EE 367L Switched Local Area Network (LAN): Part 1 Spring 2023

Outline

1 Background

You are are part of a group of new employees at a small start-up company called “iLUVEE367” which specializes in building virtual stuff. A software developer (a co-worker) is building a virtual communication network as shown in Figure 1.

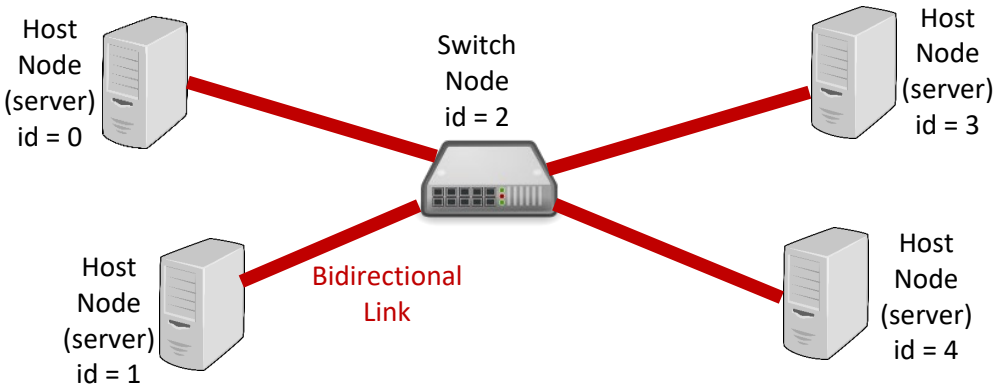

Figure 1. Example network.

The network is comprised of two types of nodes: hosts and switches. Nodes are connected by bidirectional links.

- Hosts are nodes that have data (e.g., files) to transport across the network. They have access to directories of files. A host has only one incident link, which is connected to a switch.
- Switches are nodes that just relay data.

The nodes have distinct integer identifiers (ids), that that range from 0 to 127.

Figure 2 shows a bidirectional link connecting two nodes. Links transport streams of bytes through *ports*.

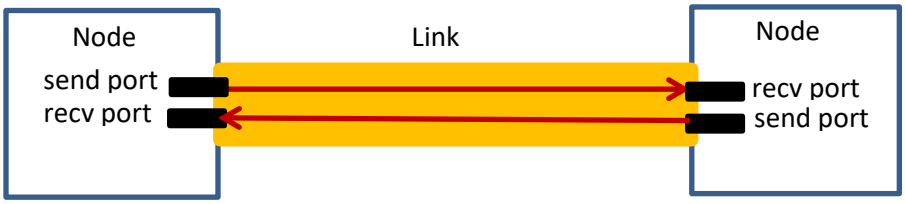

Figure 2. A bidirectional link realized by two unidirectional links

There is a *manager system* (software), that allows a user to control a host. This is shown in Figure 3. The system will connect the user to one host at a time. It allows the user to change the connection to a different host. The user can send commands to the connected host.

