Final Project : Setting up your own Enterprise Network

The project demo sign up sheets will be posted on Wednesday after class.

All code must be submitted BEFORE the demo presentation on the following dates:
Early Birds submission due date for the code: Monday, May 24th 11:59pm
Regular submission due date for the code: Sunday, May30th 11:59pm

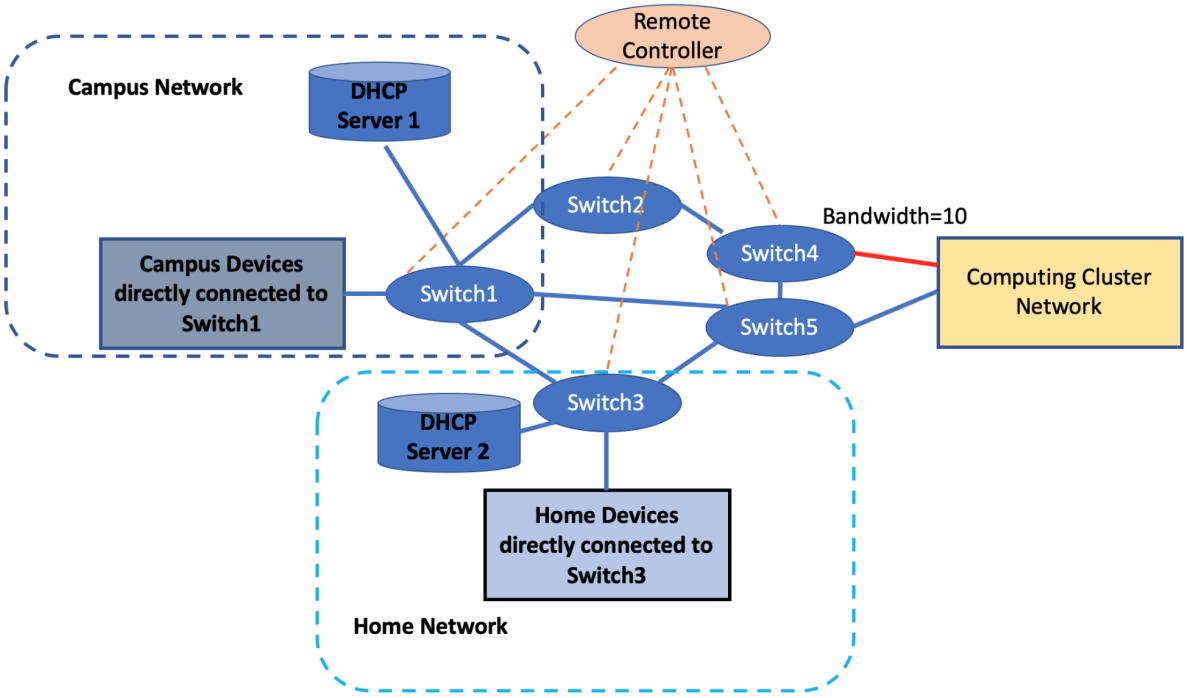
NOTE : No late submissions will be accepted for the final project

Background

In the previous labs you implemented a simple firewall that allowed ARP and ICMP packets, but blocked some TCP packets. For your final project, you will be building your own network with DHCP servers assigning IP addresses to devices, implementing routing forwarding functions between devices on different subnets and expanding on the previous lab's firewall for certain subnets. You will first construct your own enterprise network based on the given constraints. You will be using ideas from Lab 1 to help construct the Mininet topology, and ideas from Prelab3 and Lab 3 to implement the rules allowing for traffic to flow through your network based upon given constraints. Please refer back to the previous Labs for guidance on how to complete this assignment.

Assignment

For this lab, you will set up a small enterprise network. The main idea is to connect a campus network and home network to a computing cluster network. Once the network is set up, your next task is to enable routing rules/policies for the various devices with given constraints in different networks, and in particular enable access from the campus and home networks to the servers that reside on the computing cluster through a proper route path. The topology should be as follows:



Task1 : Creating the Topology

Useful Resources:
https://openflow.stanford.edu/display/ONL/POX+Wiki

You are going to create the designed network given in the previous image, and enable DHCP servers to assign the IPv4 addresses to the devices inside the Campus network and Home networks.

