ECE 4380/6380: Computer Communications **Spring 2018**
Machine Problem 3 **Due: 11:59 pm Tuesday, April 24**

For the final machine problem, choose one of two options. The first option is a GENI experiment to study OSPF routing in a wired network. The second is an experiment to build an ad hoc network using at least three laptops. Pick one of the two options and submit the lab report as described below.

Option 1: OSPF Routing in GENI

The goal of this machine problem is to experiment with IP routing and OSPF. Review the instructions in MP2 for accessing GENI. Here is a quick start guide, but see the tutorial below for full details.

- Go to https://portal.geni.net/ and click on the "Use GENI" link
- Select the Clemson University icon if visible, or use the "show a list of organizations" link
- From the Home menu select "Slices". Click on the "New slice" button to build a network. (Tip: Your MP2 slice has probably expired. But even if it did not, it is easier to start a new slice. The VM's for MP2 where configured to use OVS, and for this project OVS is not used. Furthermore, OVS and the routing software we will use in this project do not work well together. To avoid having to learn how to remove OVS configuration settings, it is quicker to just start with a new slice.)

General Notes

If you have system-specific problems related to GENI, first send email to our local GENI administrators at clemson-openflow-group@googlegroups.com. They can answer most questions related to using GENI for our project.

The tutorial required to get started is

https://clemsonopenflowgroup.atlassian.net/wiki/spaces/ece44006400sp18/pages/93388801/Introduction+to+GENI

Procedure

Create a new slice

Follow the directions on the tutorial web page to create a new slice. Create a network similar to the one you designed for MP2. For this assignment, all of the VMs can use the "Xen VM" image. However, you can also use the "OF OVS" image for the routers. Name at least six of the VMs as routers, using a naming pattern like router-1. Include at least 4 VMs to serve as hosts, and name them. Connect the routers with links so that there are at least 3 loops, and attach each host to one router. We will configure the routers to run OSPF. No additional configuration will be required for the hosts. The "Xen VM" and "OF OVS" VM's default to an acceptable image.

Tip: if you saved your rspec design from MP2 before you reserved any resources, you can reuse the rspec and simply change the name of the nodes from bridges to routers.

Reserve Resources

Make sure the site you have selected is one of the "InstaGENI" options. As described in the tutorial, check this web page to see which InstaGENI sites are not too heavily loaded: http://groups.geni.net/geni/wiki/ExpGraphs. Recently, this page has not provided reliable information so you still need to be prepared to try different sites.

This web page shows you the status (since a site with a low load might also be down): https://portal.geni.net/amstatus.php

Illustrate Figure

Because there are a large number of IP addresses and Ethernet interfaces to track, make an illustration with the details for your network. SSH into each host and router (review the tutorial for details on how to use your private key with **`ssh`**). On each VM use **`ip addr show`** to determine the IP address that is assigned to each Ethernet interface. You can use the rspec file to find the machine name and port number for each VM. Recall that the interface **`eth0`** should not be used in any of our experiments.

Configure Routers

The script **`routeconfig.sh`** is provided to automatically install quagga and set up the configuration files. Study the configuration script and notice it accomplishes the following tasks.

Disable the static routes installed by GENI
Download and install **`quagga`** (and **`traceroute`** if not already installed)
Set up the daemons file to allow **`zebra`** and **`ospfd`** to be started automatically
Build a list of the interface names excluding **`eth0`**
Set up the **`zebra`** configuration file with the IP address and mask for each interface
Set up the **`ospfd`** configuration file with a list of interfaces over which to advertise OSPF packets, and the subnetwork numbers with masks that this router advertises
Restart the **`quagga`** daemon to force it to read the configuration files

Copy this file to each router. If needed change the file permissions so the file is executable, and then run the script.

```
chmod +x routeconfig.sh
./routeconfig.sh
```

The script takes approximately 2 minutes to run. If the script was successful, the final output shows:

```
Starting Quagga daemons (prio:10): zebra ospfd.
Starting Quagga monitor daemon: watchquagga.
```

After it is complete look at the configuration files and notice the interfaces, IP addresses, masks, and subnet numbers have been added.

```
more /etc/quagga/zebra.conf
more /etc/quagga/ospfd.conf
```

Tip: do not run this script on the hosts; only on the routers.

