SOFTWARE DEFINED NETWORKING-Application Development

Abhijeet Sahu 424009758

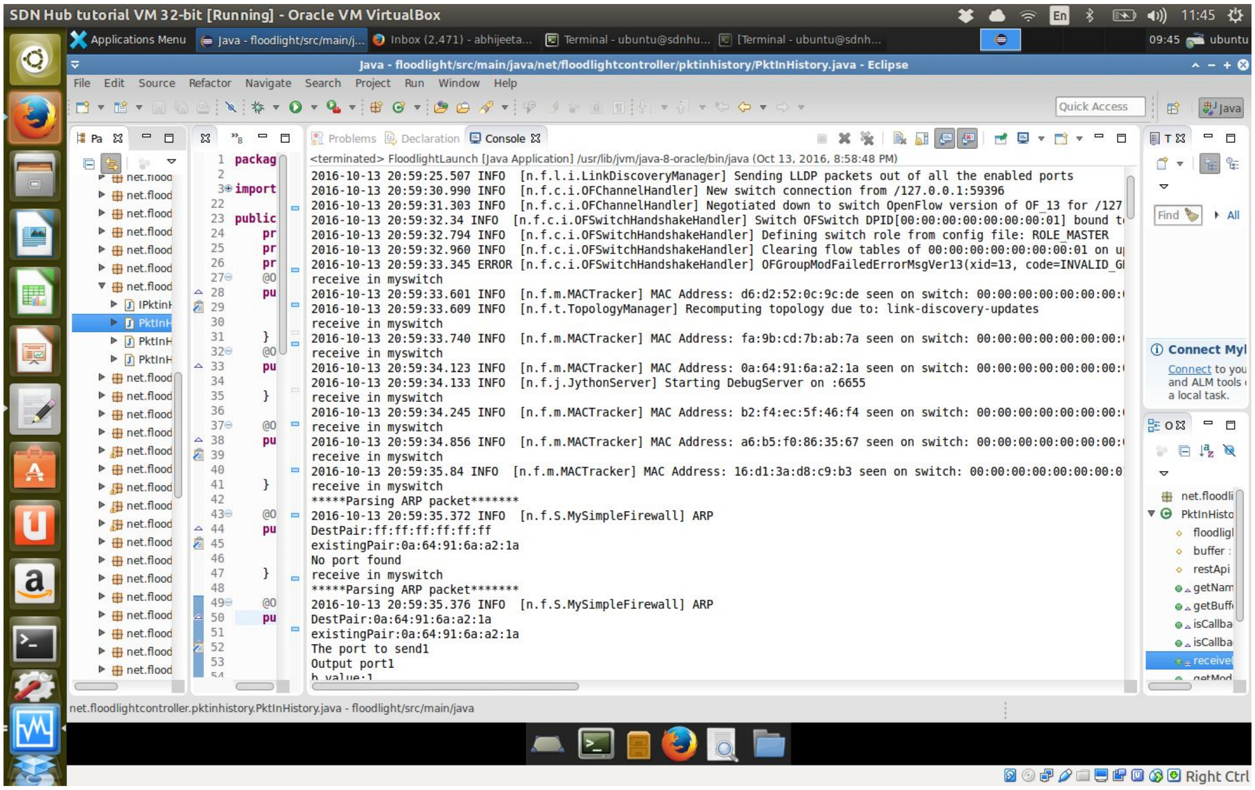
Assignment 3

1. Implementation of a MACTracker

In SDN, the main controller is made centralized and the task of taking decisions on forwarding, routing or blocking is taken away from the switches,routers or firewalls which are in data plane. The decisions are made by controller which is in control plane. The controller and switch need to communicate which each other. This is achieved by OpenFlow protocol. The packets the switch sends to controller are Packet_In messages. They contain the requests by hosts. The packets the controller sends to switches are Packet_Out and FlowMods. FlowMods instruct the switch to send the packets out through a particular port so that the packets can reach their destination. In program we will be coding the controller using FloodLight and the protocol that we use to communicate the controller with switch is OpenFLow.

Our program basically tracks the MAC address of all devices connected to our switch which is in mininet program. Whenever a host pings it first sends an ARP request. The switch checks in its forwarding table, if it finds the MAC address of destination it forwards it or else it sends a Packet_IN message to controller in FloodLight. The MACTrackers receive function takes this Packet_In function and parses the information found in the packet. The controller class in the net.floodlightcontroller.core package is the main class that calls other modules. Logger is used to printout the MAC addresses that we got from the packets. The main thing to learn from this application is how it parses the packet and we will be developing a learning switch in the next task based on the data we recieved from this packet.

To implement the MACTRACKER application in the Floodlight we would require the Forwarding module of the floodlight.

