EE 367L Lab 6

You are a new employee at a small start up company called “virtualstartup” which specializes in building virtual stuff. One of your co-workers is building a virtual network.

1 Network description

A network consists of *nodes* and *links*. Figure 1 shows a simple network. For this project, nodes are either *hosts* or *switches*. Nodes are supposed to be computers or computer equipment, e.g., a laptop or desktop could be a host. A switch is an ethernet switch or an IP router. Hosts are the sources and destinations of communication, while switches just relay the communication. Nodes are connected to each other with links.

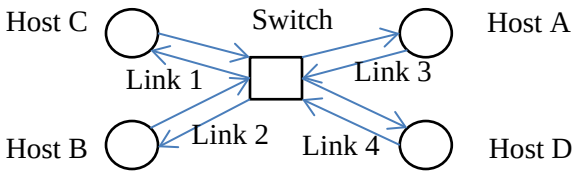


Figure 1. Simple network.

There are two types of links, *unidirectional* (or *directed*) and *bidirectional* (or *undirected*). In unidirectional links, data flows from one end to the other. Bidirectional links are composed of two unidirectional links in opposite directions. The communication in bidirectional links flows in both directions. We only consider bidirectional links, which will be referred to more simply as links.

As shown in Figure 1, hosts are at the edges of the network and the switch provides the connectivity between the hosts. Note that hosts have exactly one incident bidirectional link, while the switch can have more than one. Again, hosts are the nodes that send and receive data, while the switch just relays the data.

2. Implementation of the simulator

Each network node is implemented by a process, and links are implemented with pipes. Your co-worker has just started working on it and has built the smaller shown in Figure 2. It has two hosts and no switches. The hosts are directly connected to each other. To control the hosts, there is a system manager software that can connect to either host, but only one at a time. The manager accepts commands from a user via a console, and then executes the commands.

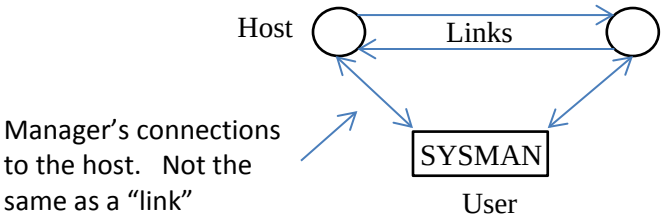


Figure 2. Diagram of project.

You just found out that your co-worker found a new job and will be leaving TODAY! Your boss wants you, the new employee, to take over and build up this project. So you scramble to find the source code, and any documentation before your co-worker leaves for another country. You get the source code which is half baked, with crudy documentation. Your co-worker tells you that it works for the most part but "it's a little buggy". There is a nice farewell party for your co-worker and then the day ends, and you are left with the project. Life isn't fair.

This is what you found out.

2.1 Links

The links can transport a stream of bytes and will be implemented by pipes (e.g., pipe or pipe2). It can transport a packet which is essentially a file with a:

- Header: Control information to process the packet. It is comprised of
    - Destination address (2 bytes)
    - Source address (2 bytes)
    - Length (1 byte), which is an unsigned integer
- Payload: The data that is carried by the packet (size ranges from 1 and 200 bytes)

The addresses are called *network addresses* since they are used to get the packet through the network. One particular address 1000 is reserved for *broadcast*, which means this packet is meant for all hosts. No node should have this network address.

2.2 Host

Each host will have its own ID, which we refer to as its *physical ID*. This is fixed. Initially, this is its network address, but it can be changed by the system manager.

The host will be able to send and receive packets through the link. The host has the following:

- A main directory, which it can access files. This must be a subdirectory in the directory of the network simulator.
- Send packet buffer, which is a buffer that holds a packet to be sent on the outgoing link. More about this below.
- Receive packet buffer, which is buffer that holds a packet that is received on the incoming link. More about this below.

The following are the state values of a host.

- physical ID: Physical ID, which is unique among nodes. It is assigned at the start of the program, and doesn't change thereafter. Implemented as an integer data type.
- maindir: Name of its main directory. This can be changed, and initially, it's blank. This is implemented as a character string.
- netaddr: The network address for the host. This can be changed. Initially, it's blank. Implemented as an integer data type.

- nbraddr: This is the address of the node the host is connected to, i.e., it's neighbor node. Initially, the host will not know it's neighbor's address. This state is initialized to 0xffff to indicate that the neighbor is unknown. The host then will learn its neighbor's address whenever it receives a packet and checks the source address field. This is implemented as an integer.
- rcvflag: This flag indicates if a new packet has been received. It is set to 1 whenever a new packet is received. A user may clear this flag to 0.

The host can send and receive packets. When it receives a packet, it stores it in a *receive packet buffer*. The receive buffer should be a struct which has members for the source address, destination address, length, and payload.

Whenever a packet is received

- If its destination address is the same as the host's network address then flags are set to indicate that a new packet has been downloaded into the receive packet buffer
- Otherwise, the packet is discarded

When a host sends a packet, it comes from its *send packet buffer*. The send packet buffer should have enough space for the payload as well as for other things like source address, destination address, and length. But that's an implementation decision.

Whenever a host sends a packet (from the send packet buffer) on the outgoing link, the source address of the packet is the host's address.