Tip: Because you will need to log into the various nodes multiple times, create aliases to make keyboard shortcuts. The notes section in MP2 describes how to create **bash** aliases and to use **sftp** to copy files.

Monitor the status of the OSPF daemon

Open an ssh window for each router and verity that ospf has found its neighbors and has used Dijkstra's algorithm to build a routing table. Use the interactive terminal **vtysh** to send commands to the ospfd, such as to display status details. You must start the **vtysh** terminal as **root**, and when you are done with a terminal session type **exit** to close the terminal, and **exit** again to end your session as **root**. For example, on **router-1** do

```
sudo -i
vtysh
show ip ospf neighbor
```

You should see a neighbor table similar to this:

```
    Neighbor ID Pri State           Dead Time Address         Interface            RXmtL RqstL DBsmL
10.10.14.2        1 Full/DR           37.976s 10.10.1.2       eth4:10.10.1.1           0     0     0
10.10.13.2        1 Full/DR           33.114s 10.10.2.2       eth5:10.10.2.1           0     0     0
10.10.11.2        1 Full/DR           36.368s 10.10.3.2       eth3:10.10.3.1           0     0     0
10.10.12.2        1 Full/DR           35.470s 10.10.4.2       eth1:10.10.4.1           0     0     0
10.10.16.2        1 Full/DR           36.820s 10.10.5.2       eth6:10.10.5.1           0     0     0
```

In my network, this router has 6 interfaces. Five of them are to other routers, and the sixth to a host. The five routers are listed as neighbors in this table. Each router is identified by a number, and in this case, the router defaulted to using the largest IP address across all of its interfaces as its ID. The state column shows if all of the link-state advertisements (LSA's) have been fully exchanged with each neighbor. For routers on a LAN, one is elected as the designated router (DR). Each of the routers in our example network will be a designated router since we have not included any LANs in our network design. The Dead Time column shows the number of seconds until the link to the neighbor is declared to be dead. The OSPF protocol sends hello packets to each neighbor once each 10 seconds, and will declare a neighbor down if a hello message is not received within 40 seconds. The Interface column shows the interface used for each neighbor.

Next examine the routing table

```
show ip ospf route
```

My example router shows

```
============ OSPF network routing table ============
N    10.10.1.0/24          [10] area: 0.0.0.0
                           directly attached to eth4
N    10.10.2.0/24          [10] area: 0.0.0.0
                           directly attached to eth5
N    10.10.3.0/24          [10] area: 0.0.0.0
                           directly attached to eth3
N    10.10.4.0/24          [10] area: 0.0.0.0
                           directly attached to eth1
N    10.10.5.0/24          [10] area: 0.0.0.0
```

```
                                    directly attached to eth6
N    10.10.6.0/24                   [20] area: 0.0.0.0
                                    via 10.10.5.2, eth6
N    10.10.7.0/24                   [20] area: 0.0.0.0
                                    via 10.10.1.2, eth4
N    10.10.8.0/24                   [20] area: 0.0.0.0
                                    via 10.10.1.2, eth4
                                    via 10.10.2.2, eth5
N    10.10.9.0/24                   [20] area: 0.0.0.0
                                    via 10.10.3.2, eth3
                                    via 10.10.4.2, eth1
N    10.10.10.0/24                  [10] area: 0.0.0.0
                                    directly attached to eth2
N    10.10.11.0/24                  [20] area: 0.0.0.0
                                    via 10.10.3.2, eth3
N    10.10.12.0/24                  [20] area: 0.0.0.0
                                    via 10.10.4.2, eth1
N    10.10.13.0/24                  [20] area: 0.0.0.0
                                    via 10.10.2.2, eth5
N    10.10.14.0/24                  [20] area: 0.0.0.0
                                    via 10.10.1.2, eth4
N    10.10.15.0/24                  [30] area: 0.0.0.0
                                    via 10.10.1.2, eth4
                                    via 10.10.5.2, eth6
N    10.10.16.0/24                  [20] area: 0.0.0.0
                                    via 10.10.5.2, eth6
```

Notice that all of the IP subnetworks that are assigned in your network are listed in the table. The cost of each link defaults to 10. Notice that some destinations (e.g., `10.10.15.0`) have two routes listed, each with the same cost.