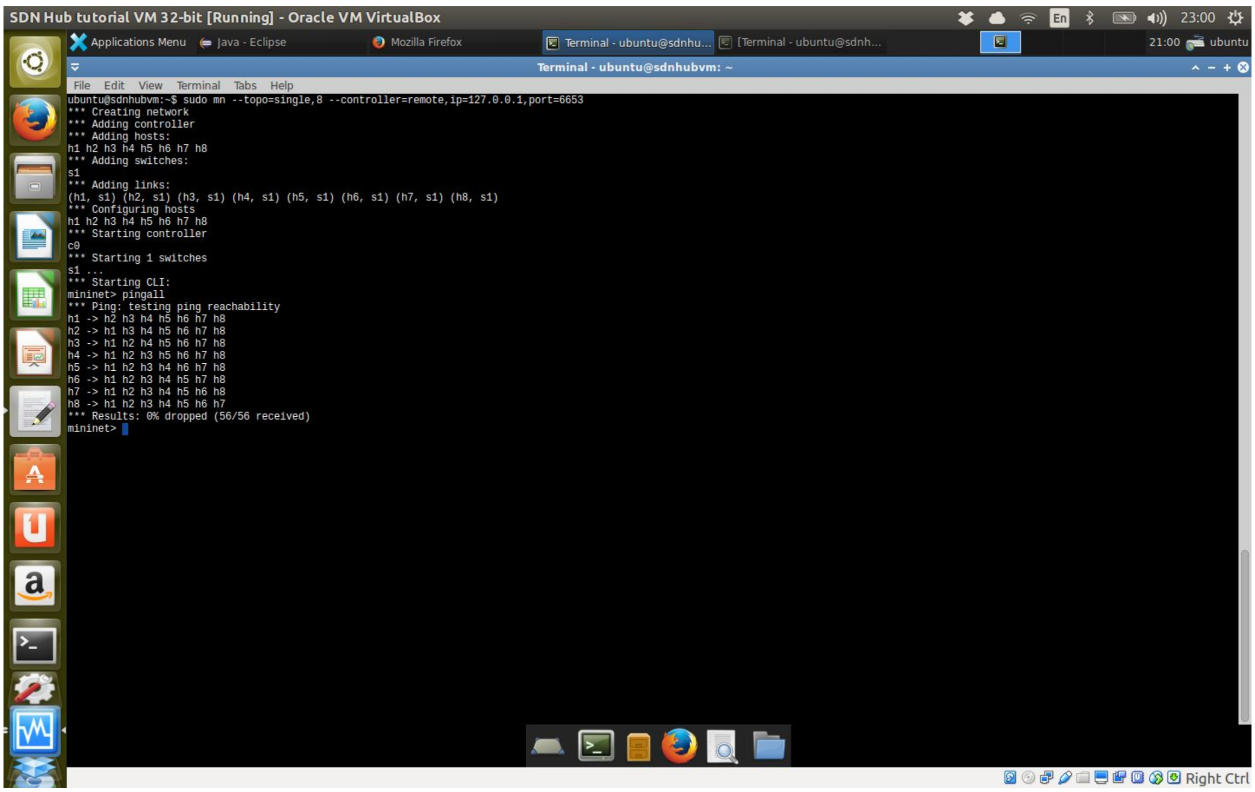


2. Implementation of Learning Switch

In this task I created a switch that is able to do forwarding and flooding by itself without using the inbuilt forwarding package . In the algorithm we form a hash map with MAC address as the key and its port number(in_port) as the corresponding value. The table is built in the following way:

