Instructions

Due: 10/24/16 11:59PM

Complete the following assignment in pairs, or groups of three. Submit your work into the Dropbox on D2L into the "Programming Assignment 3" folder. Both partners will submit the same solution and we will only grade one solution for each group.

Learning Objectives

In this lab you will:

- Packetize streams at the network layer
- Implement packet segmentation
- Implement forwarding through routing tables

Overview

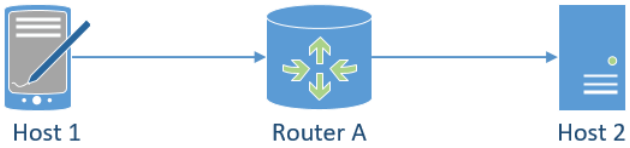
During this project, you will implement several key data plane functions of a router, including stream packetization, packet segmentation, and forwarding. The next assignment will complement these functions at the control plane.

Starting Code

The starting code for this project (`prog3.zip` in the D2L content area) 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. Please consult the video lecture for a more in-depth explanation of the code.

Program Invocation

To run the starting code you may execute:

```
python simulation.py
```

The current `simulation_time` in `simulation.py` is one second. 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 transfered.

Assignment

Your task is to extend the given code to implement several data link router functions.

1. [2 points] Currently `simulation.py` is configured to send three very short messages. Instead, generate a message for `Host_2` that's at least 80 characters long. You will notice that this messages is to large for the link MTUs. Your first task is to break this message up into two separate packets.

   Submit `link_1.py`, `network_1.py`, and `simulation_1.py`.

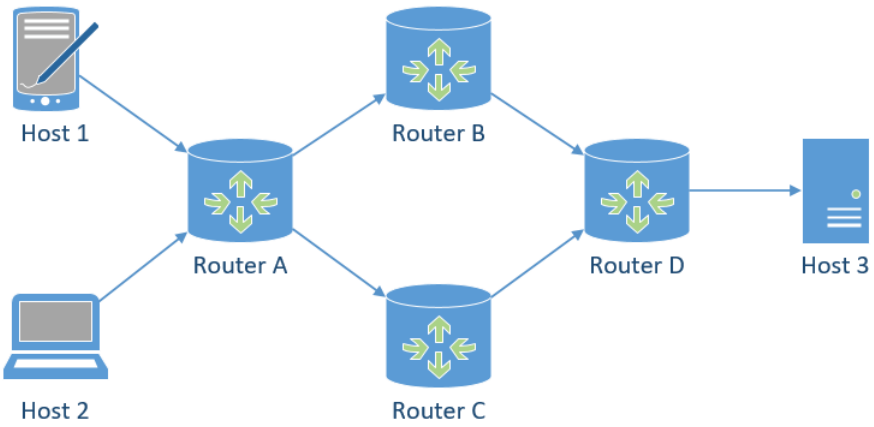
2. [10 points] The packets you created are small enough to pass over the links. However, if we change the MTU of the second link (between `Router_A` and `Host_2`) to 30, the packets will now need to be segmented.

   Your task is to extend the network layer to support segmentation. Study lecture notes and the book on how IP implements segmentation. Extend the classes (including packet format) in `network.py` to match IP's mechanisms for packet segmentation and reconstruction.

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

3. [13 points] The current router implementation supports very simple forwarding. The router has only one input and one output interface and just forwards packets from one to the other. Your tasks is to implement forwarding of packets within routers based routing tables.

   First configure `simulation.py` to reflect the following network topology.



   Second, create routing tables so that both Host 1 and Host 2 can send packets to Host 3 (whose address is 3). The routing table for each router should be passed into the `Router` constructor, and so should be defined in `simulation.py`. The format of these is up to you. You will also need to modify the `Router` class to forward the packets correctly between interfaces according to your routing tables.

   Finally, third extend `NetworkPacket` with a source address and forward packets from Host 1 over Router B and from Host 2 over Router C.

   Submit `link_3.py`, `network_3.py`, and `simulation_3.py`.

4. [1 point] BONUS: The communication in the network above is one directional. Extend the network to carry packets both ways and have Host 3 send acknowledgements to Hosts 1 and 2.

   Submit `link_4.py`, `network_4.py`, and `simulation_4.py`.