Besides sending and receiving packets, a host can

- Upload a file (from its main directory) into the payload of the send packet buffer.
- Download the payload from the receive packet buffer into a file in its main directory.
- Display the state of the host
- Change the state of the host

SYSMAN

This is connected to each host through pipes as shown in Figure 2. It allows a user to send commands and receives replies to one host at a time. The host that the SYSMAN is currently connected to is the *active h*ost, and the connection is the *active connection*. The following is what the user can do with the SYSMAN through the console:

- List all hosts (physical IDs) with an indicator of active connection
- Select host (physical ID) to be an active connection
- Display state of host: physical ID, main directory, network address, neighbor address, receive flag.
- Set host's main directory
- Set host's network address
- Clear host's receive packet buffer
- Download the payload of the host's receive packet buffer into a file in the main directory. The user specifies the file's name. The download overwrites any existing file.
- Upload a file in the main directory into the send packet buffer's payload. The user specifies the file's name. If the file is too big for a packet (i.e., > 200 bytes) then the file is not uploaded and the user gets an error message.

- Send packet in the send packet buffer.

Instructions

Your boss asks you to make three improvements to your co-worker's software.

Improvement 1:

The network must be able to simulate switching nodes. Switching nodes have multiple incoming and outgoing links. An example is the central node in Figure 1. They are not connected to the manager.

They have a queue for incoming packets.

There is a table called the forwarding table. The table looks like Table 1.

Table 1. Forwarding Table.

Valid	Destination Network Address	Outgoing Link #
1	A	3
1	B	2
1	C	1
0		
1	D	4

The valid column indicates whether the entry is valid or empty. The destination network address column is the addresses of all known host nodes to the switch node. The outgoing link # is the link that you should transmit the packet to get to its destination. Table 1 shows the entries for Figure 1.

When a packet arrives and it finds its destination in the table, it is transmitted on the appropriate outgoing link. If the packet does not find its destination then it is transmitted on all outgoing links except the one it arrived on. For example, if a packet arrived on link 3 then and it is transmitted on links 1, 2, and 4 but not on 3.

The switching node updates the table as packets arrive. When a packet arrives, the switching node examines the source address A and the link it arrived on, say link L. If A is in the table, then L will overwrite the outgoing link # entry for A in the table. If A is not in the table, then a new entry will be made with A as the destination address and L as the outgoing link #. For example, suppose a packet that with source address B and destination address C arrives in a switching node, and the node's forwarding table is as shown in Table 1. Also, suppose the packet arrives on link 4. Then the forwarding table is updated, so that in the second row with destination B, the outgoing link changes from 2 to 4. Of course, the packet is forwarded on link 1.

The switching node has a packet queue. As packets arrive, they are stored in the queue. When the switching node is ready to transmit a packet, it takes one from the head of the queue.

The switching node will run an infinite loop. Each pass through the loop, it first examines all incoming links for an arriving packets and puts them in its packet queue. As it processes each packet, it updates the forwarding

table. Next, it transmits *one* packet from its packet queue (if the packet queue is empty then it transmits nothing). Then it goes to sleep for 10 milliseconds.

Improvement 2: The currently the host can only transport a file of 200 bytes or less. You must upgrade this so files of 2000 bytes or less can be transported. Do this by developing a simple file transport mechanism using the packet service.

Here's a suggestion. When the manager commands a host to transport a file, first download the file in a buffer (a new buffer) in the host. Then break the file up into smaller pieces and put them into the following "container":

- The container is at most 200 bytes
- It has a type-length-value (TLV) format
    - The first byte is the type
    - The second byte is the length of the value in bytes.
    - The value is the payload. Note that this can be at most 198 bytes.

The type byte can indicate whether this is the last packet of the file. For example, type = 0 means there are more packets for the file; and type = 1 means this is the last packet in the file.

The host will transmit one packet between its sleeping periods, so approximately one packet per 10 milliseconds.

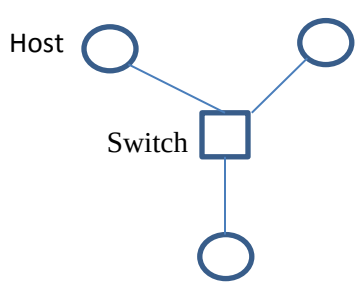
The manager has to be modified so that it can command the host to transport a file. The host will reply to the manager once the file has been completely transmitted from its packet queue.

Improvement 3: With improvements 1 and 2, the network should work for more general topologies, and in particular tree topologies. The simulator, when it starts up, should read in a data file which describes the topology. In particular, it must describe the number of host nodes, the number of switching nodes, and the links. In addition, it must specify the end nodes of the links.

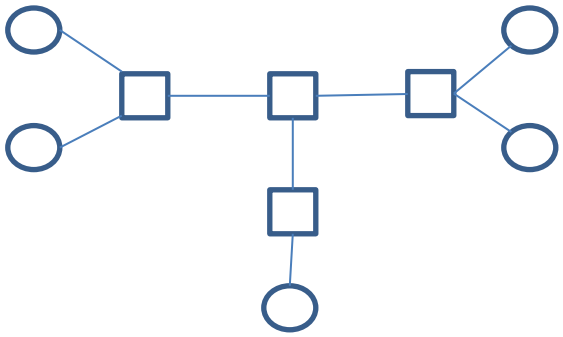
Using this data, it creates the links, connections, and creates the node processes.

Submission Instructions

Your program should run on the following two topologies. The first has three hosts and a switch, and the second has four switches and five hosts. You should a file that describes each topology and your program should read the file as part of its initialization and create the necessary links, hosts, and switches.



Star Topology



Tree Topology

Your program should transfer files between hosts. A sample file is attached in a directory. Demonstrate to the TA for both topologies.

Submit in a tarballed and gzipped directory the following items:

- Source code
- makefile
- README file that explains how to compile and run your program. It should also include your group members
- Two topology files: star and tree topologies shown above
- Directories for the hosts to download and upload files. Label these subdirectories Testdir1, Testdir2,..., Testdir2
- A datafile to transfer over the network – you can use the one given to you by the instructor