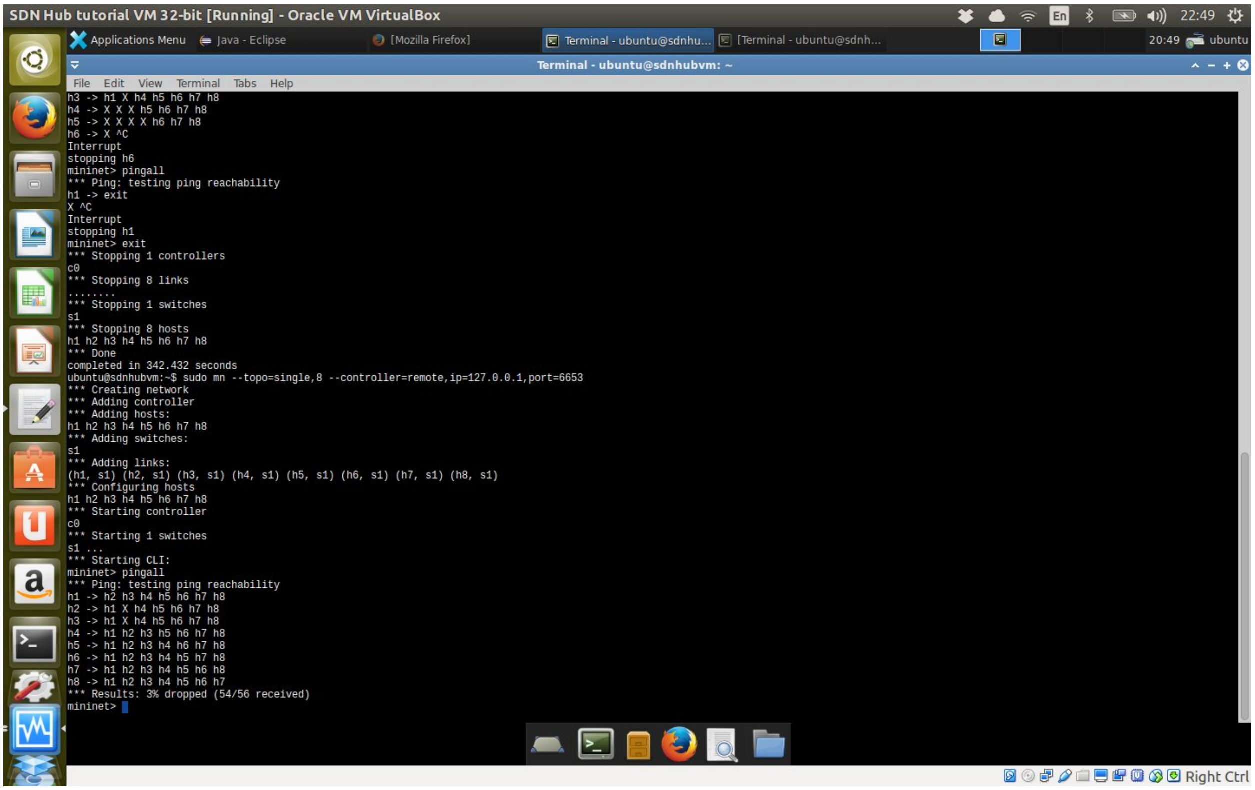
When the controller first receives a packet it stores its MAC address and port number in hash map. Now as it is the first entry in hash map it doesn't know where to send the packet. It sends a FlowMod FLOOD command which instructs the switch to flood(ARP Request). Switch performs flooding and finds the destination MAC address and port number(ARP reply). In this way hash map is filled and the controller sends a Packet_out command(or a FlowMod command) to switch about what it learns from the ARP reply. In my case as I am considering only a single switch when the key for hash table is made i consider MAC address instead of MAC address and IP address combination. Because of this when we will be able to communicate with hosts that are only on same switch not on difference switches. In our key if we include Switchid and MAC address combination then we will be able to communicate with hosts on other switches too.

The snapshot of 8 hosts communicating with each other:



3 Implementation of a Firewall(A little modification to the previous learning switch)

In this task we created a topology in which there is a switch(S) and a few host. We have created a firewall in which any number of host can communicate among each other except h2-h3 pair. For this we have to modify the flow rule so that when packets from h2 or h3 try to communicate with other host they should not reach other. Whenever we find a destination and source host with port number 2 and 3 we give a flowmod DELETE command to the open flow switch in mininet. As we are using only MAC address and port number to form hash map we can use either of them to block the communication. In our program we are using port number to block communication.



4. SDN based Application

Implementation of a simple Proxy application to handle ARP reply request. This application tries to transfer the ARP reply request handling task from data plane to the control plane in the controller.

ARP messages are not forwarded by the data plane, but handled by the control plane. The algorithm:

1. **First the command is recieved in the recieve() Function. This function calls the processPacketInMessage() function. This function parses the information from the packet_in object. If it is an ARP based packet, it opens the payload.**