Trace the routes in your network

Log into one of your hosts, and use traceroute to find the routes to all other hosts. For example, on one of my hosts

```
$ traceroute 10.10.13.1
traceroute to 10.10.13.1 (10.10.13.1), 30 hops max, 60 byte packets
 1  router-4-link-10 (10.10.11.2)  0.453 ms  0.399 ms  0.362 ms
 2  router-1-link-2 (10.10.3.1)  1.138 ms  1.089 ms  1.041 ms
 3  router-7-link-1 (10.10.2.2)  1.378 ms  1.328 ms  1.220 ms
 4  host-7-link-12 (10.10.13.1)  1.840 ms  1.787 ms  1.718 ms

$ traceroute 10.10.12.1
traceroute to 10.10.12.1 (10.10.12.1), 30 hops max, 60 byte packets
 1  router-4-link-10 (10.10.11.2)  0.472 ms  0.432 ms  0.380 ms
 2  router-6-link-8 (10.10.9.2)  1.245 ms  1.205 ms  1.170 ms
 3  host-6-link-11 (10.10.12.1)  2.312 ms  2.278 ms  2.215 ms

$ traceroute 10.10.15.1
traceroute to 10.10.15.1 (10.10.15.1), 30 hops max, 60 byte packets
 1  router-4-link-10 (10.10.11.2)  0.403 ms  0.378 ms  0.353 ms
 2  router-1-link-2 (10.10.3.1)  1.211 ms  1.172 ms  1.134 ms
 3  router-5-link-0 (10.10.1.2)  1.825 ms router-2-link-4 (10.10.5.2)  1.828 ms
router-5-link-0 (10.10.1.2)  1.743 ms
 4  router-3-link-5 (10.10.6.1)  2.251 ms  2.204 ms  2.176 ms
 5  host-3-link-14 (10.10.15.1)  2.633 ms  2.592 ms  2.525 ms
```

Examining my network topology shows that, unlike with a spanning tree, the route to each destination uses the shortest path. Also, notice the last trace in this example. Router 1 has two equal cost routes to router 3. The multiple packets sent by the traceroute program did not all follow the same route: some were sent via router 5 and others via router 2.

Tips: You can see additional information about how quagga collects entries for the routing table by trying:

```
      show ip route
```

To see additional information about quagga and ospf try these options:

```
      show ip ?
      show ip ospf ?
      show ip ospf database
```

Control packets

Use **`tcpdump`** on one of the routers to monitor control traffic in IP packets.

```
sudo tcpdump -i eth6 ip -vv
```

Responding to changes in the link status

Experiment with bringing a link down and determine the time required for ospf to respond to changes. Log in to two routers that are neighbors of each other, selecting a link for which there is an alternative path. For example, in my network router-1 uses eth5 to router-7. And router-7 uses eth2 to router-1. On router-7, the link to router-1 is taken down. On router-1, monitor the status ospf maintains about its neighbors (including router-7) to see how long it takes to remove the link from its neighbor list.

On router-7, for the interface to router-1:

```
      sudo ip link set dev eth2 down
```

Back on router-1, repeatedly use the **`vtysh`** neighbor command to monitor the dead time, and verify it counts down to zero. (Recall, that you need to use "**`sudo -i`**", and "**`vtysh`**" to open the terminal to the ospf process.)

```
      show ip ospf neighbor
```

Next, log into a host and set up a ping trace that will travel over this link to a remote host. Bring the link up (and wait 40 seconds), then bring the link down again. Are any ping packets lost?

For example, in my trials sometimes I did not lose any ping packets. However, during one trial I noticed this from my ping trace:

```
64 bytes from 10.10.13.1: icmp_seq=9 ttl=61 time=3.22 ms
64 bytes from 10.10.13.1: icmp_seq=11 ttl=60 time=3.84 ms
```

This trial shows one packet was lost (and we can see the **`ttl`** changed indicating the route length increased by one.) Trace route also verified that the route changed. When I brought the link back up, in one trial it took about 5 seconds for ping to show a change in the **`ttl`** (indicating the router switched to the shorter path). In another trial, 10 ping packets where lost when bringing a link back up. I observed that routes seemed to be updated quickly when a link goes down, but the protocol is slower to bring a new link up (and more packets are lost when a new link is added).

