CSci 4211: Introduction to Computer Networks
Programming Assignment IV

Due Date: Friday May 4, 2018 at 11:55 p.m

Handout: April 18, 2018

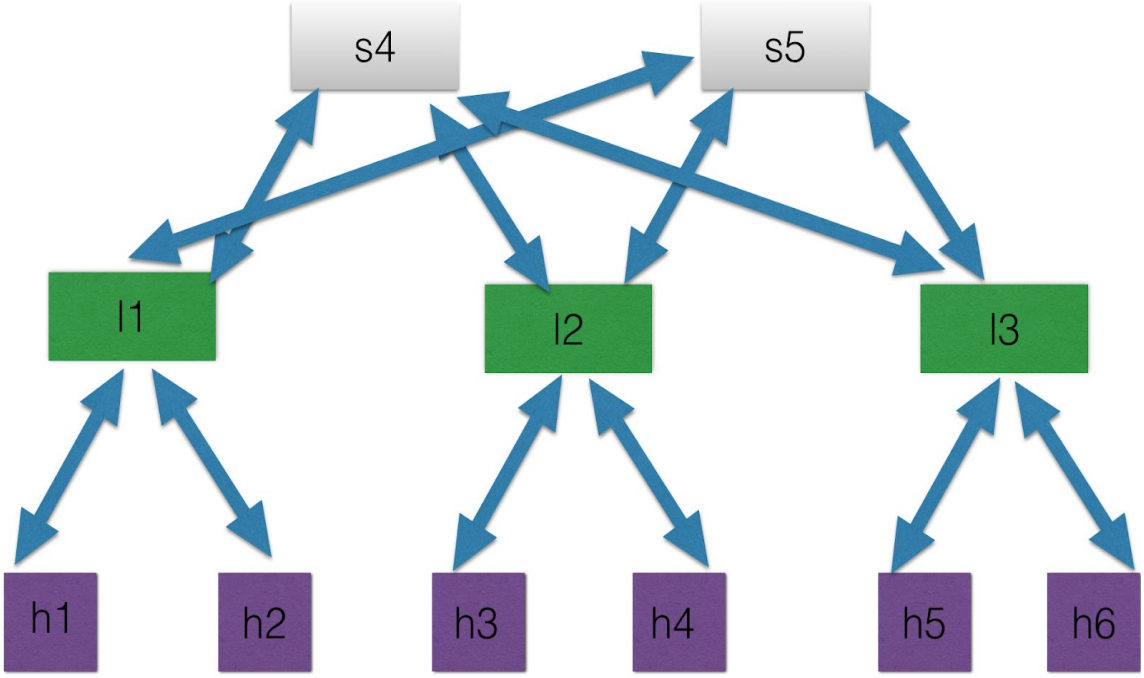
For the fourth programming project, you will be simulating a small data-center network and control it by using the SDN paradigm. Here, using mininet and the techniques you learned in the third project, you will created a network, design an ARP responder, install the rules to make traffic flow and deal with link failures which are a common occurrence in real life networks.

This project can be done in upto groups of three. You can keep your groups from project three if you like, but do mention your groups in moodle.

Leaf-Spine Topology:

Leaf-Spine architecture is widely adopted in modern data centers. Leaf-spine is a two-layer network topology composed of leaf and spine switches. Hosts are connected to leaf switches and leaf switches are connected to spine switches. Leaf switches through the spine that form the access layer that delivers network connection points for the hosts throughout the data center. Spine switches have high port density and bandwidth available and form the core of the architecture.

For this project you will also use a leaf spine topology (which is given to you, please see the topology.py file). The topology can also be seen in the figure below



Leaf Spine Topology

Here switches s4 and s5 are the spine switches, switches l1,l2 and l3 are the leaf switches and h1-h6 are the hosts connected in this topology.

For sending traffic between the hosts, please follow the following rules:

- If both hosts are connected to the same leaf switch, then the traffic between the hosts will only traverse through that leaf switch. For example, the path between h1 and h2 is h1-l1-h2.
- If the two hosts are connected to different leaf switches, then the traffic between them need to go through a spine switch. The policy of choosing the spine switch is: traffic source from h1, h3 and h5 should traverse through spine switch s4 and traffic source from h2, h4 and h6 should traverse through spine switch s5. For example, the path between h1 and h5 is h1-l1-s4-l3-h5, while the path between h2 and h5 is h2-l1-s5-l3-h5.

