fast and lightweight data transfer for local packet broadcast and remote multicast. In UDP, the client does not form a connection with the server like in TCP and instead just sends a datagram. Similarly, the server need not accept a connection and just waits for datagrams to arrive. The datagram on arrival contains the sender's address, which the server uses to send the data to the correct client. It is mainly used to establish low latency and fault tolerant connections between applications on the internet. UDP speeds up transmission by enabling data transfer before the recipient offers the protocol.

Understand of acknowledgement message: In data networking, telecommunications, and computer buses, an acknowledgment (ACK) is a signal that is passed between communicating processes, computers, or devices to signify acknowledgment, or receipt of message, as part of a communications protocol. The negative-acknowledgement is a signal that is sent to reject a previously received message or to indicate some kind of error. Acknowledgments and negative acknowledgments inform a sender of the receiver's state so that it can adjust its own state accordingly.

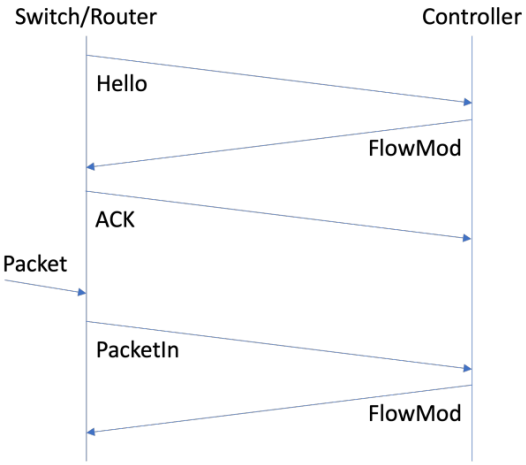
Understand of udp datagram packet: Datagrams are data packets which contain adequate header information so that they can be individually routed by all intermediate network switching devices to the destination. These networks are called datagram networks since communication occurs via datagrams. They exist in packet switching networks. A datagram is a basic transfer unit associated with a packet-switched network. Datagrams provide a connectionless communication service across a packet-switched network.

Datagram packet switching is a packet switching method that treats each packet, or datagram, as a separate entity. Each packet is routed via the network on its own. It is a service that does not require a connection. Because there is no specific channel for a connection session, there is no need to reserve resources. In this project I am using udp datagram packet to transmit each node and controller information and data by using udp datagram packet. In this assignment, I use UDP Socket and UDP Datagram to solve the problem and send the datagram packets. My protocol and send packets process are all base on UDP Socket and UDP Datagram. Datagram packets are used to implement connectionless packet delivery services. Each message is routed from one machine to another, based solely on the information contained in the packet. Multiple packets sent from one machine to another may be routed differently and may arrive in any order. A datagram socket is a sending or receiving point for a packet delivery service. Each packet sent or received on a datagram socket is addressed and routed separately. Multiple packets sent from one machine to another may be routed differently and may arrive in any order. UDP

socket routines use the User Datagram Protocol (UDP) for simple IP communication. The User Datagram Protocol runs on top of the Internet Protocol (IP) and is developed for applications that do not require reliability, acknowledgment, or flow control features at the transport layer.



Forwarder	IP address	Connections
FW1	IPaddress1	FW2, FW3
FW2	IPaddress2	DD:AA:AA, FW1
FW3	IPaddress3	CC:BB:AA, FW1



For the communication of my project, which is shown as the first above figure. Client node(first node) will send UDP datagrams to a forwarding service on the local host. The forwarding service will consult a table to determine from the header information of your protocol where to forward the datagram to. The table will provide it with the IP address of the network element that is the next hop. As we can see each node only have a small routing table and only have the beside node route and beside node routing information. In my client part, my client part will ask user to enter a message they want to transmit, and then after when the Client ask for the user input message, it will use udp datagram to set up a datagram packet and then put the user input message into the udp datagram packet and then send it to the forwarding services, when the forwarding services receive the packet they will show an acknowledgement response message to show they are successfully receive the packet from client. The same principle as the second above figure, the forwarding services and

controller they are both receive the acknowledgement response message to make sure and show they are successfully receive the packet. When the forwards receive the packet from client, However forward only have the routing table which only have the beside node routing, they doesn't know how to transmit the packet, and then they will ask controller to tell them the routing about how to go to the unknown destination by using the routing table in controller part. And then controller send back the routing to destination, when forwarding services receive the routing they will know the routing to destination node, and then they will send the user input message to the destination node(Server node). The second above diagram shows a router contacting the controller with an initial 'Hello' message and receives a modification message to its flow table. It replies to this with an acknowledgement. When a router receives a packet with an unknown destination, it will contact the controller and receive an additional modification to its flow table. For all packets used in the transmission process between clients, forwarding services, acknowledgement message, controller, server, I all use udp datagram packet for transmitting data.