Other techniques could be used to further investigate the dynamics of the routing protocol, such as monitoring IP control traffic with **`tcpdump`**, and taking snap shots of the routing table contents with

```
route -n
```

Further Reading

Many additional detail about quagga are found on the web site for the project
http://www.nongnu.org/quagga/

Many of the details for this machine problem are based on a project from the University of North Carolina at Chapel Hill.
http://www.cs.unc.edu/Research/geni/geniEdu/index.html

In particular, the descriptions at the following page provided the template for the configuration script used in our project.
http://www.cs.unc.edu/Research/geni/geniEdu/06-Ospf.html

A overview of the main features of of **`quagga`** and the **`vtysh`** terminal are found here

https://wiki.gentoo.org/wiki/Quagga

Questions

1. Investigate how OPSF is configured.

The primary task of the **`routeconfig.sh`** bash script is to build two configuration files: **`zebra.conf`** and **`ospfd.conf`**. Read the bash script and identify where the **`zebra.conf`** template file is located. Compare it with the **`zebra.conf`** file that the bash script created in the /etc/quagga directory. Notice the file contains a list of the interfaces and the IP address and mask for each interface. Next, compare the **`ospfd.conf`** template to the corresponding file the bash script created in the /etc/quagga directory. Notice a subtle difference: the **`ospfd.conf`** file lists the subnet numbers and subnet masks instead of IP addresses. These are the networks that ospf advertises in the link state packets.

For this question, you are to write a short bash script that prints to standout output the interface name, IP address, subnet number, and mask length. Your script should show these values for all interfaces on a router expect for **`eth0`** and **`lo`**. For example, your output might look like this:

```
eth1 has IP address 10.10.5.1 and connects to subnet 10.10.5.0/24
eth2 has IP address 10.10.4.2 and connects to subnet 10.10.4.0/24
```

```
eth3 has IP address 10.10.1.2 and connects to subnet 10.10.1.0/24
```

You should follow the same approach as in the **`routeconfig.sh`** bash script. Note you can assume that the mask is always 24 bits. However, a nice extension would be to show that any size mask can be supported by your script. For your lab report, simply include a printout of your bash script and an example output when run on any one of your routers.

2. Illustrate your network

Create figures similar to those from MP2 to show your network, IP addresses, and eth interface labels.

3. Link failure and recovery

Identify a link in your network that can be brought up and down without disconnecting your network. Find a pair of hosts that utilize this link in their shortest path route. Experiment with bringing the link up and down. Report on the status of the ospf neighbor table and the dead time status. Use traceroute to show how the route changes after the link is brought down. Use ping traces to determine the approximate time for the routing protocol to find a new route, and how many packets are dropped while the routing tables are updated.

4. Change the cost of a link

The **`vtysh`** terminal allows the parameters of ospf to be modified. For example, examine the settings for the interfaces at the router on the path you identified in the previous experiment.

```
show ip ospf  interface  eth5
```

Notice the field "**`Cost: 10`**". The cost of this link to this specific neighbor can be changed from within the terminal session. Use:

```
configure terminal
```

to enter an interactive session to change parameters at a router. Next, use

```
interface eth5
```

to specify the link, and

```
ip ospf cost 50
```

to change the link cost. Use "**`exit`**" to save changes and exit out of the configuration mode. Verify the status with

```
show ip ospf interface brief
```

Tip: Until I set the link cost the same in both directions on a link, I had connectivity problems.

Use traceroute to show how the change in the cost of a link results in a change to some routing table. Use tcpdump to show that a router that is located far from the change still receives LSP packets.

Lab Report

Your laboratory report must include the following sections:

- A cover page with your name, course information, lab title, and date of submission
- Summary: a summary of the objectives of the lab
- Implementation: a brief description of your implementation
- Results: figures and answers to the given questions
- Conclusion: a brief summary that includes what you learned, difficulties you encountered, and any suggested extensions or improvements.

The report has to be electronic (in PDF format) and must be submitted to Canvas by completing the assignment (submit on the same page that has the PDF for the assignment).