The project consists of three parts which are described as follows. The controller program you need to implement should consist of these three parts.

Part 1 ARP Responder (30 points)

For traffic delivery, MAC addresses are used by the network, but each host on the network initially does not know the other host's MAC address. Here the ARP responder comes into play. It keeps track of all the IP-to-MAC mapping and responds to the ARP queries sent from the hosts. The mapping can be known beforehand and stored in the controller for responding to the queries.

You need to implement the ARP responder as one component in your controller program, and you can not use any built in mechanism in the controller for arp responders.

SInce the arp responder resides at the controller, rules should be installed at the switches to forward all ARP requests directly to the controller. The controller creates ARP replies and then sends the ARP replies back to the switches. The switches further deliver the ARP replies to the hosts.

For checking whether the arp responder is working correctly, use the command "<host> arp -n" (e.g., h1 arp -n). It prints the arp table for each host.

Part 2 Installing Rules (30 points)

Since the topology and the link details are known beforehand, you can figure out the rules that will be required for forwarding traffic in the network.In this project you will need to install rules in the switches so that the switches can know where to forward packets.

In this routing component, due to the static topology and the initial installation of rules, after the topology has been bootstrapped, there should be negligible traffic between the controller and the switches about traffic handling.

The leaf spine topology contains forwarding loops, flooding packets will cause a broadcast storm and therefore it should be avoided at all costs. Make sure that the rules are working properly.

For checking that the rules are working, make sure that all hosts are reachable by other hosts. For example, use pingall.

Part 3 Link Failures (40 points)

In data centers, link failure is a common occurrence, due to the fact that there exist a large number of links and the links are not 100% reliable. Mitigating link failures then becomes one of the most important management tasks for operators. To do that, one approach is to add a certain degree of redundancy to the data center architecture, and design a proper mechanism to reroute the traffic through “good” links if link failure happens. In this part, you will simulate link failures and implement mechanism to deal with them.

For simulating the failure of a link, please use the “link down” command of mininet in the CLI.

For completing this part you have to do the following two things:

1. Detect Link Failure:

You should be able to detect that a link has failed in the topology. You can use built-in mechanism of the controller for this part.

2. Mitigating Link Failure:

After detection of failures, you should mitigate the effect of such failures by installing rules at the appropriate switches to bypass the failed link so that end-to-end connectivity remains.

In this part, you should fail all three links of any one spine switch and check whether pingall works. Please note that only one spine switch's links should be affected, and all its links will be disconnected.

We assume that even in the event of link failure there is always a physical path available between the hosts. Therefore if a subset of each spine switch links are failed such that a physical path is available, even in that case your solution should route traffic. For example if l1-s4, l2-s4 and l3-s5 are all failed even then traffic should flow.

Notes:

- The above three parts are different components of your controller and have to run side by side. So make sure that there is no error when all three components are run concurrently.

- For obtaining the details about link connection on the controller, you can use the fact that mininet assigns ports sequentially to switches, hence the details of links will always be the same and the details can be seen in the mininet CLI using the command “links”.
- The mac address that mininet assigns to the hosts are also sequential, so h1 gets the mac 00:00:00:00:00:01, h2 gets the mac 00:00:00:00:00:02 and so on. Make sure to use the option “--mac” for this behaviour to exist.
- The IP address for the hosts are also sequentials and the details can be found using the command “ifconfig”.
- For this project you will use the leaf spine topology that has been provided to you. Please do not make any changes to the topology file.

  For running the topology while using POX as the controller, use the following command

  ```
  sudo mn --controller=remote --custom topology.py --topo mytopo --mac
  ```

  If you are using Floodlight as the controller, then you the following command

  ```
  sudo mn --controller=remote,ip=127.0.0.1,port=6653 --custom topology.py --topo mytopo --mac
  ```

Submission:

1) All the code files for controller.
2) Readme for describing the following things
   a) Code implementation and design details
   b) Any relevant details for running the project
   c) A detailed breakdown of individual contributions
3) A session file to show the output for the following three things
   a) The arp table of each host after the pingall (you can use the command h1 arp -n for the arp table of h1).
   b) Pingall result with the initial topology
   c) Pingall result with one spine switch (s4) link failures (all three links should be failed).