Question 1. In a conventional network, how can a switch know where to forward an

incoming packet? Please explain briefly the mechanism.

**Answer:-** In a conventional network, the controller of the maintains ARP lookup table, which is filled with MAC and IP address entries, as assigned by the DHCP server. The DHCP server assigns a static or dynamic IP address depending on the perspective it is being used, for example static IPs from Network Interface Controllers or by using the link local ARP, in the event, the static IP is not present. The ARP table maintained by the switches is updated at periodically set intervals, or in the event a device alteration occurs in the network(added or removed a device). When a packet is incoming to switch, it will have an IP address, in the header to where it must be forwarded. The switch then uses the ARP lookup table, to translate the IP address into the MAC address, to where the packet must be finally forwarded, and stores it in the ARP cache. In the event the IP to MAC address translation already exists, the ARP request is not necessary(this occurs for subsequent pings).

Question 2. Ping from one host to another. You should see that at the very beginning,

the RTT is significantly longer. Why? Stop pinging and wait for about 30 seconds,

and try again. Is the RTT now stable? Why?

**Answer:-** The reason the RTT of the first ping is longer is because the ARP table is not yet populated. The ping request is placed on hold, to send out an ARP broadcast, to learn the IP addresses and corresponding MAC addresses of the devices connected on the network. The response of the ARP broadcast is received. Then the ARP table is used the translate the IP address in the packet header, to the corresponding MAC address, and then the first ping is sent through. The translated address is stored in the ARP cache. Due to this the RTT of the first ping is significantly longer. The subsequent ping requests, do not have to wait even for the ARP request, as the IP address to the MAC address translation already exists in the ARP cache, hence the RTT is significantly shorter.

On waiting for 30 seconds and then sending the ping requests, the RRT of the first ping is reduced significantly, as is evident from the below screenshot. This is because the ARP table is already populated, and there no need to send out an ARP broadcast. Hence, a significant amount of time is reduced. However, the ARP request needs to be generated as the ARP cache is refreshed periodically, hence the RTT of the first ping is lesser the second time, but still greater than the subsequent pings. As explained previously, the subsequent pings do not need to wait for the ARP request as the IP address to MAC address translation already exists!Image

Question 3. Plot the Round Trip Time (RTT) evolution over time. You need to

produce a plot with the runtime as the x-axis and the RTT time as the y-axis. You

need two curves, the RTT from h1 to h2, and the RTT from h2 to h1.One measurement

is not enough! You will need to restart both the network and the controller many times

and average over multiple measures.

Remember, if Mininet crashes or has any problem, use the command:$ sudo mn -c

To demonstrate this graph we are ping from h2 to h1 5 times, using the command h2 ping h1 -c 5.

| Avg RRT(y axis in ms ) | Runtime(x axis in ms) |
| --- | --- |
| 157.306 | 4070 |
| 385.612 | 4057 |
| 553.333 | 4057 |
| 893.718 | 4062 |
| 546.157 | 4071 |

To demonstrate this graph we are ping from h2 to h1 5 times, using the command h1 ping h2 -c 5.

| Avg RRT(y axis in ms ) | Runtime(x axis in ms) |
| --- | --- |
| 98.664 | 4074 |
| 90.544 | 4057 |
| 95.328 | 4070 |
| 375.041 | 4066 |
| 887.822 | 4046 |

Please find below the graph of runtime vs Avg RTT, we have taken average values for each of the 5 entries shown above

Question 4. Explain the roles of the different packets you see. You can use http:

//flowgrammable.org/sdn/openflow/message-layer/ to understand them. We are

using Openflow 1.3.Do not describe all the packets, but only the most important ones!

Typically at the very beginning, once the switch contacts the controller, and when a

new packet is received.

**Answer:-**

OFPT\_HELLO - It is a packet containing the OpenFlow header. It contains the highest version number of openflow supported by the switch and controller respectively.

OFPT\_PORT\_STATUS - Is used to notify a change in ports, changes in read only flags,etc to the controller

OFPT\_FEATURES\_REQUEST - Used by controller to request identity of the switch and read it's basic features

OFPT\_FEATURES\_REPLY - Used by switch to respond it's identity and it's basic features

OFPT\_MULTIPART\_REQUEST - Used in conjunction with OFPT\_MULTIPART\_REPLY to transmit large chunks of data which is greater than 64KB

OFPT\_GET\_CONFIG\_REQUEST - It is the sequence of messages is used to query and set the fragmentation handling properties of the packet processing pipeline. Should packets that are fragmented be: matched as-is, dropped, reassembled and matched, etc. Additionally, this message helps determine how much of a packet will be shared with the controller. GetConfigReq is an acknowledged message (GetConfigRes) and is only initiated by the controller.

OFPT\_BARRIER\_REPLY - It is the State modification messages from the controller may be executed in an arbitrary order by the switch. A barrier request can be used by the controller to set a synchronisation point, ensuring that all previous state messages are completed before the barrier response is sent back to the controller.