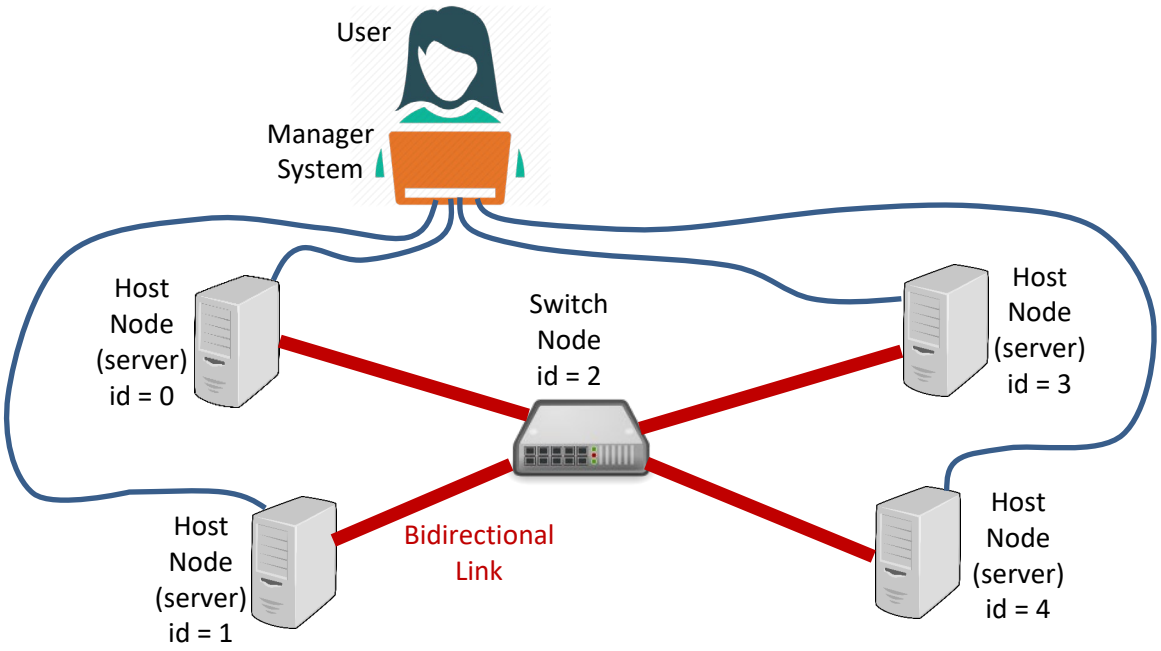

Figure 3. Manager system.

Example operations of the manager system are as follows (here, "current host" is the host the manager system is currently connected to):

- List all host nodes
- Change the current host.
- Set the directory of the host
- Display the state of the host
- Transfer a file from the current host to another host.
- Ping another host, i.e., check if the other host is available through the network.
- Quit

The simulator runs as a multi-process system, where the manager system is the parent process, and the nodes are child processes. The links between nodes and between the manager system and hosts are implemented as pipes.

You learned that your co-worker found a new job in South America, and is leaving TODAY! And the farewell party was great!

Your boss tells your group to take over the project. Your group scrambles to find the source code and documentation from your co-worker. The source code is half-baked and sketchy with minimal comments. It's also "a little buggy". The documentation is virtually non-existent.

Life isn't fair. 

Currently, the software cannot implement switch nodes. In addition, it has limited file transfer capability.

Here are the upgrades you are expected to accomplish:

- Improvement 1: Implement switch nodes
- Improvement 2: Improve file transfer capabilities
- Improvement 3: Allow links to be implemented as sockets as well as pipes.

2 Network

As mentioned earlier, the network has two types of nodes: host and switches. Nodes have distinct identifiers (ids).

2.1 Packets

The network nodes exchange information in the form of *packets*. A packet is a file with the following format:

Field	Size in bytes	Description
Source	1	The host id of where the packet originated (sourced)
Destination	1	The host id of where the packet is destined
Type	1	The type of packet
Length	1	Length of the payload. The size of the payload can range from 0 to 100
Payload	Ranges from 0 to 100	The data that is being transported

Thus, the packet has total length between 4 and 104 bytes.

2.2 Network Configuration

A simple network configuration is shown in Figure 1, where there is a single switch node connected to multiple host nodes. A *configuration file* is one that specifies a network configuration. The following is a configuration file for the network in Figure 1.

```
5
H  0
H  1
S  2
H  3
H  4
4
P  0  2
P  1  2
P  2  3
P  2  4
```

The first line is the number of nodes, which is 5. Each of the next 5 lines is specifies a node. For example, "H 1" means a host node that has id 1, while "S 3" means a switch node with id 3. The subsequent line is the number of links. Following this line, are specifications for each link, which is 4. For example, "P 1 3", means a link between nodes 1 and 3, and where "P" indicates that the link is implemented by pipes.

Each node has a distinct id, which is a number that ranges from 0 to 127.

An even simpler configuration is where two hosts are connected by a link as shown in Figure 4.

Figure 4. Point to point connection between two hosts.

Your ex-coworker has included a configure file for Figure 4 named p2p.config:

```
2
H 0
H 1
1
P 0 1
```

The network simulator will work with this configuration because it has no switch nodes.

Nodes

Host nodes each have their own main directories, where it can access files. A host can

- Change its main directory.
- Upload a file from its main directory to the main directory of another host via the network.
- Download a file from the main directory of another host to its main directory via the network.
- Ping another host, which is to determine if the other host is connected via the network.

The host node organizes its work into "jobs". Example jobs:

- Send a packet.
- Initiate the ping process.

Jobs are put in a job queue, and the host is scheduled to take jobs off the job queue and execute them. Note that a job can create other jobs. For example, the job of initiating the ping process, will create another job of sending a ping packet.

