Instructions

Due: 12/9/16 11:59PM

Complete the following assignment in pairs, or groups of three. Submit your work into the Dropbox on D2L into the "Programming Assignment 5" folder. All partners will submit the same solution and we will only grade one solution for each group.

Learning Objectives

In this lab you will:

- Implement priority-based forwarding on routers
- Implement MPLS forwarding on routers
- Control forwarding paths using MPLS labels

Overview

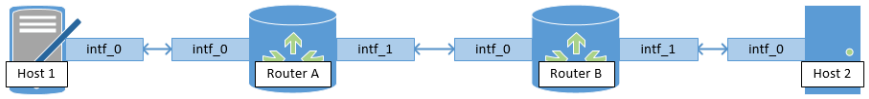
In this project, you will implement priority-based forwarding and MPLS forwarding at routers. In this assignment you will also have a greater autonomy over and responsibility for the design of your protocol based on requirements.

Starting Code

The starting code for this project is available on GitHub at `https://github.com/mwittie/CSCI_466/` **in branch prog5**. Please clone the project code by following these instructions: `https://help.github.com/articles/cloning-a-repository/`. The code provides you with the implementation several network layers that cooperate to provide end-to-end communication.

```
NETWORK LAYER (network.py)
DATA LINK LAYER (link.py)
```

The code also includes `simulation.py` that manages the threads running the different network objects. Currently, `simulation.py` defines the following network.



At a high level a network defined in `simulation.py` includes hosts, routers and links. `Hosts` generate and receive traffic. `Routers` forward traffic from one `Interface` to another based on routing tables that you will implement. `Links` connect network interfaces of routers and hosts. Finally, the `LinkLayer` forwards traffic along links.

The major change between this and your previous assignment is that router interfaces are now configured with capacities. Consequently the transmission time at the link layer depends on those capacities and the length of the packet. I will go over the details of the code changes to the code in class. If you are curious to see exactly what has changes compare the code for this and previous assignment using Beyond Compare (`http://www.scootersoftware.com/download.php`), or a 'diff' tool of your choice.

Program Invocation

To run the starting code you may execute:

```
python simulation.py
```

The current `simulation_time` in `simulation.py` is **10 seconds** to account for non-zero transmission times. As the network becomes more complex and takes longer to execute, you may need to extend the simulation to allow all the packets to be transferred.

Assignment

1. [10 points] When running the code you will notice a bottleneck exists on the outgoing interface 0 of Router B. The output shows you the remaining queue size after a packet is transmitted over a link. Your first task is to implement strict priority forwarding to make sure that higher priority packets skip over lower priority packets in the outgoing queue.
   (a) [3 points] The `udt_send()` function in `simulation.py` sends packet with priorities 0 and 1. Assume higher number priorities are higher priorities, i.e. 1 is higher than 0. Extend the `NetworkPacket` to carry the priority number with which it was sent.
   (b) [3 points] At each router implement strict priority forwarding. My suggestion would be to change the implementation of `Interface.get()`, but other approaches are possible as well.
   (c) [3 points] Modify the output to show that packets with priority 1 are forwarded first if the interface outgoing queue has packets of mixed priorities. Specifically the current output shows the queue size of an outgoing interface as:

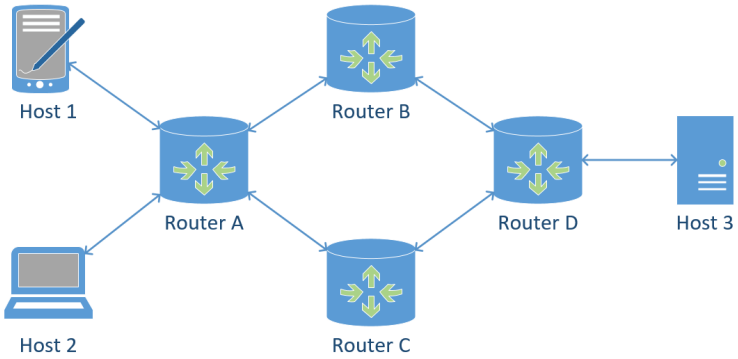
   ```
   Link Host_1-0 - Router_A-0: transmitting packet ...
    - seconds until the next available time 0.416000
    - queue size 3
   ```

   Modify that output to show the number of packets queued at each priority level as:

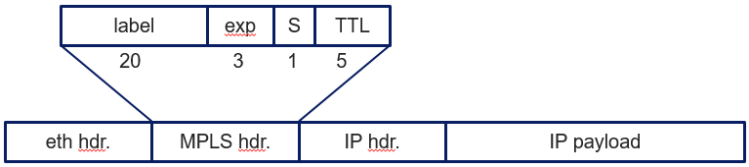
   ```
   Link Host_1-0 - Router_A-0: transmitting packet ...
    - seconds until the next available time 0.416000
    - queue size 3: priority 0: X packets, priority 1: Y packets
   ```

   (d) [1 point] Make sure that output at the host shows what priority packet has been received. Hint: you should be able to get this 'for free' by correctly modifying `NetworkPacket` to carry the priority. Submit `link_1.py`, `network_1.py`, and `simulation_1.py`.

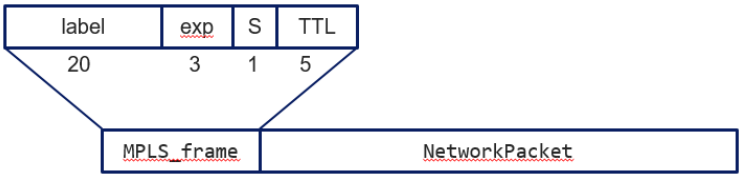
2. [10 points] Instead of forwarding packets of all priorities to the destination on the same path, your next task is to implement MPLS forwarding, such that packets from different hosts follow different paths. In this part use the following topology, with both Host 1 and Host 2 sending packets to Host 3.



(a) [3 points] Implement `MPLS_frame` class to encapsulate `NetworkPackets`. When a `NetworkPacket` arrives from a host at a router, the router should encapsulate the packet in an MPLS frame. In the slides we presented the MPLS frame structure and position with respect to link (Ethernet) and network (IP) layer packets as:



In the context of this project, where we have no link-layer frame and we do have a `NetworkPacket`, your transmissions should look like:



The MPLS frame should contain the label at the least. You may add experimental bits, S bit, and time to live if you choose. The values of these fields will depend on `NetworkPacket` priority and network topology.

(b) [3 points] Modify `Router()` constructor to accept MPLS forwarding tables. The tables should contain the `in_label`, `in_intf`, `out_label`, and `out_intf`. Pass in correctly designed forwarding tables so that your routers achieve end-to-end connectivity and forward traffic from different hosts on different paths.

(c) [2 points] Modify `Router.forward_packet()` function to encapsulate and decapsulate packets and to forward MPLS frames. Only links incident on hosts should carry `NetworkPackets`. Other links should only carry `MPLS_frames`.

(d) [2 points] Implement priority forwarding on your MPLS routers. Remember that priority is originally carried in the `NetworkPacket`. You will need to devise a method for your routers to somehow have access to packet priority (there are three good alternative solutions for implementing this).

Submit `link_2.py`, `network_2.py`, and `simulation_2.py`.

3. [1 point] BONUS: Implement Weighted fair queuing (WFQ) instead of strict priority in question 1. Submit `link_3.py`, `network_3.py`, and `simulation_3.py`.

4. [1 point] BONUS: Implement a central controller to automatically configure MPLS forwarding tables in question 2 based on a global knowledge of network topology. Submit `link_4.py`, `network_4.py`, and `simulation_4.py`.