3 Implementation

3.1 Component 1: Client

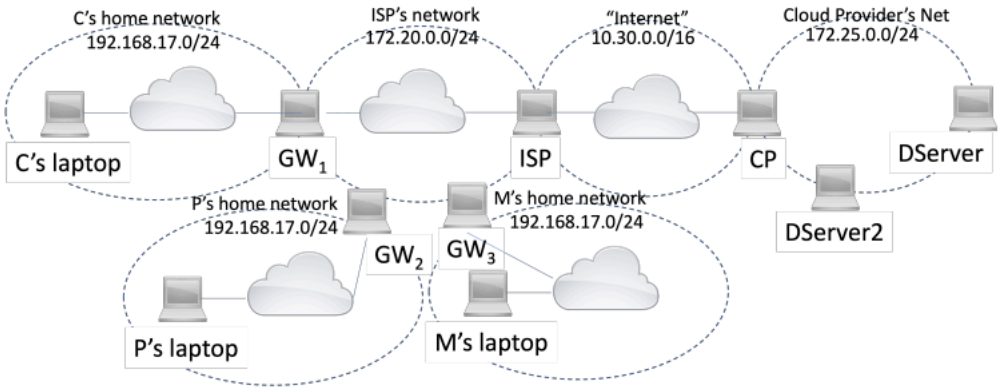
```
public class Client extends Node {
    static int DEFAULT_SRC_PORT = 54321;
    static final int DEFAULT_DST_PORT = 54321;
    InetSocketAddress dstAddress;
    InetAddress IPv4;
    static int number=1;
```

As the figure shown above, at the beginning of my Client part, The implementation of the forwarding services for the overlay will use a UDP socket bound to port 54321 at every network element.

```
public synchronized void start() throws Exception {
    String fname;
    FileInfoContent fcontent;

    int size;
    DatagramPacket packet= null;

    System.out.println("Which node do you want to transmit: ");
    fname= System.console().readLine();
    System.out.println("Enter message: ");
    fname= fname + " " + System.console().readLine();
    size= 0;
    fcontent= new FileInfoContent(fname, size);
    packet= fcontent.toDatagramPacket();
    packet.setSocketAddress(dstAddress);
    socket.send(packet);
    System.out.println("Packet sent");
    this.wait();
}
```

As the figure above shown, that's the main part of my Client, In my Client, it will ask user to input which node do you want to transmit to determine the destination node that user want to transmit to. And then it will ask user to enter the message they want to send to. I used System.console().readLine() to get the user enter input content. And then put the user input content into the datagram packet, when we finish set up packet and put the user input into the datagram packet and then we use socket.send() to send the datagram packet to next node address. All the sending processes are shown in the above figure, the code in the figure is a demonstrating sender packet method.

3.2 Component 2: Forwards



As the above figure shown, I have five forwards in my project, which are forwardA(GW1),forwardB(ISP),forwardC(CP),forwardD(GW2),forwardE(GW3).

Each forwards have their own small routing table, which only have the beside node routing.

```
HashMap<String, String> table = new HashMap<String, String>();
/*
 *
 */
forwardA(int port) {
    try {
        socket= new DatagramSocket(port);
        table.put("node1","node1");
        table.put("node3","node3");
        listener.go();
    }
```

```
HashMap<String, String> table = new HashMap<String, String>();
/*
 *
 */
forwardB(int port) {
    try {
        socket= new DatagramSocket(port);
        table.put("node2","node2");
        table.put("node4","node4");
        table.put("node6","node6");
        table.put("node8","node8");
        listener.go();
    }
```

```
HashMap<String, String> table = new HashMap<String, String>();
/*
 *
 */
forwardC(int port) {
    try {
        socket= new DatagramSocket(port);
        table.put("node3","node3");
        table.put("node5","node5");
        listener.go();
    }
```

```
HashMap<String, String> table = new HashMap<String, String>();
/*
 *
 */
forwardD(int port) {
    try {
        socket= new DatagramSocket(port);
        table.put("node2","node2");
        table.put("node3","node3");
        table.put("node7","node7");
        table.put("node8","node8");
        listener.go();
    }
```

```
HashMap<String, String> table = new HashMap<String, String>();
/*
 *
 */
forwardE(int port) {
    try {
        socket= new DatagramSocket(port);
        table.put("node2","node2");
        table.put("node3","node3");
        table.put("node6","node6");
        table.put("node9","node9");
        listener.go();
    }
```