Each host has a main loop, where it does the following in each pass

- Receive and execute a command from the manager.
  - This may create jobs, which are put in the job queue.
- Examine its incoming network link and convert a packet into a "job", then put the job in a job queue
  - A host will discard packets that are not destined for it, though there may be exceptions that depend on the packet type.
- Take exactly one job from the job queue, and execute the job.
  - This may create more jobs, which are put in the job queue.
- Go to sleep for ten milliseconds.
  - This delay will allow each host to process at the same rate, i.e., it will avoid some host being able to process much faster than other hosts. The delay will have all hosts process jobs at around the same rate of one per 10 milliseconds.

The following is an example of uploading a file from host 0 to host 1. The file is haha.txt in directory TestDir0, and it is to be transferred to TestDir1. The contents of haha.txt is “Hello world”.

Manager	**Host 0**	**Host 1**
Change current host to Host 1		
Command the current host to set its directory to TestDir1		
		Sets its main directory to TestDir1
Change current host to Host 0		
Command the current host to set its directory to TestDir0		
	Set it main directory to TestDir0	
Command the host to upload file “haha.txt” to Host 1		
	Create a packet with - type “initiate file upload” - destination = Host 1, and - payload = the file name “haha.txt” Put the packet into a job with type “send packet”. Put the job into the job queue Create a second packet with type “end file upload” with - destination = Host 1, and - payload = the contents of the file “haha.txt”, which is “hello world” Put the second packet into a job with the type “send packet”. Put the job into the job queue.	
	Take a job off the job queue. The job is to send the first packet. Send the packet	

		Receive the first packet. Create a job to process the packet.
		Take a job off the job queue. Process the job, which is to initialize a data structure. The data structure has a file buffer and a file name "haha.txt".
	Take a job off the job queue. The job is to send the second packet. Send the packet.	
		Receive a packet. Create a job to process the second packet.
		Take a job off the job queue. Process the job, which is to download the payload "hello world" of the packet into the file buffer. The file is opened with the file name "haha.txt" and the contents of the file buffer is stored in the file. The file is closed.

The following is another example, which is where Host 0 *pings* another host. The next example, shows what happens when Host 0 pings a disconnected host, say host 20. It also shows the actions of another host, Host 1.

Manager	**Host 0**	**Host 1**
Change current host to Host 0		
Ping Host 20		
	Create a ping packet with destination 20, and no payload. Put the packet into a job with type "send packet". Put the job into the job queue. Initialize the variable received_ping_reply=0.	

	Create a job with type “wait for ping reply”. Initialize the “timer” in the job to 5. Put the job into the job queue.	
	Take a job off the job queue. Send the packet.	
		Receive the packet. Since the packet is destined for 20 and not for Host 1, ignore and discard the packet.
	Take a job off the job queue, which is “wait for ping reply”. Check receive_ping_reply, which is still 0 Reduce “timer” to 4. Put the job back into the job queue.	
	Take a job off the job queue, which is “wait for ping reply”. Check receive_ping_reply, which is still 0 Reduce “timer” to 3. Put the job back into the job queue.	
	Take a job off the job queue, which is “wait for ping reply”. Check receive_ping_reply, which is still 0 Reduce “timer” to 2. Put the job back into the job queue.	
	Take a job off the job queue, which is “wait for ping reply”. Check receive_ping_reply, which is still 0 Reduce “timer” to 1. Put the job back into the job queue.	
	Take a job off the job queue, which is “wait for ping reply”. Check receive_ping_reply, which is still 0 Reduce “timer” to 0. This means that there was no ping reply after 5 timer units, so the	

	host will conclude that it's not connected to Host 20. Send a reply back to the manager "Host 20 does not reply to ping"	
Receive and display the reply, ping failed.		

The next example, shows a successful ping to Host 1.

Manager	**Host 0**	**Host 1**
Change current host to Host 0		
Ping Host 1		
	Create a ping packet with - Destination 1 - No payload. Put the packet into a job with type "send packet". Put the job into the job queue. Initialize the variable received_ping_reply=0. Create a job with type "wait for ping reply". Initialize the "timer" in the job to 5. Put the job into the job queue.	
	Take a job off the job queue. Send the packet.	
		Receive a packet. Create a ping reply packet. Create a send packet job for the packet. Add the job to the job queue.
		Take a job off the job queue. Send the packet.
	Receive a packet. Create a job "received a ping reply". Add the job to the job queue.	