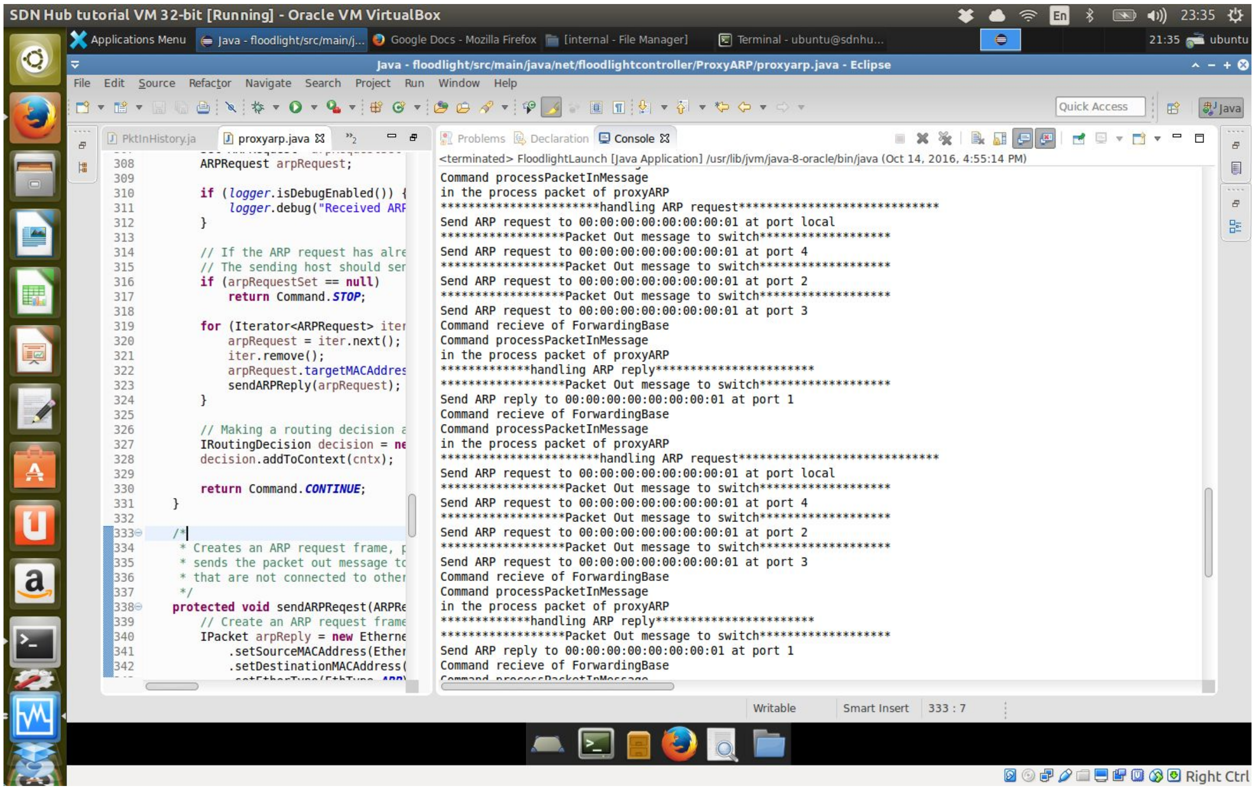
2. Then if a routing decision is already made for the packet, then it follows the decision. If a routing decision is made for the packet then it checks if it is a DROP,FORWARD,FORWARD_FLOOD or multicast DECISION. It only handles the packet if its routing decision has a FORWARD,FORWARD_flood OR MULTICAST decision in the OP_code of the ARP payload. If it is a ARP request it calls handleARPrequest function and if it is ARP reply it calls handleARPreply function. Based on the reply it changes the routing decision(with the knowledge of the destination MAC address and its port number).

3. In the handleARPrequest(), it calls the devicequery() function created in net.floodlightcontroller.devicemanager.IDeviceService service class to get the MAC address associated with the given IP address.If it finds the mac address for the destination device it calls the sendarpreply() function where a new ARP packet is built and send to the specific host by calling the sendpo() function . And if it doesn't find the mac address then it first calls the putarprequest() where it stores all the request in a arprequests Map holding the destination IP address and the ARP request. After holding the map it calls the sendarprequest() function where it sends the ARP request to all the switch ports that are not connected to openflow switches. For this we use the getAllSwitchDpids() function of the switchserviceprovider class.

4. In the handleARPreply() it looks into the ARP request timeout. If it is already timed out then the host must send a new ARP request.Here it calls the sendarpreply() function which sends the arp reply to the host.

The ARPProxy uses the topology information to extract location of the traffic's destination. Then it offers the MAC address directly to a requesting host. ARP

messages are sort of tunneled to the OpenFlow network.



The above image shows how the ARP request are sent to all the devices connected to the switch(h2,h3 and h4) to acquire the MACaddress associated with the IP address and forward them to the host h1 by sending a ARP reply. So here the ARP reply request payload is dealt completely by the application in the control plane rather than the data plane. This makes the switch more lighter and would demand higher computational capacity of the server acting as the controller.

Due to time constraint i had to built an already existing project which was built in the previous version of floodlight. I have tried to understand the code and built the whole code that can run in the Floodlight 1.3 version. This proxy application can still be improvised so that the controller can have a better way of learning rather than learning all the devices, it can use some filter to probe specific host based on their usage. And this application will reduce the load of switch which has a dhcp server connected to the network.