As those figures shown in the all above figure, those are the small routing table in my each single five forwards, the routing table in forwards only have the routing information for beside nodes.

```
public class forwardA extends Node {
    public void onReceipt(DatagramPacket packet) {
            System.out.println("Received packet");

            PacketContent content= PacketContent.fromDatagramPacket(packet);

            if (content.getType()==PacketContent.ACKPACKET){
                System.out.println(content.toString());
            }
            if (content.getType()==PacketContent.FILEINFO) {
                AddressA = packet.getSocketAddress();
                DatagramPacket response;
                response= new AckPacketContent("OK - ForwardA Received this").toDatagramPacket();
                response.setSocketAddress(packet.getSocketAddress());
                socket.send(response);
                if(((FileInfoContent)content).getFileSize()==2){
                    String fileName2 = ((FileInfoContent)content).getFileName();
                    System.out.println(fileName2);
                    dstAddress3 = new InetSocketAddress(fileName2, 54321);
                    dataA.setSocketAddress(dstAddress3);
                    socket.send(dataA);
                    table.put(str2,fileName2);
                    }
                if(((FileInfoContent)content).getFileSize()==0){
                    String str = ((FileInfoContent)content).getFileName();
                    String[] splited = str.split(" ");
                        String value = splited[0];
                        str2 = value;
                        String val;
                            if (table.keySet().contains(value)) {
                                val = table.get(value);
                                dstAddress2 = new InetSocketAddress(val, 54321);
                                packet.setSocketAddress(dstAddress2);
                                socket.send(packet);
                                System.out.println("Packet sent");
                            }
                            else {
                                FileInfoContent  content2;
                                content2 = new FileInfoContent(value,0);
                                dataA = packet;
                                dstAddress2 = new InetSocketAddress("controll", 54321);
                                packet = content2.toDatagramPacket();
                                packet.setSocketAddress(dstAddress2);
                                socket.send(packet);
                                System.out.println("Packet sent");
                            }
                    }
            }
```

As the above figure shown, that's my main part of my forwards. We take forwardA code as an example, because most of the code are similar between every forwards, they are both the same principle. First if statement have

condition equal ACKPACKET, which is use to receive the acknowledge message and then print the acknowledgement message in the forwards part to show they are successfully receive the packet. And then next if is to use to file.getName to get the node information so that it get the next following node name and then set that as a datagram packet address and then send the packet to that address node. The first part of if statement are used to when they know the routing to destination they will send the packet to that destination node. By the way, when they receive one time that controller send the destination node routing to them, forwards will not ask the same destination route again, if it receives forwarding information, integrate this into its forwarding table. The else statement part is use to when forward receive the packet from Client thy don't know how to go to the destination, therefore they will send an request packet to controller and then ask the destination route so that they know how to go to the unknown destination.

3.3 Component 3: Controller

```
controller(int port) {
    try {
        socket= new DatagramSocket(port);
        edge = new DirectedEdge(1,2,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(2,1,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(2,3,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(3,2,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(3,4,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(4,3,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(4,5,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(5,4,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(6,2,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(2,6,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(6,3,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(3,6,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(6,7,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(7,6,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(8,2,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(2,8,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(8,3,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(3,8,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(8,9,1);
        graph.addEdge(edge);
        edge = new DirectedEdge(9,8,1);
        graph.addEdge(edge);
        listener.go();
    }
```