OFPT\_GET\_CONFIG\_REPLY - It is the sequence of messages is used to query and set the fragmentation handling properties of the packet processing pipeline. Should packets that are fragmented be: matched as-is, dropped, reassembled and matched, etc. Additionally, this message helps determine how much of a packet will be shared with the controller. GetConfigReq is an acknowledged message (GetConfigRes) and is only initiated by the controller. The controller may alter this state with a SetConfig message. SetConfig can only be initiated by the controller, and is unacknowledged.

OFPT\_ROLE\_REQUEST - It is the set of messages used by the controller to modify its role among multiple controllers on a switch.

OFPT\_PACKET\_IN - It is used to send a captured packet by the switch to the controller

OFPT\_PACKET\_OUT - It is used to inject a packet into the dataplane of the switch

OFPT\_FLOW\_MOD - It is used by the controller to modify the state of an Openflow Switch

OFPT\_GROUP\_MODE - The controller uses GroupMod messages to modify group tables.

Question 5. Explain what each flow represents. Briefly explain what each field is

used for.

**Answer:-**

On running the mentioned command three flows were observed.

The first flow is between the host h1 and host h2. In this flow, h1 is the source and h2 is the destination.

The second flow is between the host h2 and h1, here h2 is the source and h1 is the destination.

The third flow is between the controller and the switch.

In the each of the flows the various fields are:-

Cookie- It represents an identifier for the flow entry.

Duration - It represents the amount of time, the flow has been active

Table- It is the flow table in which the packets will be checked

n\_packets- It represents the number of packets sent

n\_bytes- It represents the number of bytes sent

idle\_timeout - It is the amount of time after which the flow is dropped.

Priority - This field is used for matching precedence of the flow entry. When a packet is matched with the flow table, only the highest priority flow entry that matches the packet is selected.

in\_port- This field represents the packet ingress port. It can be used only as an output port, send the packet out through its ingress port.

dl\_src - It is the MAC address for the source

dl\_dst - It is the MAC address for the destination

nw\_src - It is the IP address for the source

nw\_dst - It is the IP address for the destination

actions - Actions contains the port number and the type of flow and the output for the dataplane and the controller for the control plane

Image

Question 6. Imagine now that a web server is running on h1. h2 starts an HTTP

request to h1. What will happen? Describe briefly the messages exchanged between

the switch and the controller, and the flows that would appear in the dump-flows result.

Note: you can try to emulate the behavior by opening a terminal on h2 and by using

the wget command and python -m SimpleHTTPServer.

**Answer:-** When running the above mentioned configuration, the messages exchanged between the switch and the controller are:-

OFPT\_PACKET\_IN - The Server first receives SYN flag which is used for initiating TCP handshake

OFPT\_FLOW\_MOD -> In this message, the controller requests the switch to change its state to output. Hence the OFPT\_PACKET\_OUT is the very next packet that was passed.

OFPT\_PACKET\_OUT -> This messages represents the real data transfer between h2 and h1, which occurs on the data plane. Subsequently the SYN and ACK are received by client side.

OFPT\_PACKET\_IN -> The Server obtains the ACK flag from the client for completing the TCP handshake

Image

There were three flows observed. The first flow is between the client h2 and the server h1. In this case, the source is h2, the client and the destination is the server h1.

The second flow is between, the between the server h1 and the client h2. In this case, the source is h1, the server and the destination is the client h2

The third flow is between the network and controller.

Question 7. How many switches are present? What kind of topology is it? How many

switches must be programmed to ping from h1 to h8?

**Answer:-** In the given topology, there are 7 switches, this was observed in the switches tab, in the floodlight. The topology is a tree, this was observed from the topology tab in floodlight, the same was also mentioned in the command used for creating the topology.

To ping from h1 to h8, 5 switches needs to be configured. If we look at the topology, h8 is connected to switch s7, s7 is subsequently connected to s5, then we can see that s5 is connected to s1. s1 is then connected to s2, and finally s2 is connected to s3, which is connected to h1. So in order to ping from h1 to h8, 5 switches namely s3,s2,s1,s5,s7 needs to be configured.

Question 8. How does the first RTT evolve with the number of switches to cross?

To get the requisite Avg RTTs, we ping from host exactly 5 times using the command “h1 ping hx -c 5”, where x = 2 to 7.

| Ping | Avg RTT(in ms) |
| --- | --- |
| h1 to h2 | 17.457 |
| h1 to h3 | 24.953 |
| h1 to h4 | 30.573 |
| h1 to h5 | 35.785 |
| h1 to h6 | 69.437 |
| h1 to h7 | 82.193 |

In the process of pinging from the h1 to each of the host, by restarting the controller every time, we saw, that the RTT values where increasing linearly. The average value observed as mentioned in the above table. Another observation regarding the pings, was that, as the we were pinging further from h1, especially to h7 and h6, multiple time we received “destination host unreachable”, and all packet were lost.