In Mininet, the DHCP server can be implemented with the remote pox controller, dhcpd.py. You can execute the DHCP server as follows:
./pox.py proto.dhcpd --network=x.x.x.0/24 --ip=x.x.x.1 --first=x --last=None….
(reference: https://openflow.stanford.edu/display/ONL/POX+Wiki)

In order to get an IP address assigned, DHCP must be enabled in the client/host that needs an IP address. You can find the example in dhcp-example.py.

Hint: You have to run one DHCP server at a time. For example, when you have DHCP Server 1 running with the specific Hosts/Clients who need IP addresses assigned, you must stop/revoke DHCP Server 1 before running DHCP Server 2.

Campus Network:

In the Campus network, there are 4 hosts, h1, h2, h3, h4.

- Assign **valid IP addresses** to these hosts through DHCP server1, which is implemented by you.
    - IP address constraints: any IPv4 address, with network address 10.1.1.0, and subnet mask /24
- h1, h2, h3, h4 connect to switch1

Home Network:
The Home Network has two connected devices, d1 and d2.

- Assign **valid IP addresses** to these devices through DHCP server2, which is implemented by you.
    - IP address constraints: any IPv4 address, with network address 10.2.2.0, and subnet mask /24
- d1, d2 connect to switch3

Computing Cluster:
The computing cluster has two servers - CCServer1, CCServer2. The Computing Cluster subnet has the network address 10.3.3.0 /29. Choose valid addresses for the servers accordingly.

- Note: these IPv4 addresses should be pre-assigned, as in previous labs, i.e., they are NOT required to be assigned by DHCP server.
- CCServer1 connects to Switch4
- CCServer2 connects to Switch5

Link Parameters:
As you did in previous labs, you must set the link parameters while building the network topology.
Link Parameter settings are as follows:

- The link between Switch 4 and CCServer 1 has bandwidth 10Mbps.
- All other links in the topology have bandwidth 3Mbps.

Testing Task 1: Using a command of your choice, demonstrate how your implementation meets all the requirements of task1. In specific, run a command that can demonstrate that your topology is set up as required. Add tests to Readme.

Controllers for Task2 and Task 3:

For Task2 and Task 3 you will use the POX Controller **finalcontroller_skel.py** for routing and the firewall implementation. To assign IP Addresses to hosts in the network, you can choose to either use the DHCP Controller as you did in Task 1 (DHCP server1 and DHCP server2) or you can just apply the IP addresses assigned through Task1 by directly incorporating these addresses into your **final_skel.py**

Task2: Routing Function

In this part, you are going to implement the routing forwarding functions between devices in different subnets based on some constraints.

Based on the topology you built in Task 1, there might be multiple paths between the hosts(clients or servers) in different networks. For instance, the host1 in the Campus Network needs to communicate with Servers in the Computing Cluster Network. Host1 can reach CCServer1 over multiple ways/paths.

Implement the flow table to achieve the following routing forwarding functions on switches:

1) Clients in the campus network communicate with the devices in the Home Network through the shortest path (Hint: through the least number/hops through switches from the source to destination) NOTE: if there is more than one shortest path, either one can be selected.

2) Clients in the campus network communicate with one of the servers in the Computing Cluster Network through the shortest best performance links (*e.g.*, higher bandwidth link), which means, finding the best performance links first, then selecting the shortest link among the best performance links. NOTE: if there are more than one shortest path, either one can be selected.
    a) Which server, CCServer 1 or CCServer 2, will be selected based on the shortest best performance link constraints?

Testing Task 2:

Using commands of your choice, demonstrate how your implementation meets all the requirements of task2. Add the command(s) used and explanation to your README file.

NOTE:

Before you run any commands in mininet>, to verify Task2 and Task3, we need to bring down two links by using the following two commands:

For example:
mininet> link s4 s5 down
mininet> link s1 s5 down
mininet> h1 ping ccs1
mininet> all other test commands

If you restart your topology, you have to run these two commands again.