As the above shown that's the routing table in my controller. In my controller I also use Dijkstra's shortest path algorithm and DirctedEdge and EdgeDiGraph those java to create the path and then make those route to a graph, and then use shortest path algorithms to find the exist path and shortest path between node to the destination node.

```
public void onReceipt(DatagramPacket packet) {
    try {
        System.out.println("Received packet");

        PacketContent content= PacketContent.fromDatagramPacket(packet);

        if (content.getType()==PacketContent.ACKPACKET){
            System.out.println(content.toString());
        }
        if (content.getType()==PacketContent.FILEINFO) {

            String name = ((FileInfoContent)content).getFileName();
            int index=0;
            for (int i=0; i < string.length; i++){
                 if(string[i].equals(name)){
                      index = i;
                     }
            }
            index = index+1;
            String hostName = ((InetSocketAddress) packet.getSocketAddress()).getHostName();
            String[] parts = hostName.split("\\.");
            String[] afterDot = parts[0].split("");
            DijkstraSP sp = new DijkstraSP(graph, Integer.parseInt(afterDot[4]));
            List<DirectedEdge> DirectedEdge = new Stack<DirectedEdge>();
            int x = index;
            DirectedEdge = sp.pathTo(x);
            int integer = DirectedEdge.get(DirectedEdge.size()-1).to();
            FileInfoContent content3;
            content3 = new FileInfoContent(string[integer-1],2);
            AddressA = packet.getSocketAddress();
            packet = content3.toDatagramPacket();
            packet.setSocketAddress(AddressA);
            socket.send(packet);

            DatagramPacket response;
            response= new AckPacketContent("OK - controller Received this").toDatagramPacket();
            response.setSocketAddress(packet.getSocketAddress());
            socket.send(response);

        }
        if (content.getType()==PacketContent.FILEINFOA) {
            packet.setSocketAddress(AddressA);
            socket.send(packet);
            System.out.println("Packet sent");
        }

    }
```

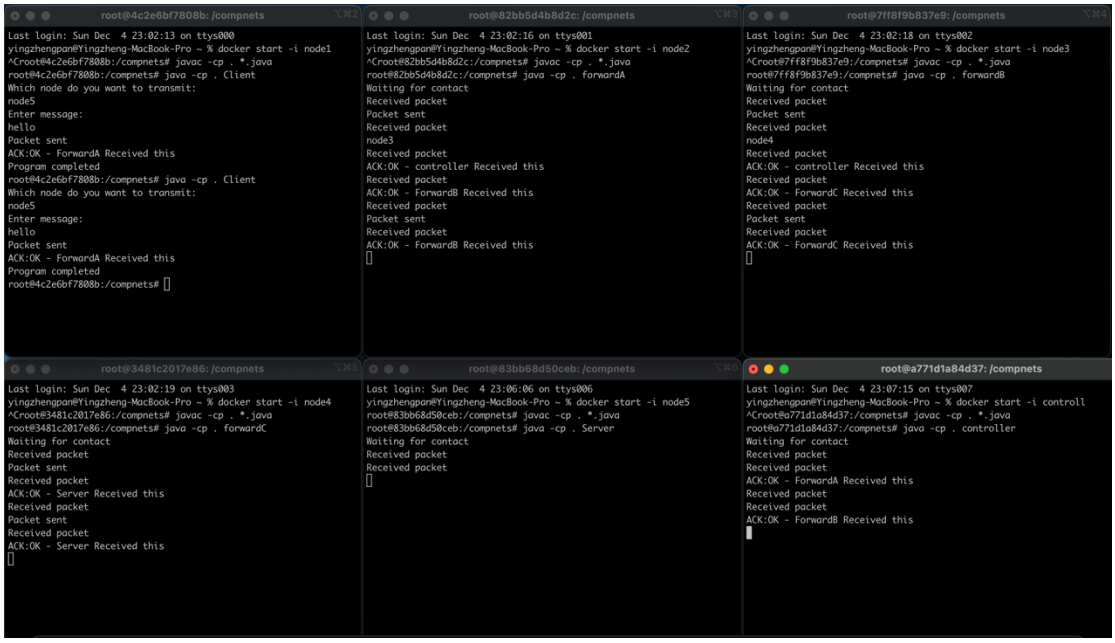
As the above figure shown, that's the main part of my controller, I will get the forwards send packet node and then use the routing table to check there are any route that forwards have the path to another, and then controller will send the routing back to the forwards which send the request to it. It will use routing table find the path between each forwards and destination, each forwards will send an request to controller, also controller will also send back the ack message to forwards.