	Take a job off the job queue, which is "wait for ping reply". Check receive_ping_reply, which is still 0 Reduce "timer" to 4. Put the job back into the job queue.	
	Take a job off the job queue, which is "received a ping reply". Set variable received_ping_reply=1.	
	Take a job off the job queue, which is "wait for ping reply". Check receive_ping_reply, which is now 1. This means that a ping reply was received. Send a reply back to the manager "Connected to Host 1"	
Receive and display the reply, ping succeeded		

2.3 Switch Node

Switch nodes just relay packets between host nodes. They do not upload or download data files. Note that host nodes do not relay packets.

Switch nodes each maintain a *forwarding table*, as shown in Figure 4. Assume that the table will have 100 entries.

Valid	Destination (Host Id)	Port #
1	3	0
1	0	2
0		
1	1	1

Figure 4. Forwarding table.

The valid column indicates whether the entry is valid or empty. The destination column has the IDs of all host nodes known to the switch node. The Port # column has the link-port to use when sending a packet to the host node. For example, to send a packet to Host 0, the switch will use port # 2.

If a packet has a destination that is not in the table, then the packet is sent to all outgoing links except the one it was received on. For example, suppose a switch has four ports numbered 0, 1, 2, and 3. Suppose a packet is

received on port 2, and its destination is not in the forwarding table. Then the packet is sent on ports 0, 1, and 3.

Note that the packet is not sent on port 2 because the packet just came from the port. But since it is unknown where the packet should be sent to, the switch will send the packets to all the other ports.

The forwarding table is initially empty, i.e., all its entries are invalid. Then as packets are processed, the table is updated as follows. Suppose a packet arrives on port 2 that is sourced at Host 1 (it can find out that it's sourced from Host 1 from the source field). The switch node can conclude that the packet came from Host 1. Since the packet came from port 2, it knows that port 2 will lead to Host 1. If the forwarding table has no entry for Host 1, then it creates an entry for Host 1 and records port 2. As an example, Figure 5 shows the updates of a forwarding table when a ping occurs given an initially empty forwarding table.