Task3: Firewall

Device 1 (d1) in the Home network has some malicious behavior detected. Implement a firewall to completely isolate Device 1 (d1) from the network.. In addition, block all IP traffic from d2 as well.

Testing Task 3:
Using commands of your choice, demonstrate how your implementation meets all the requirements of Task3. Add the command(s) used and explanation to your README file.
NOTE:
Before you run any commands in mininet>, to verify Task2 and Task3, we need to bring down two links by using the following two commands:
For example:
mininet> link s4 s5 down
mininet> link s1 s5 down
mininet> h1 ping ccs1
mininet> all other test commands

Note:

- You may do the routing and firewall implementation however you choose -- however, you may find it easiest to determine the correct destination switch port by using the destination IP address and source IP address, as well as the source port on the switch that the packet originated from. Additional information has been given to you in the do_final() function to allow you to make these decisions. Please see the comments in the provided code for guidance.

- Based on your personal topology, the IP addresses of hosts and devices might vary.

Provided Code :

Final_skel.py
finalcontroller_skel.py
Dhcp-example.py

We have provided you with starter code (skeletal files) to get you started on this assignment.

- The dhcp-example.py shows an example of how to get IP addresses assigned through a DHCP server. You have to make your own final-dhcp.py to satisfy Task 1. Note: You don't have to change the dhcpd.py in proto.
- The controller file (finalcontroller_skel.py) serves as a skeletal for implementing task2 and task3.
- The controller file for task2 and task 3 needs to be placed in ~/pox/pox/misc and mininet file (final_skel.py) needs to be placed in your home directory (~).

You need to modify all three files to meet the lab requirements.

You will be using slightly different commands to create the Hosts and Links in the Mininet file to give you more information to make decisions within the Controller file. Additionally, you will notice that you have additional information provided in the do_final function. This is documented in the comments within the files.

Summary of Goals:

1) Creating a Mininet Topology to represent the above topology and assign IP Addresses to hosts through the DHCP servers.
2) Implement routing, considering shortest path and link bandwidth, using the Pox Controller.
3) For the Firewall, all hosts are able to communicate EXCEPT :
    a) d1 cannot send or receive any traffic at all
    b) All IP traffic to and from d2 will be blocked

Testing:

You may test with Mininet commands and observe packets with Wireshark inside your VM. We will not be telling you what commands to run to verify the assignment goals are met in this document. Figuring out how to prove your work is a part of this assignment. Please test your code comprehensively before coming to the demo sessions.

Grading Rubric:

Total: 100 points

25 points: Mininet Topology (tested with Mininet commands in the demo)
5: Devices are successfully created**. Please name your hosts and servers using the names specified in the topology, and in specifications listed. Do not assign different names.**
10: Links are successfully created, and the topology is correct.
10: IP addresses are assigned correctly (using DHCP and based on constraints)

15 Points: Commands for Testing Task1, Task2 and Task3 in README file.

40 points: POX Controller for routing and firewall implementation

20 points: Quality of your demo presentation with the TAs.

You will need to attend the demo sessions and explain your code and results to the TAs/Tutors. Credit is given based on the clarity of your explanation and justification of results in the demo presentation. Before attending the demo sessions, **you have to submit your code files on Canvas according to the due date**. Partial credit may be awarded for incomplete assignments based upon the submitted code and explanations in the demo as to why something may not be functioning properly. If you are not able to get the expected results, you could explain what you think is going on (for partial credit).

10 extra points for Early Birds: You can choose to submit your code and present the demo earlier than the project due date to earn **10 extra points** for the project. To do that, you need to commit your code on Canvas **by 11:59:59 pm on May 24th** and sign up the early bird demo in **Week 9**.

Deliverables on Canvas:

1. **final_dhcp.py:** you topology code with dhcp for task1
2. **final_skel.py**: Your topology code for take2 and task3.
3. **finalcontroller_skel.py**: Your controller code for task2 and task3.
4. **README.txt**: A readme file explaining your submission and also the commands used to verify and test your implementations (Task1, Task2 and Task3).