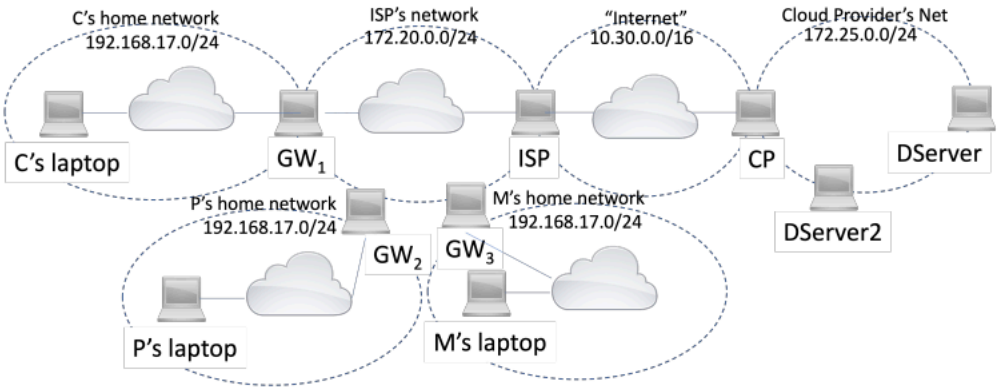
3.4 Component 4: Server

```
public void onReceipt(DatagramPacket packet) {
    try {
        System.out.println("Received packet");

        PacketContent content= PacketContent.fromDatagramPacket(packet);

        if (content.getType()==PacketContent.ACKPACKET){
            System.out.println(content.toString());
        }
        if (content.getType()==PacketContent.FILEINFO) {
            AddressA = packet.getSocketAddress();
            DatagramPacket response;
            response= new AckPacketContent("OK - Server Received
                this").toDatagramPacket();
            response.setSocketAddress(packet.getSocketAddress());
            socket.send(response);
        }
        if (content.getType()==PacketContent.FILEINFOA) {
            packet.setSocketAddress(AddressA);
            socket.send(packet);
            System.out.println("Packet sent");
        }

    }
    catch(Exception e) {e.printStackTrace();}
}
```

Finally, my last component is Server node. My server node is my destination node, which is the node that receive all the user input message from client. Client will send the udp packet to forward, and then forward will send an request to ask controller for the routing for how to go to the destination node by using the routing table in the controller. And then controller will send the routing to forwards, and then the forwarding services will send the client user input message to the Server(destination node), my server node receive the user input message packet and then will send back an acknowledgement response packet to forwarding services, when we use Wireshark to capture the last node(Server node) traffic, which will show user input message packet that Server received and also user message packet content in pcap file. The basic part of my server is receive packet from Client.

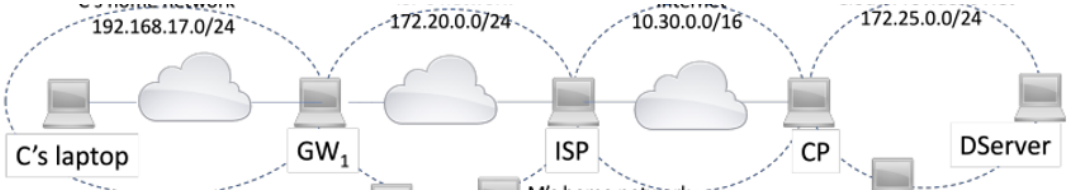
3.5 User Interaction



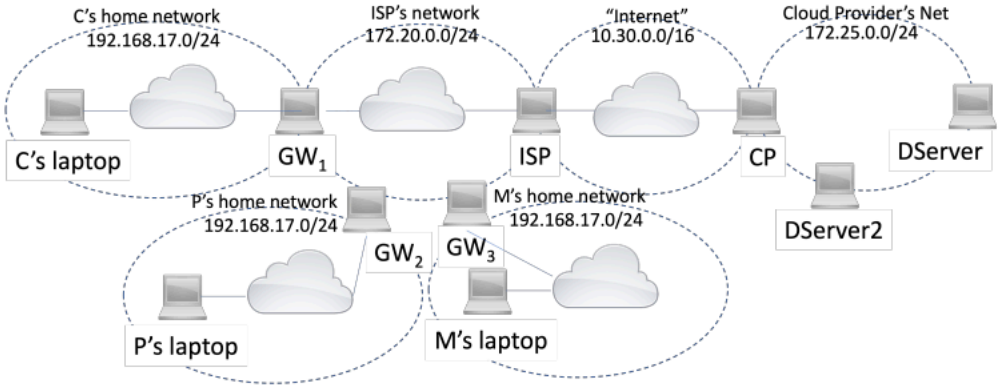


As the figure shown above, In Client part user will input which node do you want to transmit and enter the message you want to transmit to. When user input destination node and message, Client will send the packet to forwardA, as we know in forwards, they are only know the beside node route, but cannot know the destination node that user enter, because in forwards they didn't have the destination node route in their routing table, therefore they need to ask controller for routing to destination, In controller routing table, it have all route or path between each node, therefore controller will tell forwardA, it can go to the forwardB, because there is an path between them and also can go to the destination, and then controller send back the packet include the next node route to forwardA, and also show an acknowledgement message to show controller receive the packet from forwards, and then forwardB receive the packet form forwardA, and then forwardB didn't have the destination node route in its routing table, therefore it also send an request to controller, and then controller send back the packet contain forwardC route to forwardB, and then finally forwardC receive the packet from forwardB, and now forwardC in its