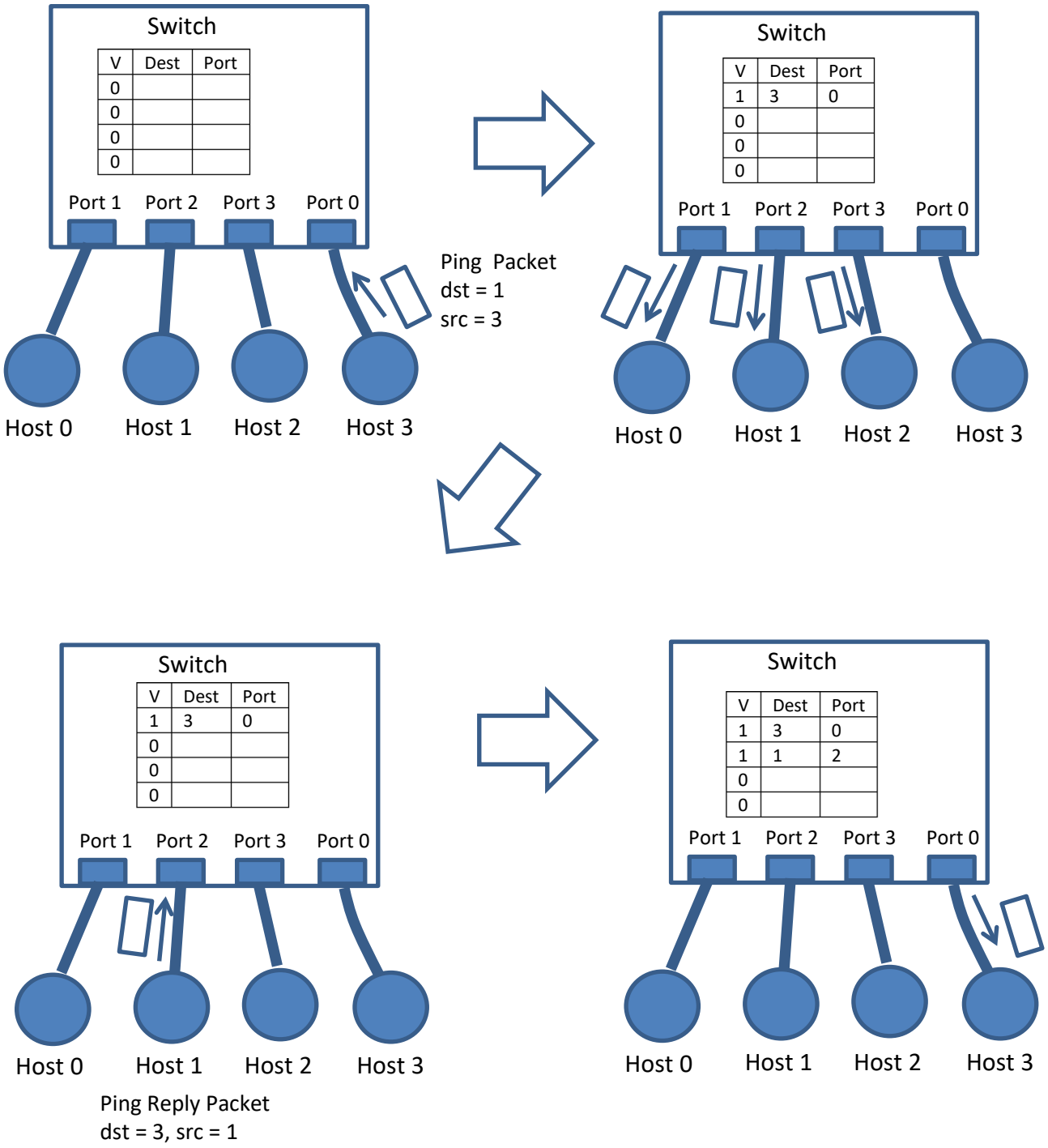

Figure 5. Ping from host 3 to Host 1.

Note that in the figure, hosts discard packets with destinations different than the host.

The switching node is similar to a host node. It runs an infinite loop, where at each pass through the loop, it first examines all incoming links for arriving packets, converts them to jobs, and adds them into a job queue. Then it processes one job from the job queue, and goes to sleep for 10 milliseconds.

2.4 Manager

The manager is connected to each of the hosts through pipes. It sends commands to a host by sending a text message through the pipe attached to the host. The first character of the message indicates a command. For example, 'p' is a ping request. This is followed by a space and then possibly other parameters for the command. For example, 'p 1' means to "ping host 1".

After the host receives a command message through the pipe, it parses the message to find the command and the parameters. After it completes execution of the command, it sends a reply message back to the manager via the pipe if necessary. Note that execution of a command may take some time.

These are the current commands

(s) Display the host's state. (Currently this is the main directory of the host)
(m) Set host's main directory
(h) Display all host
(c) Change host
(p) Ping a host
(u) Upload a file to a host
(d) Download a file from a host
(q) Quit

3 Implementation

The following are the files of the software, so far

main.c

This is the main routine. It creates the network nodes as children processes. Then it goes to the main routine for the manager.

net.c

It reads a configuration file. Then it creates the pipes that will implement the communication links. For a node, it provides functions/methods to find the ports of its links.

host.c

This is the host node software including the main routine for the host.

man.c

This is the manager software including the main routine of the manager.

packet.c

This has methods to send and receive packets through pipes.

4 Assignment

Your boss asks you to make three improvements

4.1 Improvement 1: File transfers

The currently host nodes can upload files to another host node, but the files must be at most 100 bytes. Improve the upload so that it can transfer files of at most 1000 bytes. Use the same packet format as before, but you will have to break up a large file into multiple packets. The packets are transferred and reassembled at the destination host. You may specify and use other types of packets but the payload must be no more than 100 bytes.

Also implement a file download from another host, where a file can have up to 1000 bytes. For this operation, a host will request a download of a file from another host, and provide the file name. The destination node will either transfer the file to the requesting host or indicate it has no such file in its current directory.

4.2 Improvement 2: Switch node

Allow switch nodes. Your boss suggests that you create a new files switch.c and possibly switch.h

4.3 Improvement 3: Sockets as a link option

Network links are currently implemented with pipes. The limitation with pipes is that it works only within the run of a program. It doesn't allow the situation where the network is composed of two subnetworks in different machines. To connect the two subnetworks, we can use sockets, as shown in Figure 6.

