REPORT

G.Sai Bharath Chandra -14CS10020
R.Sri Charan Reddy -14CS10037

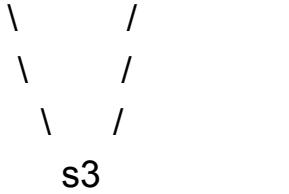
Instructions:

In the topology module, controller is by default the actual module itself. But for the case of Spanning tree, the inability of controller to eradicate the looping of packet is what makes the complete topology useless and inefficient. So, Controller is given the chance to analyze the traffic and channelize it in such a way, that there is no duplication of packets at a node more than once. And if we make the controller to set up the spanning tree in the given network and in someway or the other, make it allow packets via specific paths, disabling redundant and unnecessary links, We can avoid this packet looping condition.

In mininet, there already exists a remote controller namely pox controller, We shall provide the controller rights to this and let it create spanning tree

Capture the data in Wireshark and observe them separately.

Create topology h1-------s1---------s2-------h2 using stp.py topo file



1) For switches without spanning tree protocol:

>> run sudo "mn --custom stp.py --topo recttopo " on mininet virtualmachine
>> xterm h1,h2,s1,s2,s3
>> For each xterm open a wireshark instance by typing "sudo wireshark &" and start packet capture
>> run iperf server on h2 --> do "iperf -s" on h2
>> run iperf client on h1 --> do "iperf -c 10.0.0.2" on h1
---

2) Implementing switches using spanning tree protocol(stp):

>> run sudo "mn --custom stp.py --topo recttopo --controller remote --switch ovsk --mac" on mininet virtual machine

>> h1's mac== 00:00:00:00:00:01
>> h2's mac== 00:00:00:00:00:02

>> we use a remote pox controller by running

"pox/pox.py --verbose openflow.spanning_tree --no-flood --hold-down openflow.discovery forwarding.l2_multi"

>> xterm h1,h2,s1,s2,s3
>> For each xterm open a wireshark instance by typing "sudo wireshark &" and start packet capture
>> run iperf server on h2 --> do "iperf -s" on h2
>> run iperf client on h1 --> do "iperf -c 10.0.0.2" on h1

The path from h1 to h2 here according to the results is h1----s1----s2-----h2 using stp protocol

Hence we capture no packets from either h1 or h2 at either s3-eth1 or s3-eth2

The assignment deals with a network having a cycle and the various ways to control data traffic and data duplication. In the network without spanning tree protocol for this assignment:

1) H1 sends a packet to s1.
2) S1 sends it to s2 and s3
3) S2 duplicates it and sends it to s3
4) S3 duplicates it and sends it to s1
5) This leads to an unending loop of a packet and makes data transmission impossible

To avoid this, we build a spanning tree across the network, with new edges as those of the spanning tree edges. In this way, there would be only a single path from one node to another.

Let us extend our observation to the given network topology

Node H1:

Once a request is made for transmission from h1 to h2, it travels via clear path and receives acknowledgment. There isn't any packet loss, in the given network.

Once a request is made for transmission from h1 to h2 , it will ask its s1 for the location of h2, which in turn asks s2 and s3, they again ask among themselves, thus flooding the network creating multiple broadcast requests due to the loop in a never ending manner. So h1 could not get any response for location of h2 within the TTL of the broadcast request, thus assumes there is no host to route.

