Task 1:

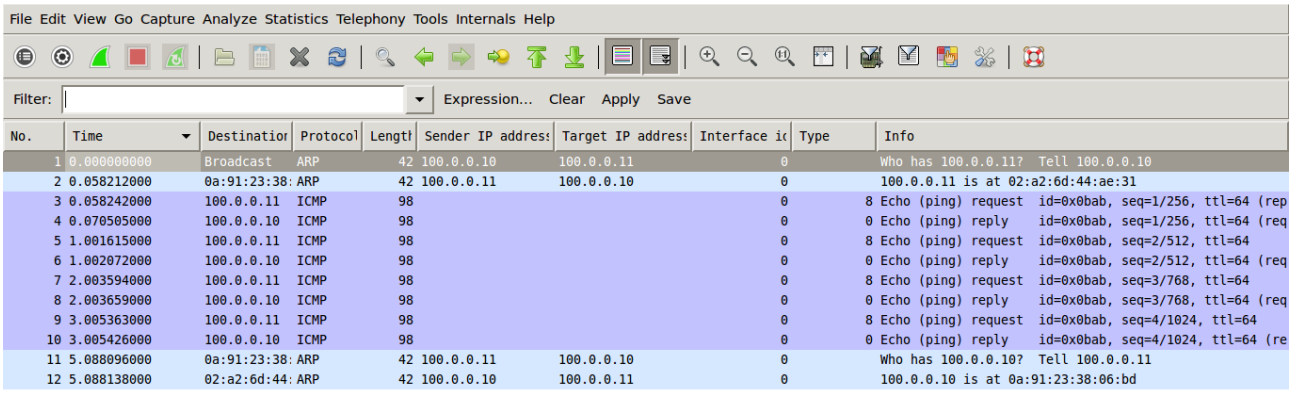
wireshark on sw1

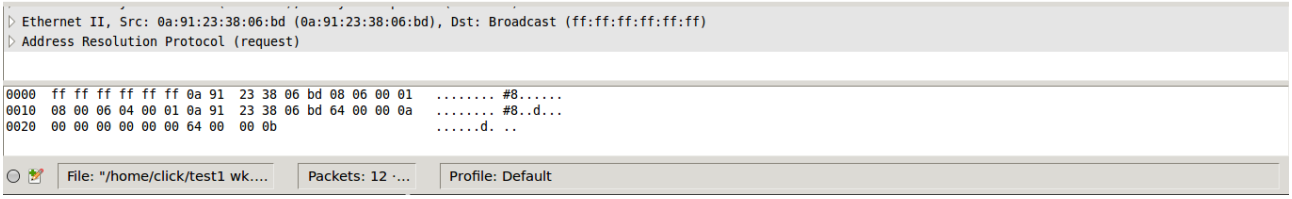
mininet> h1 ping -c 4 h2
PING 100.0.0.11 (100.0.0.11) 56(84) bytes of data.
64 bytes from 100.0.0.11: icmp_seq=1 ttl=64 time=70.5 ms
64 bytes from 100.0.0.11: icmp_seq=2 ttl=64 time=0.512 ms
64 bytes from 100.0.0.11: icmp_seq=3 ttl=64 time=0.120 ms
64 bytes from 100.0.0.11: icmp_seq=4 ttl=64 time=0.115 ms

--- 100.0.0.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 0.115/17.834/70.589/30.458 ms

Open flow table of sw1:
command: sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=5.507s, table=0, n_packets=1, n_bytes=42, idle_timeout=10, hard_timeout=30, idle_age=5, priority=65535,arp,in_port=2,vlan_tci=0x0000,dl_src=02:a2:6d:44:ae:31,dl_dst=0a:91:23:38:06:bd, arp_spa=100.0.0.11,arp_tpa=100.0.0.10,arp_op=1 actions=output:1
cookie=0x0, duration=5.504s, table=0, n_packets=1, n_bytes=42, idle_timeout=10, hard_timeout=30, idle_age=5, priority=65535,arp,in_port=1,vlan_tci=0x0000,dl_src=0a:91:23:38:06:bd,dl_dst=02:a2:6d:44:ae:31, arp_spa=100.0.0.10,arp_tpa=100.0.0.11,arp_op=2 actions=output:2
cookie=0x0, duration=10.56s, table=0, n_packets=4, n_bytes=392, idle_timeout=10, hard_timeout=30, idle_age=7, priority=65535,icmp,in_port=1,vlan_tci=0x0000,dl_src=0a:91:23:38:06:bd,dl_dst=02:a2:6d:44:ae:31,nw_src=100.0.0.10,nw_dst=100.0.0.11,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:2
cookie=0x0, duration=10.555s, table=0, n_packets=4, n_bytes=392, idle_timeout=10, hard_timeout=30, idle_age=7, priority=65535,icmp,in_port=2,vlan_tci=0x0000,dl_src=02:a2:6d:44:ae:31,dl_dst=0a:91:23:38:06:bd,nw_src=100.0.0.11,nw_dst=100.0.0.10,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:1

Wireshark:





Series of events:

1. h1 does ARP req for h2’s mac addr.
2. SW1 floods this, while learning h1’s mac to port mapping.
3. h2 does ARP reply
4. SWI dlivers this sdirectly to h1 and learns h2’s port to mac info
5. h1 sends the ICMP Ping packet which is delivered by the learning switch to h2, and then a PING reply is sent back.

…….

Task 2:

Topology built, every device is a learning switch.

>pingall
Succedded for all hosts.

Now firewalls enabled

mininet> h1 ping -c 4 h3
PING 10.0.0.50 (10.0.0.50) 56(84) bytes of data.

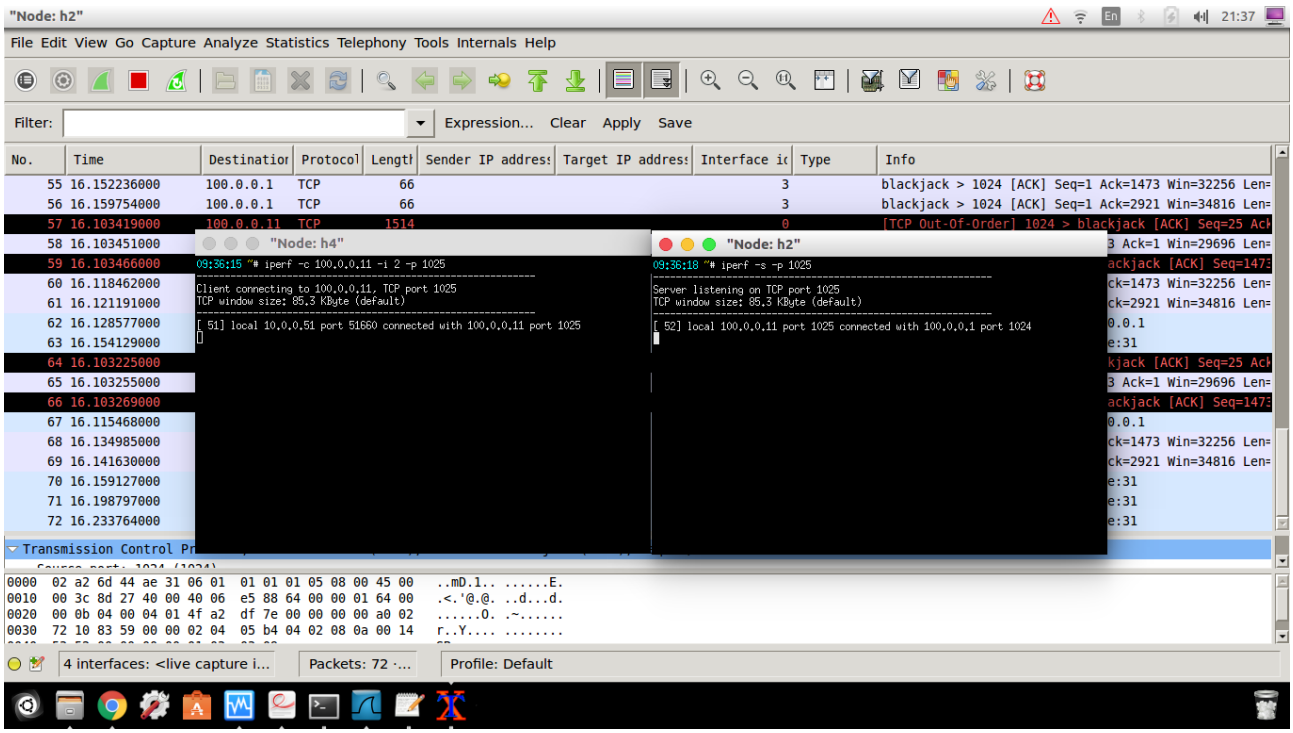
--- 10.0.0.50 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3005ms

h1 unable to reach h3 since its the private zone!

mininet> h3 ping -c 4 h1
PING 100.0.0.10 (100.0.0.10) 56(84) bytes of data.
64 bytes from 100.0.0.10: icmp_seq=1 ttl=64 time=84.2 ms
64 bytes from 100.0.0.10: icmp_seq=2 ttl=64 time=2.07 ms
64 bytes from 100.0.0.10: icmp_seq=3 ttl=64 time=1.16 ms
64 bytes from 100.0.0.10: icmp_seq=4 ttl=64 time=1.13 ms

--- 100.0.0.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3025ms
rtt min/avg/max/mdev = 1.138/22.147/84.211/35.834 ms

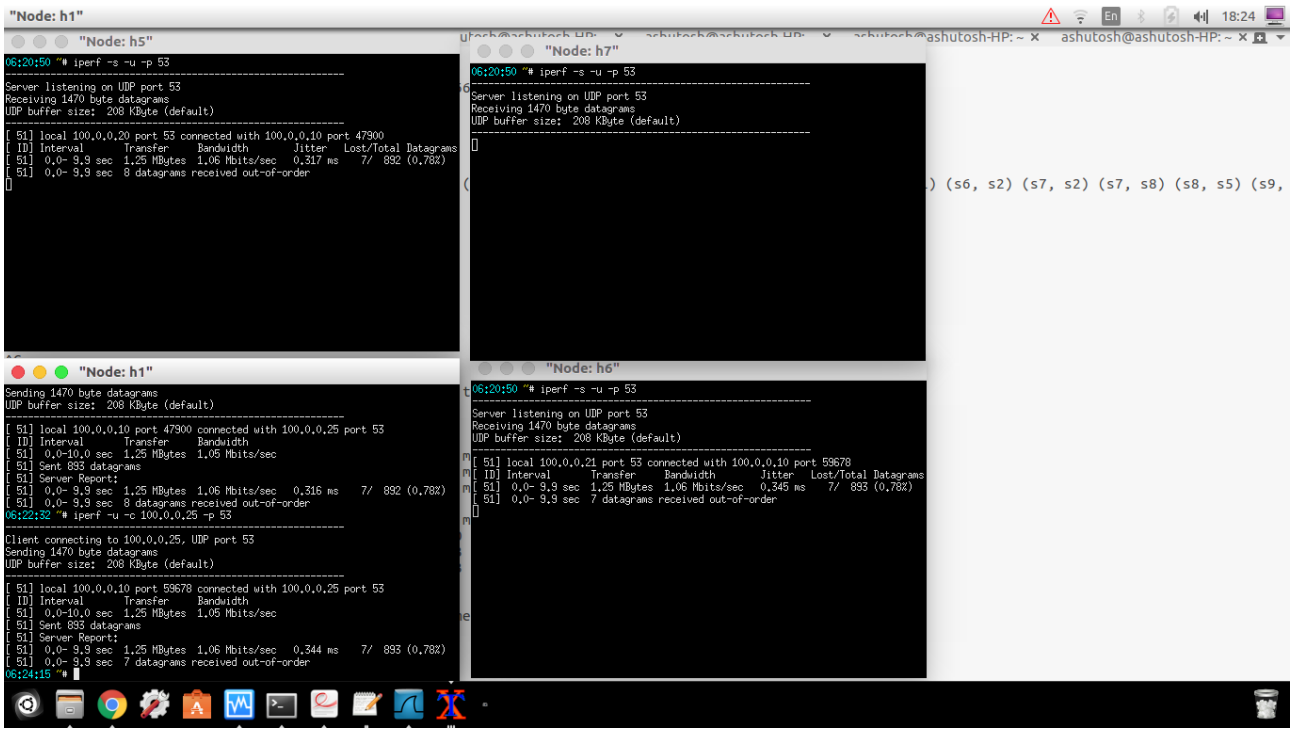
h3 was able to easily reach h1!



Here, Starting server at h2 port 1025 and client at h2:
Connection was possible. However the other way round, communication was not possible.

……

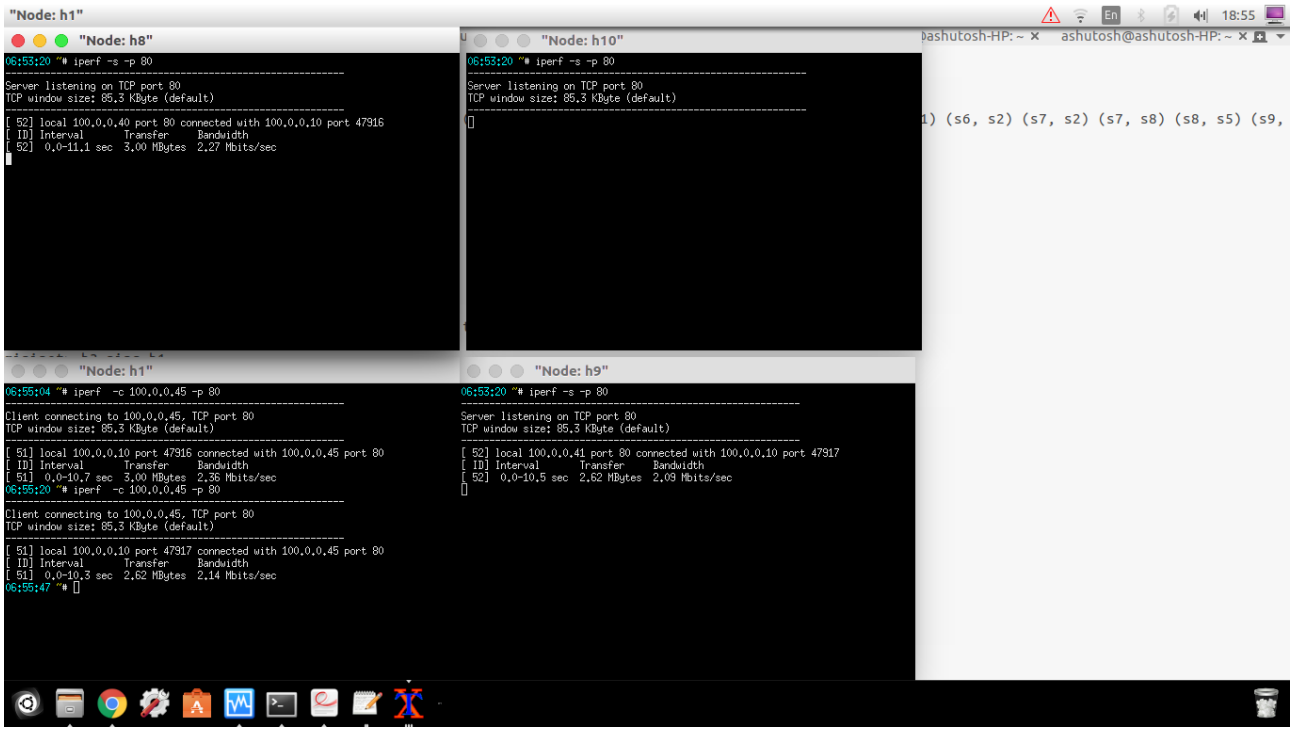
Task 3:



Here we can see **load balancer** in action.

H1 is udp client, where as the h5, h6 and h7 are udp servers.

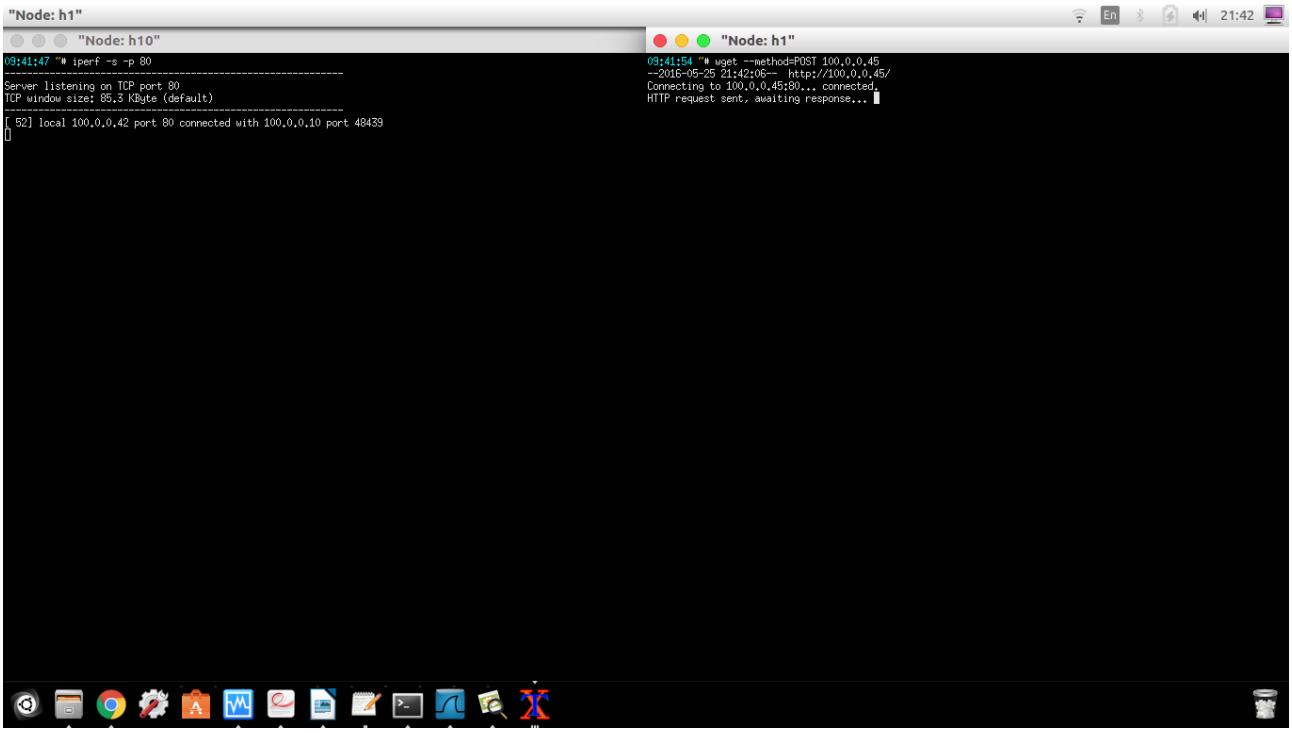
When h1 is doing consecutive **UDP** session, the session is being round robin amongst the three servers.



Here we can see similar **load balancing** with the **HTTP** servers. H1 is the client and h8, h9 and h10 are HTTP servers. Again the sessions are getting round robbin amongst the three servers.

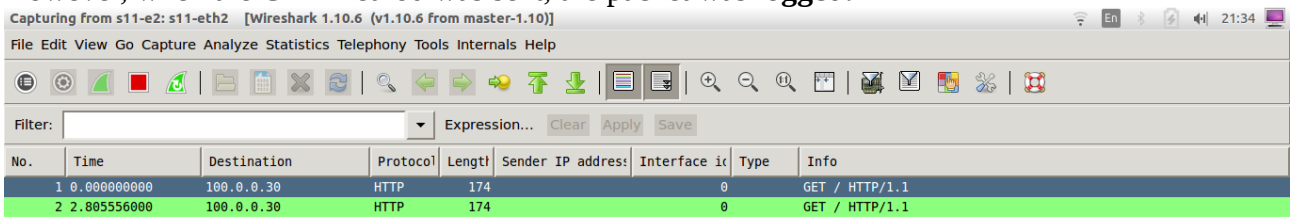
……

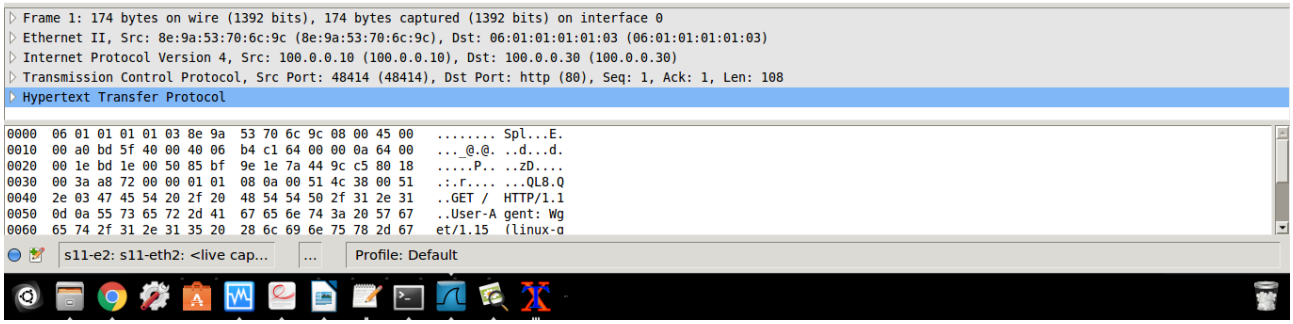
Working of IDS:



In the above screenshot you can see that h1 was able to send a POST method to the server through the ids.

However, when the GET method was sent, the packet was **logged.**





This the wireshark capture at the interface connected to the logging server.

Now, I sent an http **PUT** method packet with **DELETE** written in it.

Again, wireshark capture at the interface connected to the logging server shows that this packet was logged.

Thus, the above two examples show that IDS is checking the HTTP method type as well as content in the case of PUT.

……

Task 4:

Report files with counter and rate values is present in the tar ball with names ids_test.report, napt_test.report, lb_test.report