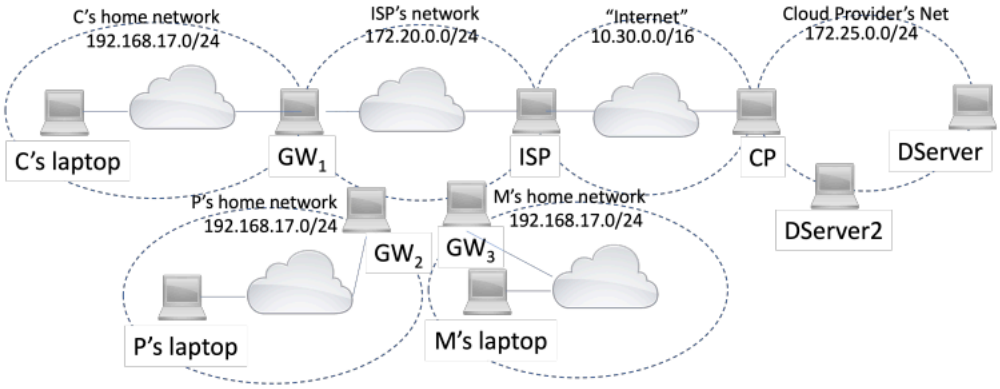
routing table it has the last beside node Server node route, therefore it just send it to the destination node that user ask for. So as you can see the first above figure, controller only show forwardA and forwardB acknowledgement message without forwardC, that's the reason why. And then when user input again that destination and message, we can see in the first figure, forwards didn't ask controller for the second time of transmitting packet, because if it receives forwarding information, they will integrate this into its forwarding table. So when user want to send the message again to the same destination node second time again, the forwards will have the first time routing records in its forward routing table therefore they will just send packet that by using the routing to destination node. That's is one route as an example for the process of send packet to destination node and how does controller works.



As the figure show, I take this route as an example, but I have many routes as the above second figure shown, I have the same structure as the following figure shown. Those route and how the other forwards works are as the same as the route example shown, the other route and the other destination server node and controller are both have the same principle. I have many routes as the figure shown here.



4 Summary



For my protocol program, as the figure shows that's the main logic and approach for the project. Firstly, Client node will send UDP datagrams to a forwarding service on the local host. The forwarding service will consult a table to determine from the header information of your protocol where to forward the datagram to. The table will provide it with the IP address of the network element that is the next hop. Client will ask user input message and then it will put it into the udp packet and then send the packet to the forwarding services, the forwarding service accept incoming packets, inspect the header information, consult the forwarding table and forward the header and payload information to the destination. In each forwards they have their own small routing table but only have the route information nodes beside them. If they don't know the destination, the forwarding service needs to contact the controller to enquire about a forwarding information to the unknown destination and if it receives forwarding information, integrate this into its forwarding table. The forwarding service need to send an request to controller for asking how to go to the destination node by using routing table inside controller, my controller use EdgeWeightedDigraph and DirectedEdge to make each route and create each path between each node and then make it as a routing graph. And then my controller will use Dijkstra's shortest path algorithms to find the exist path between each nodes and shortest path to the destination node and then send the routing to the forwarding services, when the forwarding services receive the routing the controller will also show a acknowledgement message that shows the forwarding services receive the packet successfully. As the figure above shown I have six network and nine nodes in my project. The forwarding services use the routing that controller give them and integrate this into its forwarding table and then it will send the packet to my last node(destination) Server, when the server receive packet it will also send an acknowledgement message to the forwarding services to show that Server receive packet successfully, and when we use Wireshark to capture the last node traffic, it will show packet content and user input message in pcap file that Server received.

5 Reflection

In conclusion, I have learned a lot from this assignment, I know more knowledge about UDP in this module. I expanded my understanding of network and knowledge of Flow Control and OpenFlow. From this assignment, I made my own OpenFlow and Flow Forwarding program, which made me more familiar with and better understand OpenFlow and Flow Control. I am looking forward to and will work harder to complete the future programs. For completing this assignment, which help me learn more knowledge and broaden my horizon about computing network. This assignment help me learn more knowledge about OpenFlow and flow forwarding, which make me better and better for future work. I am really grateful for the computing network course.