While you are encouraged to discuss the general approach of the problem with others, you are expected to do your own work. **In particular, you must write your own lab report**. **Do not share text, figures, or your rspec file with other students.**

The machine problem is worth 100 points. For each day your submission is late your score will be reduced by an additional 2^i points where $i = 1$ for a submission after the due date any anytime during the next 24 hours, etc. (Note that this rule applies to machine problems only. Late homework sets are not accepted.)

Turning in a PDF is required. Microsoft Word can save as PDF.

Option 2: Create a Multi-hop Wireless Ad Hoc Network

For this machine problem set up a multi-hop wireless ad hoc network for communicating with nearby computers without an existing Wi-Fi LAN. Completion of the project requires a wireless laptop of your own, and two more laptops with the same ad hoc network capabilities. All laptops must be running some version of Linux (Ubuntu or Debian preferred). Hence it is suggested you complete the project with two classmates, friends, or your own extra laptops. If you complete the project with classmates, the group should submit one report with all group members listed as co-authors. The lead author should submit under his or her Canvas account, and notify the instructor by email of the members in the group.

The objective is to build a multiple-hop network using optimized link state routing (OLSR) as the routing protocol. OLSR is a modified version of OSPF designed to reduce the flooding required to distributed link-state packets in a wireless ad hoc network. The most recent version is olsrd(2), and you should experiment with this version.

http://www.olsr.org/mediawiki/index.php/Main_Page

Follow the directions on the above web page, and see the User FAQ page. Before installing any software, begin with the FAQ about troubleshooting. Conduct the "link layer tests" to verify that you can configure your system into an ad hoc network that uses wifi.

Procedure

i) Create a single-hop ad hoc network. Verify the link layer tests and confirm that each laptop can see at least two other laptops. All computers must have same SSID and channel.

ii) Connect to a nearby laptop (neighbor #1) directly (i.e., in one hop) using your created ad hoc network. Use “ping” and “traceroute” to verify that you can communicate with neighbors that are in range.

iii) Create a 2-hop network by having a neighbor #2 that is far enough from you so that you cannot connect with it directly. Let neighbor #1 be between you and neighbor #2, such that you can connect with #1, and #1 can connect with #2. Verify the connection between adjacent neighbors. Next, use your laptop to ping neighbor #2. Verify that the packets are not forwarded to neighbor #2. (Optionally, you can experiment with installing static entries in the routing table to enable forwarding. However, notice that the entries in the routing table are not updated as the topology changes.)

iv) Install the OLSR.org Network Framework (OONF) that includes olsrd2. Once it is installed and running, create the same 2-hop network as iii). From your laptop, “ping” and “traceroute” neighbor #2.

Questions

1. Link configuration for ad hoc mode

Describe the configuration details required to allow your laptop to use wifi in ad hoc networking mode. Use figures or a screen capture to show the steps required to configure the wireless interface

2. Link tests for single-hop network

Capture output from ping and traceroute to show connectivity to neighbors. Optionally, use tcpdump or wireshark to show packet level traces.

3. Multiple-hop network without dynamic routing

Document the experiment in step 3 to show that connectivity is lost when routing in not enabled. If you experiment with static routes, demonstrate a scenario in which the static routes support forwarding. Even if you did not experiment with static routes, describe a scenario is which static routes will fail.

4. OLSR

Describe the process for installing and configuring OLSR.

5. Multiple-hop ad hoc network with mobility

Design an experiment to show that the OLSR protocols can adapt to a change in your topology and estimate how quickly the routes can be updated after a change. For example, measure how many ping packets are lost while the network updates routes as the laptops move.

Lab Report

Your laboratory report must include the following sections:

- A cover page with the name of all group members, course information, lab title, and date of submission
- Summary: a summary of the objectives of the lab
- Implementation: a brief description of your implementation
- Results: figures and answers to the given questions
- Conclusion: a brief summary that includes what you learned, difficulties you encountered, and any suggested extensions or improvements.

The report has to be electronic (in PDF format) and must be submitted to Canvas by completing the assignment (submit on the same page that has the PDF for the assignment).

The machine problem is worth 100 points. For each day your submission is late your score will be reduced by an additional 2^i points where $i = 1$ for a submission after the due date any anytime during the next 24 hours, etc. (Note that this rule applies to machine problems only. Late homework sets are not accepted.)