Christopher Brant

OSPF Routing in GENI

ECE 6380 Computer Communications

Machine Problem 3

Due 4/24/18

Summary

For this machine problem, we were tasked with learning about how to configure and utilize OSPF in a network, and how to monitor the routes throughout our network that we have created. We were also expected to learn how OSPF is configured, as well as how to monitor the status of the OSPF daemon and trace the routing throughout our network that we have created. Lastly, we were to explore how Dijkstra's algorithm routes packets in our network.

Implementation

My implementation includes a design that instantiates with 6 routers, 4 hosts, and 3 loops. Routers 1,4,5, and 6 are the only routers that have hosts connected to them as host-1, host-2, host-3, and host-4 are connected to router-1, router-4, router-5, and router-6 respectively. This implementation had 3 loops which were created using routers (1,2,6), (2,3,5,6), and (2,3,5,4). It was observed that by default, host-1 would communicate with host-2 directly through router-1 -> router-2 -> router-4, however when the link between router-1 and router-2 was brought down, or its cost was increased by 5x, the route changed to router-1 -> router-6 -> router-5 -> router-4 to get from host-1 to host-2, which was 3 hops rather than the 2 hops of the original implementation.

Results

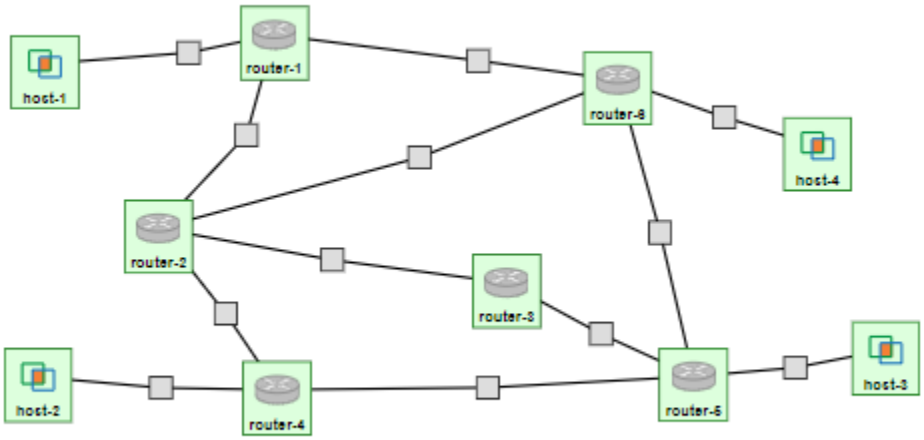

Figure 1: Example Layout of the Network

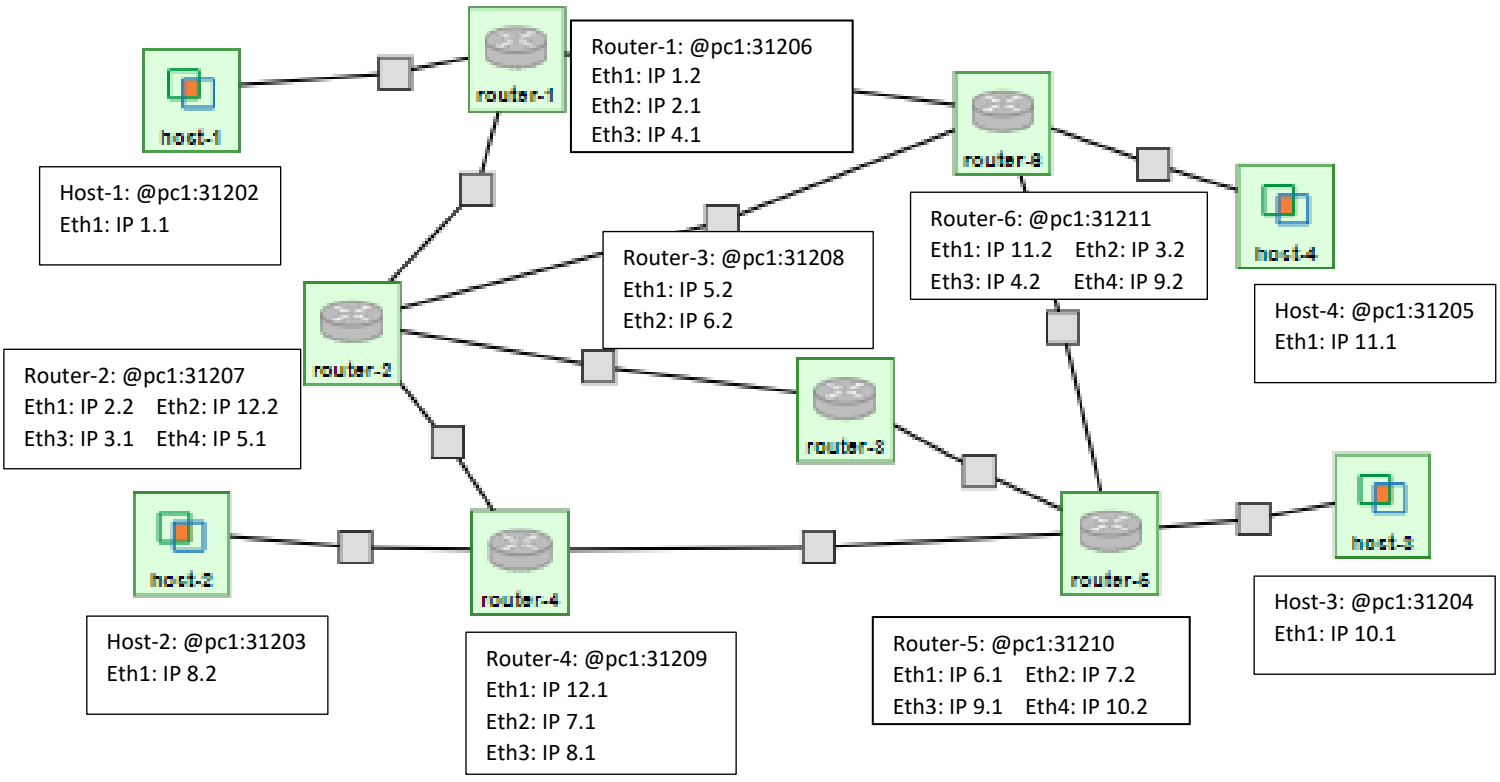

Figure 2: Subnet 10.10.0.0/16

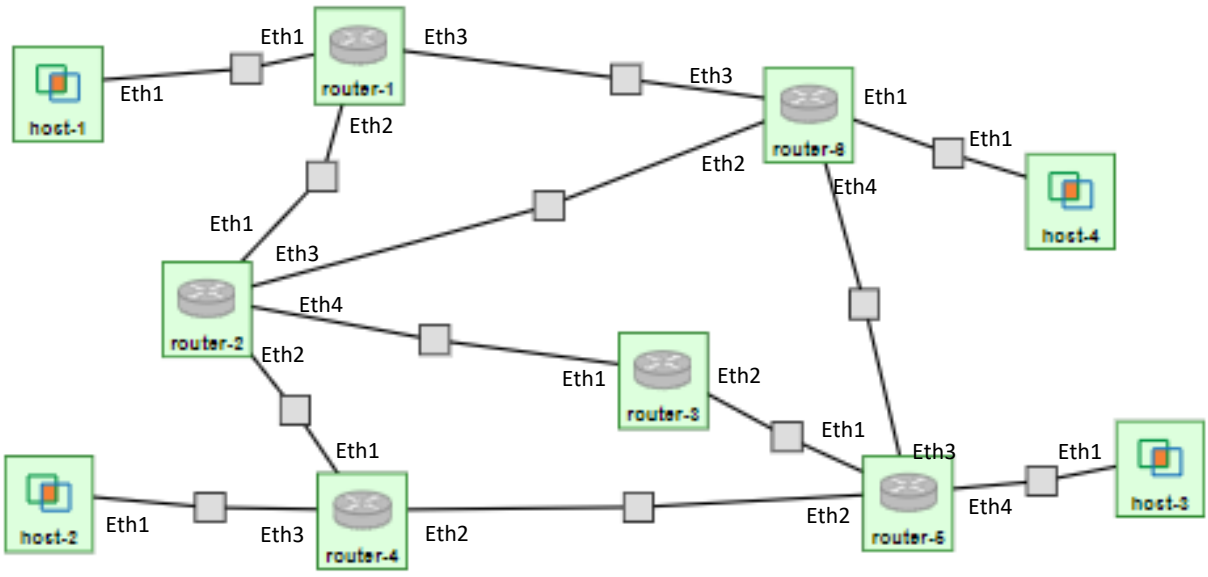

Figure 3: Network Interface Connections

The previous three figures are the results for question 2: Illustrate your network.

Question 1 Results:

Bash Script is as follows:

```
#!/bin/bash
# ECE 6380
# Spring 2018
# Clemson University
# Christopher Brant
# This script dprints out interface names, IP addresses, subnet numbers, and mask lengths
# for any router in the given network
# This script is based on one from:
# Harlan Russell
# Clemson University
# April 24th, 2018
##
  #make list of all eth interfaces expect 0
  ETHNUMS=`ip addr show | grep -Eo 'eth[1-9][0-9]*$'`
  # for each interface add ip address with mask
  for ETH in ${ETHNUMS[@]} ; do
    # this extracts the full IP address with mask
    ETHIP=`ip addr | grep $ETH$ | awk -F " " '{print $2}'`
    INDIP="${ETHIP%%/*}"
    SUBMASK="${ETHIP: (-2)}"
    SUBNET="${ETHIP%.*/$SUBMASK}"
    SUBNETIP="$SUBNET.0/$SUBMASK"
    sudo echo "$ETH has IP address $INDIP and connects to subnet $SUBNETIP"
  done
```

Example Output when run on router-1
eth1 has IP address 10.10.1.2 and connects to subnet 10.10.1.0/24
eth2 has IP address 10.10.2.1 and connects to subnet 10.10.2.0/24
eth3 has IP address 10.10.4.1 and connects to subnet 10.10.4.0/24

Question 2 Results: Listed above as the three figures given in the Results section.

Question 3 Results: The link between router-1 and router-2, specifically router-1 eth2, is brought down such that the connection between host-1 and host-2 must take a new route as the previous route was from host-1 -> router-1 -> router-2 -> router-4 -> host-2.

The ospf neighbor table for router-1 prior to taking the link down had the following important information in it:

Neighbor	Dead Time	Address	Interface
10.10.12.2	34.708s	10.10.2.2	eth2:10.10.2.1
10.10.11.2	34.876s	10.10.4.2	eth3:10.10.4.1

After taking the link down and letting the network update, the ospf neighbor table for router-1 had the following important information in it:

Neighbor	Dead Time	Address	Interface
10.10.11.2	39.552s	10.10.4.2	eth3:10.10.4.1

When taking the link down, there was no packet loss, however the ttl count went from 61 to 60 for the packets that took the new route. The new route between host-1 and host-2 is as follows: host-1 -> router-1 -> router-6 -> router-5 -> router-4 -> host-2.

After bringing the link back up and letting the network update, the ospf neighbor table for router-1 had the following important information in it:

Neighbor	Dead Time	Address	Interface
10.10.12.2	33.495s	10.10.2.2	eth2:10.10.2.1
10.10.11.2	33.677s	10.10.4.2	eth3:10.10.4.1

When bringing the link back up, the same observation was made that the ttl count changed, however this time it went from 60 back to 61 once the routing tree updated itself. There also was rather significant packet loss when bringing the link back online, from host-1 to host-2 there was 16% packet loss out of 43 packets transmitted, and from host-2 to host-1 there was 18% packet loss out of 38 packets transmitted. From host-1 to host-2 there was an average RTT of 2.301 ms, and 2.301 ms * 7 dropped packets would approximate a time of 16.107 ms taken to update the route, which is supported by the host-2 to host-1 timing of average RTT of 2.381 ms, giving us 2.381 ms * 7 dropped packets, which approximates a time of 16.667 ms taken to update the route.

Question 4 Results: Traceroute showed me that taking the link between router-1 and router-2 to a cost a 50 instead of 10, changed the route from host-1 to host-2 from
host-1 -> router-1 -> router-2 -> router-4 -> host-2 to
host-1 -> router-1 -> router-6 -> router-5 -> router-4 -> host-2.

Running the command “sudo tcpdump -i eth1 ip -vv” on router-3 gives me an output of multiple LSP packets that are still making it to router 3 as it is the only router in the network other than router-2 that is not in the route between host-1 to host-2 now. Running for just a few seconds gave 4 LSP packets received by the filter.

Conclusion

The things that I take away from this machine problem that I did not already understand would be checking the individual routes used by each host to talk to another host within this network specifically such a network that uses Dijkstra’s algorithm. It was also interesting to learn about the process to configure the router link cost values as well as how the network is affected when the link is either taken down or brought back up. The main difficulty encountered, however, would be writing the bash script as I had entirely forgotten how to do string manipulation in a bash script, which is what the bottleneck in the time taken to complete this machine problem was for myself. One other slightly surprising hurdle that I had to overcome was actually getting the ping traces to produce problems, as my first attempt did not seem to introduce any issues within the network with regards to routing tables and routes. One neat extension that could be done with this machine problem in the future would be to have the students create a network that utilizes these routing protocols of OSPF, but each host would instead be a bridge that connects to a secondary extended LAN so it could be observed how both routing amongst a higher level protocol would work, while also observing how the packets would get forwarded to their destinations amongst the LANs as well.