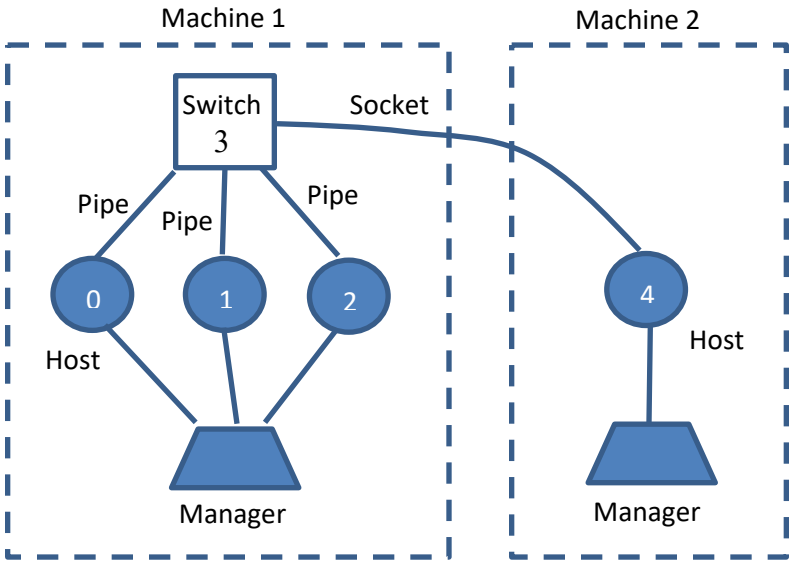

Figure 6. Link implemented by sockets.

For the scenario in Figure 6, you would have two configuration files and two managers. Each manager would be connected to all host nodes in its configuration file.

In the configuration file, the list of links can include socket links, e.g.,

S 4 wiliki.eng.hawaii.edu 3000 spectra.eng.hawaii.edu 3001

where

- "S" indicates a socket link
- "4" is the node that is attached to the socket link
- "wiliki.eng.hawaii.edu" and "3000" is the domain name and TCP port number, respectively, corresponding to the link port attached to node 4
- "spectra.eng.hawaii.edu" and "3001" is the domain name and TCP port number, respectively, corresponding to the port attached to the other node of the link.

With this information, you can set up sockets to send and receive packets from node 4.

Your group should have three members. If the number of students isn't a multiple of 3 then we may have groups of size 4. But these large groups will be formed at the very end. Note that the number of groups of size four are # Students % 3, e.g., if # Students = 17 then # Students % 3 = 2.

This is a long lab project, so it's important to have structure in your team and a time schedule to follow. Do the following related to teamwork:

- A group must write a plan to complete the assignment which includes
    - Dividing the assignment into *tasks*.
        - Note that some tasks could be done in parallel. For example, the three implementations can be done in parallel (or nearly parallel, where the code can be merged but with some fixes).
    - Estimating the amount of time to do the tasks.
    - Creating a *schedule* of when the tasks are to be completed, i.e., there should be a deadline to complete each task.
    - Assigning the tasks to group members
        - Note that tasks can be done in parallel
    - Drawing a Gantt chart for the tasks
    - Determining a *leader*, who is responsible for the completion of the assignment. The leader's responsibilities includes
        - Determine if the group is on schedule
        - If it's not on schedule then rework schedule, reassign group members to tasks, or add a group member to a task that is behind schedule.

      A group can take turns being the leader but they must have a leader at all times.

  Submit the plan to the TA, who will evaluate the group's progress. Any subsequent modifications to the plan must be submitted to the TA.

You will explain your group's organization in a *future* written assignment, which should be Lab 9. You will also rate your group's collaboration in Lab 10. Your grade will depend on whether you provided the explanation and ratings, rather than how well your group worked as a team. This information will be used in evaluate the level of achievement of the CENG program's Student Outcome 5:

> An ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives.

(The CENG student outcome's are presented on the Department web site:
Objectives & Outcomes | UH Electrical & Computer Engineering (hawaii.edu)

The following are performance indicators and rubrics to measure the level of achievement.

Performance Indicator	Unsatisfactory	Marginal	Satisfactory	Exemplary
a) Team Organization	Not organized as a team; no clear roles or tasks are defined.	Loosely organized with some leadership; not well-defined team roles.	Tightly organized with leadership; clear-defined team roles.	Well organized as a team with strong leadership; clear-defined team roles.
b) Team Collaborations	Show almost no collaboration.	Very little collaboration; everyone performs individually.	Show clear collaboration among member.	Well-planned strong collaboration, and active interactions among members; show team effect.
c) Project Goals	No clear goals are defined.	Goals are defined but vague and not well-justified.	Clear goals are defined.	Clear goals are defined with strong motivations.
d) Project Plans	No concrete plans are given.	Plans are given but not-well-justified.	Clear plans are defined.	Clear plans are defined with solid ideas to achieve goals.

Attachments

Attached to this assignment is a tarred and gzipped directory LabSwitch that has

- the network simulator files: main.c, net.c, man.c, host.c, packet.c and their header files
- Test directories: TestDir0 and TestDir1
- p2p.config, which is a network configuration file for a network of two host nodes connected by a pipe link.