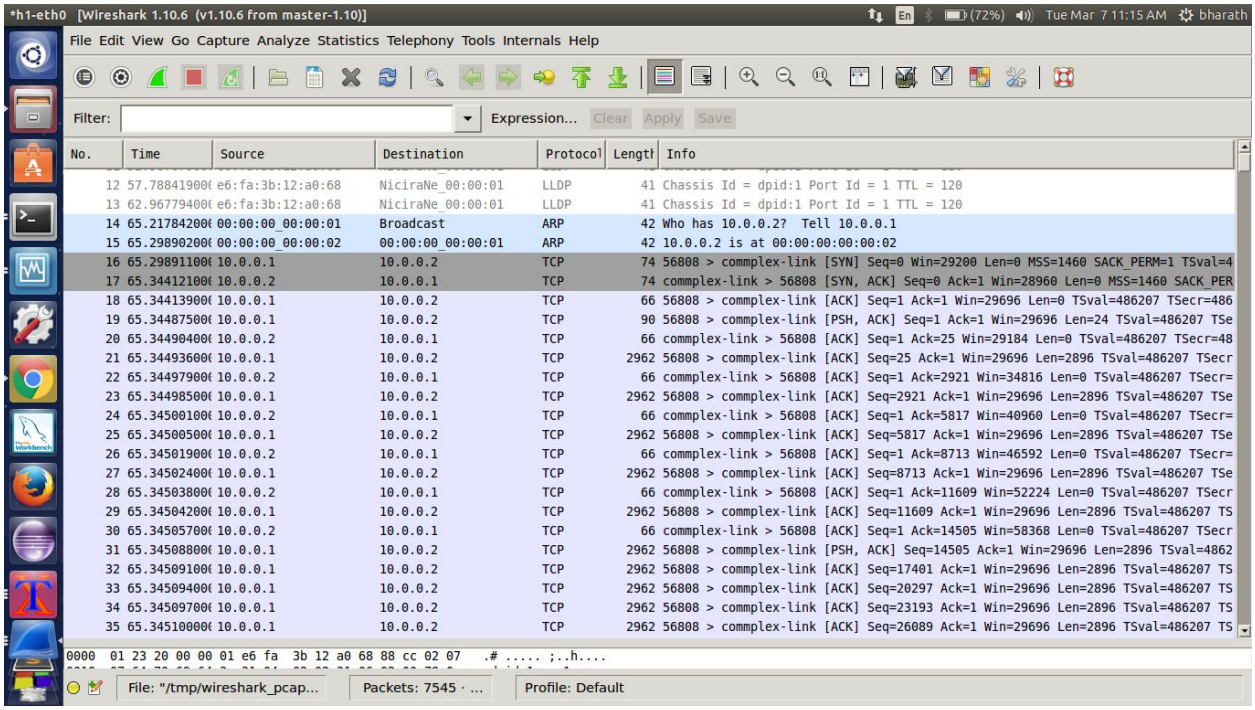
Node H2:

Since it is TCP network and the links are lossless links ,same Goes with the node H2, without spanning tree, it doesn't even receive any data transfer from h1, since h1 cannot get the location of h2 it doesn't send any packet data to the network and with spanning tree, it can receive data and send acknowledgments along the unique path to h1.

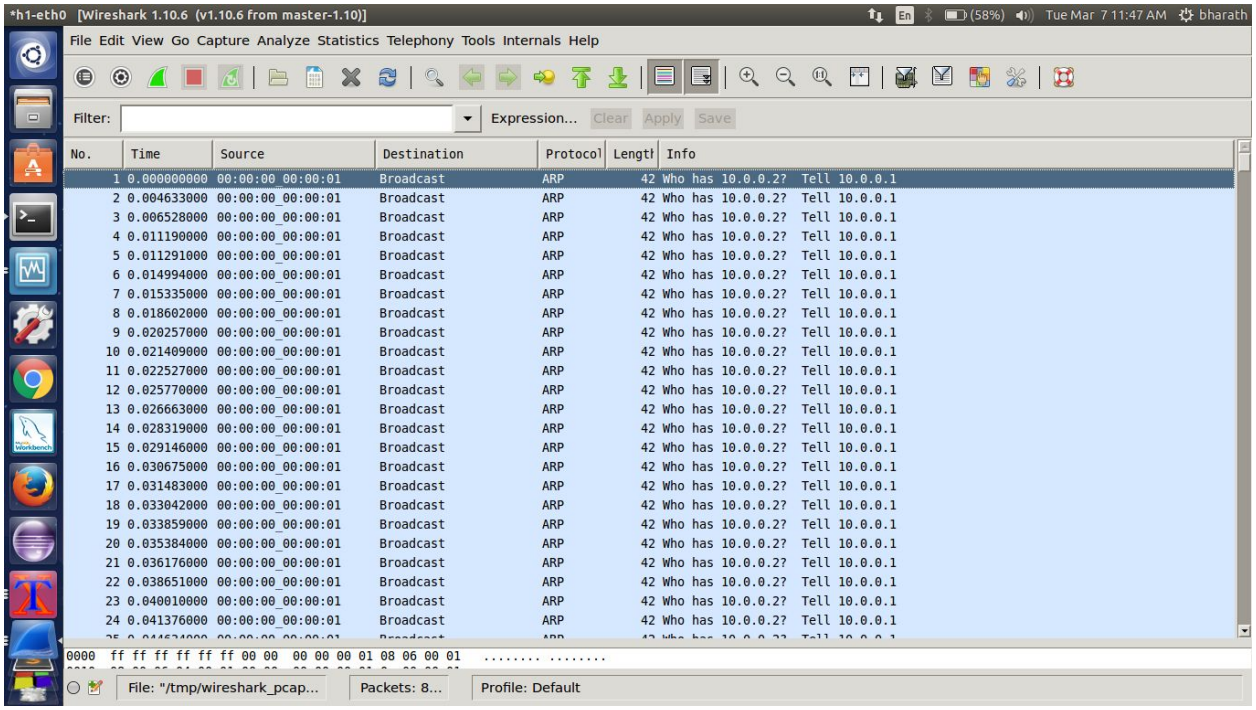
Switches:

For Switches S1, S2, S3, they are data transferring modules due to which, they receive and send unending cycles of packets in a case without spanning tree. And in spanning tree case, the data is received and send by a switch without any problems.

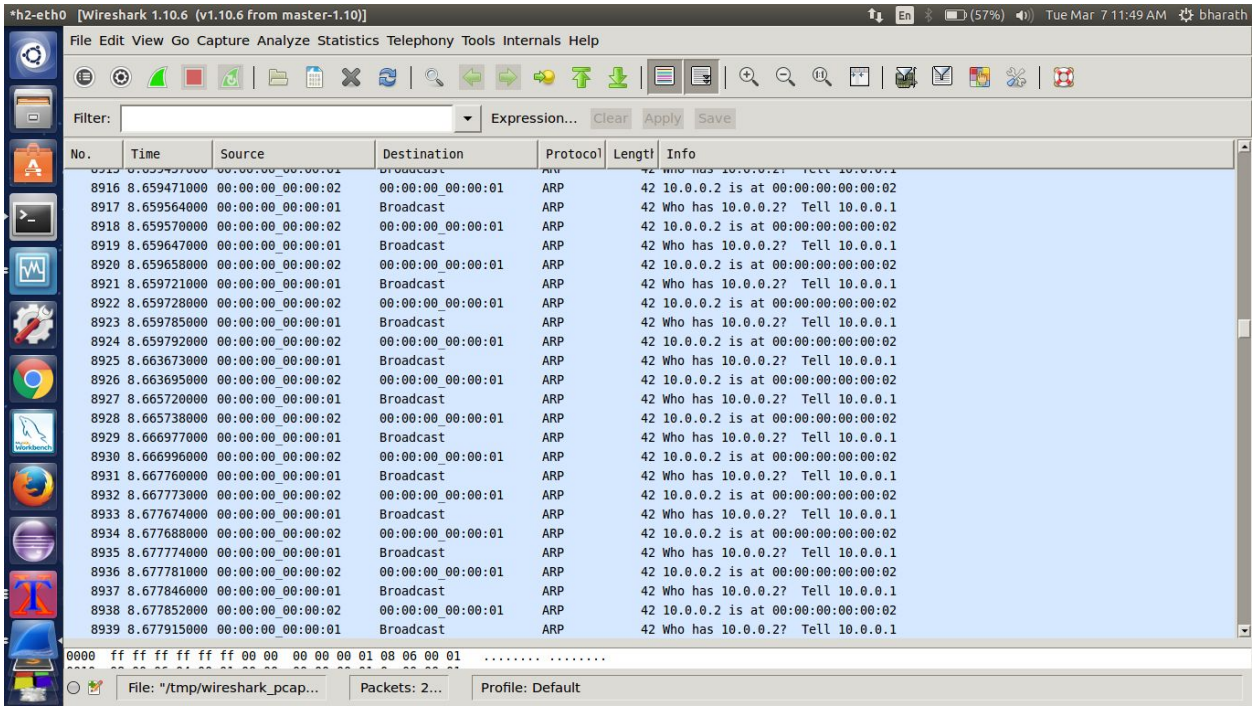
Node H1 with Spanning Tree:



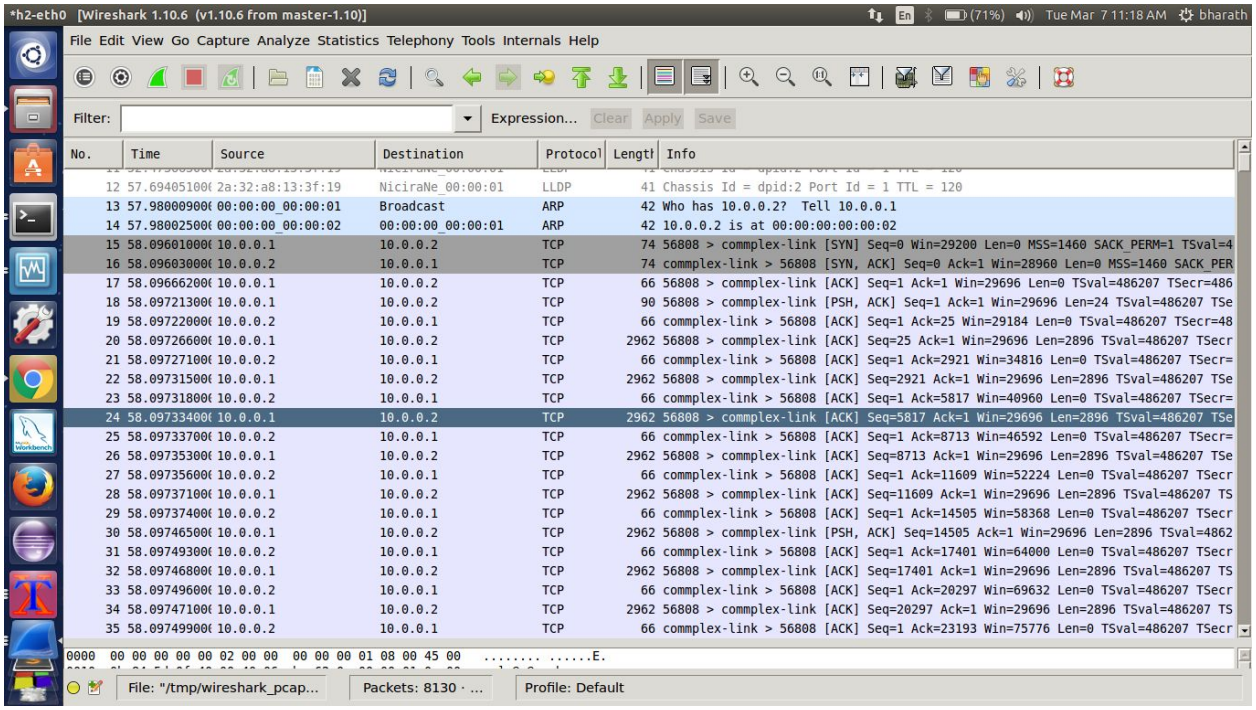
Node H1 without Spanning Tree:



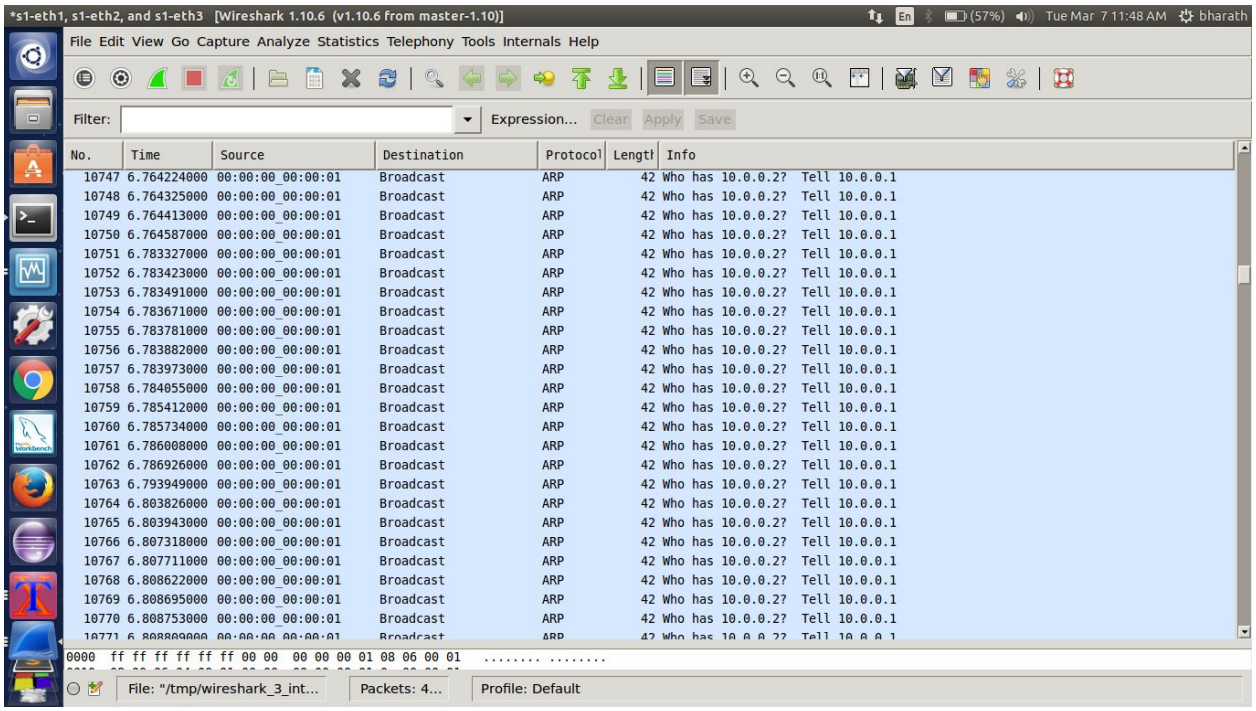
Node H2 without Spanning Tree:



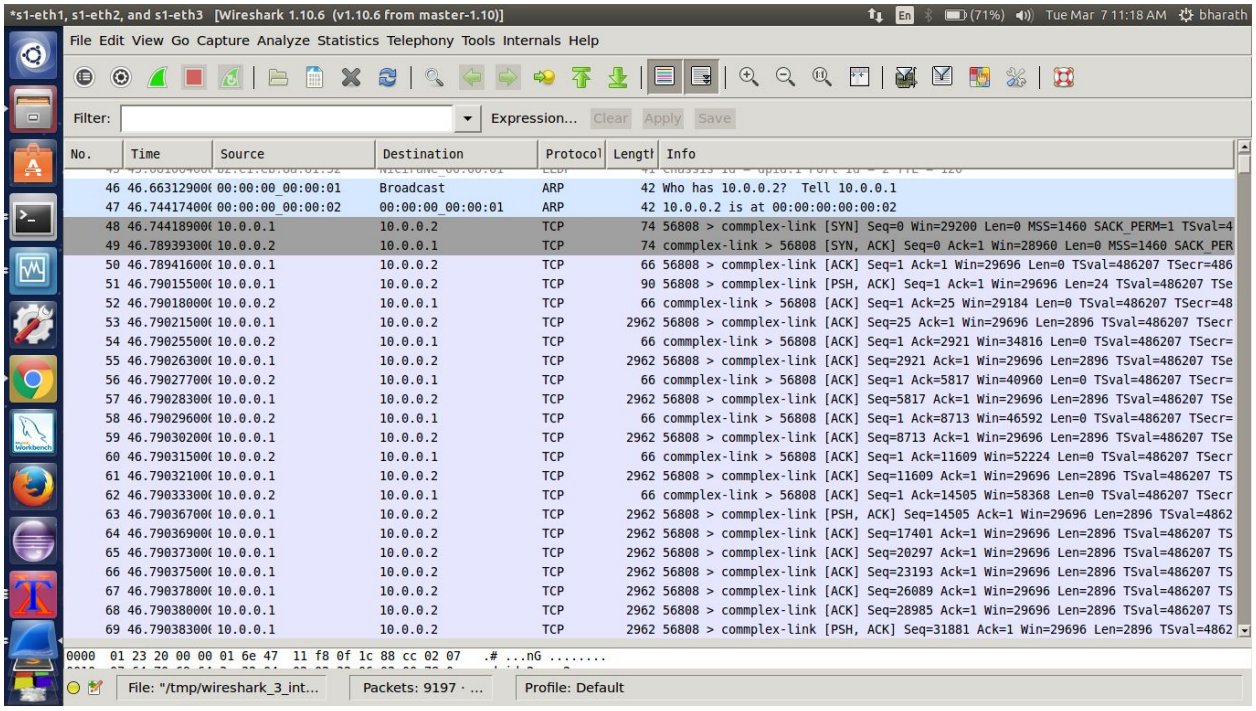
Node H2 with Spanning Tree:



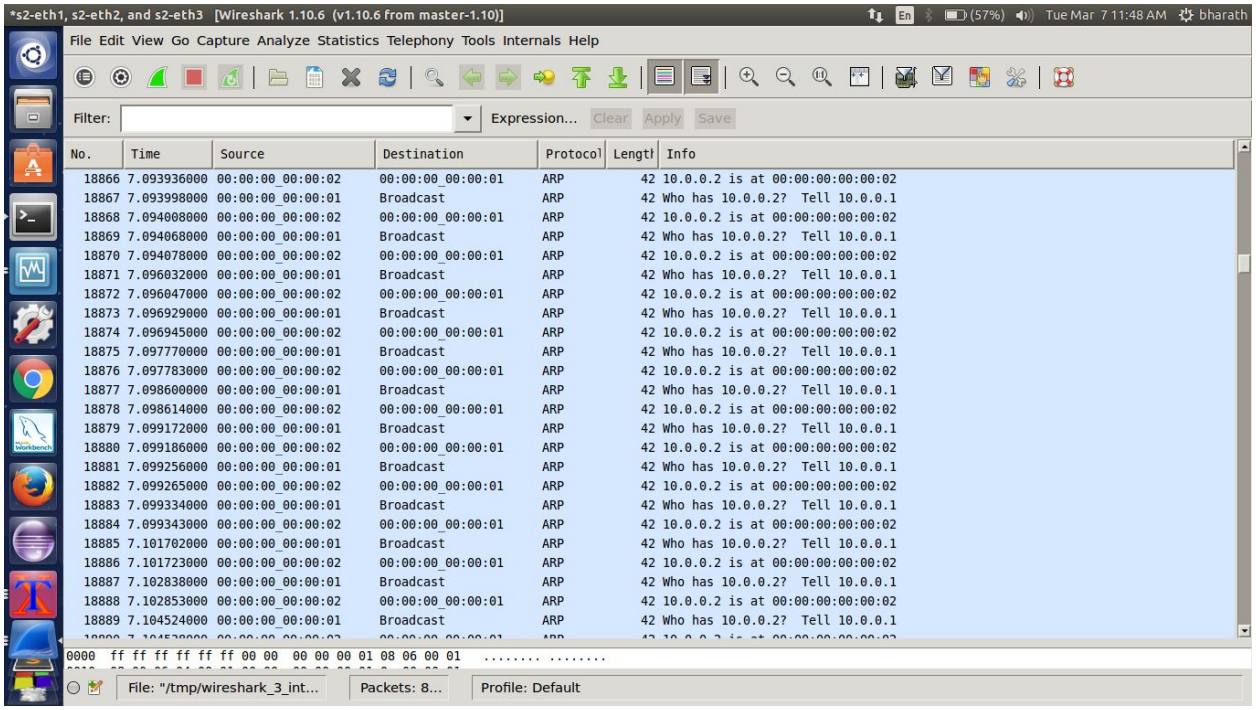
Switch S1 without SPT:



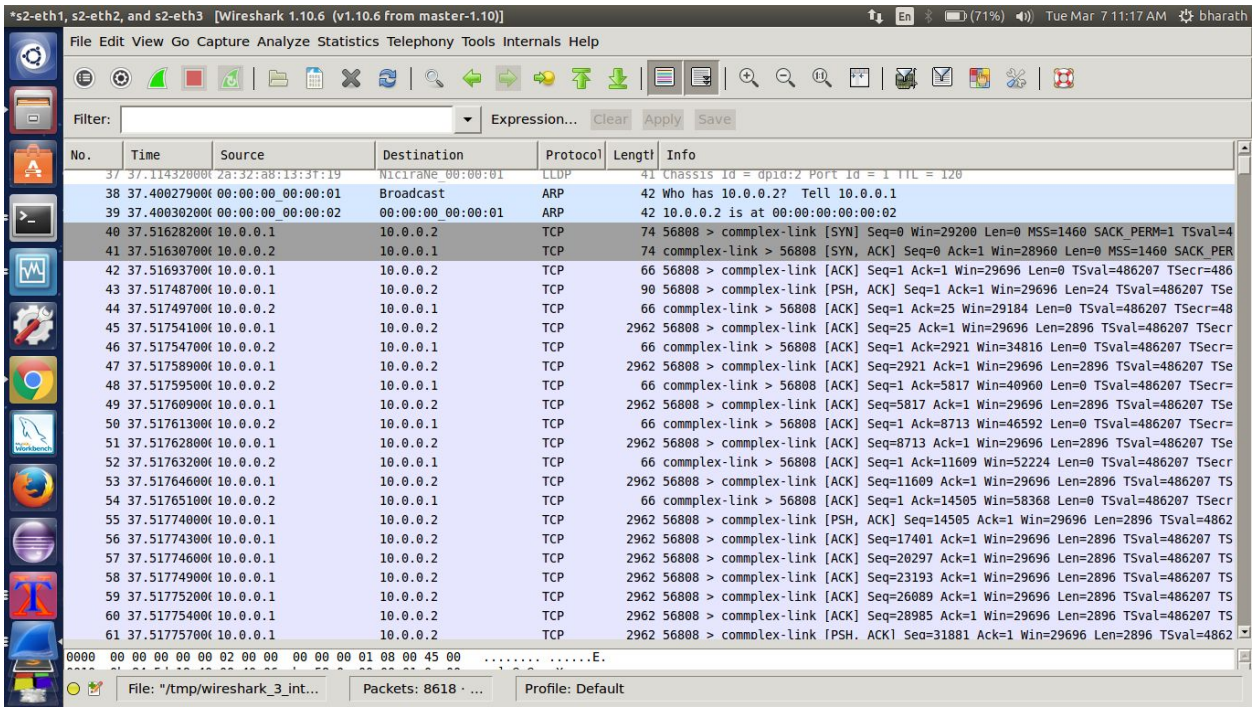
Switch S1 with SPT:



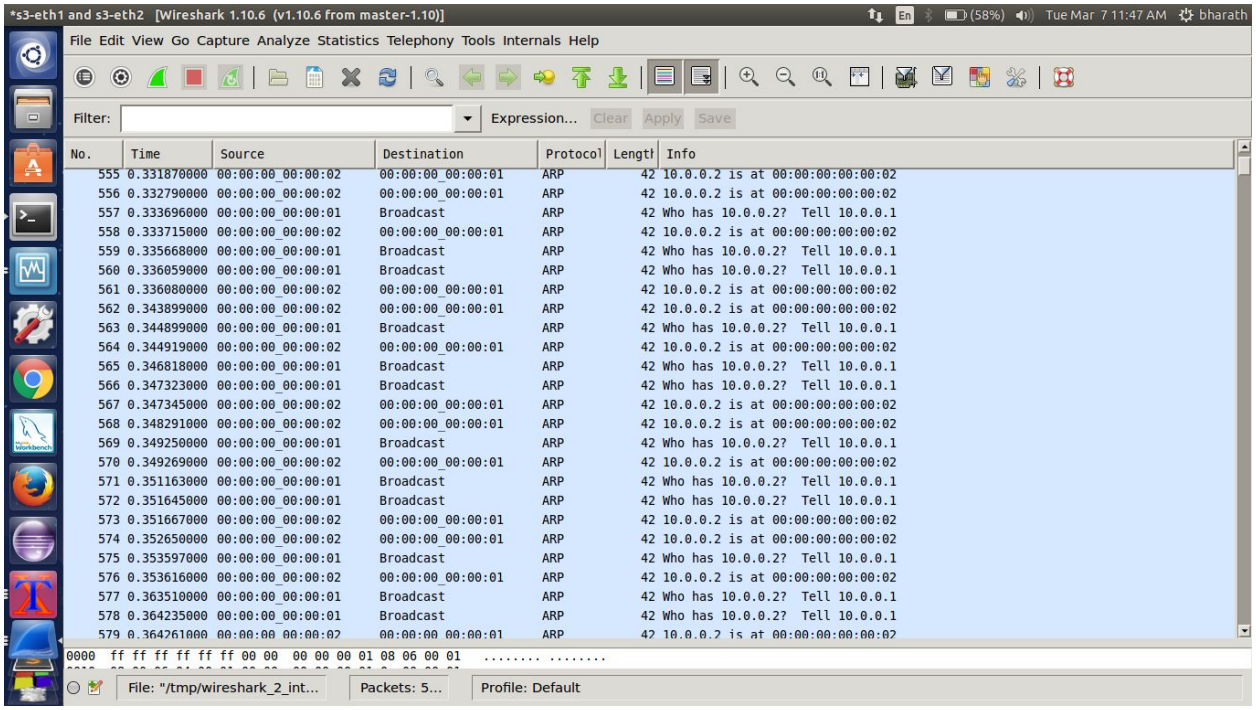
Switch S2 without SPT:



Switch S2 with SPT:



Switch S3 without SPT:



Switch S3 with SPT: