

Last updated: 7/17/96, Ting-hsien Lin
First author: Shivkumar Kalyanaraman

Latest changes are in the end of the file...

Explanation of the configurations:

All configurations have three nodes, each having 3 layers: the APPLICATION layer, the DLC LAYER, and the PHYSICAL LAYER. The physical layers are connected by point-to-point links. There are links between nodes 1 and 2, 2 and 3, and 3 and 1. There is a connection from application layers 1 to 2, 2 to 3, and 3 to 1. Data flows only on these connections. Links are characterized by a transmission speed, a propagation delay, and an error probability.

3nodes_basic.config:

This configuration is a symmetric configuration, with all links having the same (transmission) speed (155.52Mbps), (propagation) delay (5 us), and error probability (zero). You should see packets going from source 1's application layer to source 2's application layer, and similarly from source j's application layer to source (j+1) mod 3 's application layer.

In the graphical interface, you should see the packets leaving the sources at the same time, and arriving at their destinations at the same instant.

The graph for this configuration has the number of packets received plotted against time for each of node 1, node 2 and node 3. You should see a single line on the graph, where all the curves overlap.

Recommended delay: 1000

3nodes_delay.config:

This configuration differs from the 3nodes_basic.config in that the link propagation delays are different. The link between nodes 1 and 2 (link1) has propagation delay 5 us. The link between nodes 2 and 3 (link2) has propagation delay 50 us. The link between nodes 1 and 2 (link3) has propagation delay 200 us.

In the graphical interface, you should see the maximum packets on link1 (closely clustered), some packets on link2 (somewhat clustered), and at most one packet on link3.

The graph for this configuration has the number of packets received plotted against time for each of node 1, node 2 and node 3. You should see three distinct lines in the graph.

Recommended delay: 1000

3nodes_error.config:

This configuration differs from the 3nodes_basic.config in that the link error rates are different. link1 has error probability .1, link2 has error probability of 0.2, link3 has error probability of 0.3.

The errors are detected in the DLC layer. If an error is detected, the packet is not passed on to the APPLICATION layer. Hence, in the graphical interface, you will see many packets at app3 and app1 terminating at the DLC layer. However, some packets, which are not in error pass through to the application.

The graph for this configuration has the number of packets received plotted against time for each of node 1, node 2 and node 3. You should see three distinct lines in the graph, corresponding to the number of packets received without error at each destination node.

Recommended delay: 20000

3nodes_speed.config:

This configuration differs from the 3nodes_basic.config in that the link transmission speeds are different. link1 has transmission speed 155.52Mbps. link2 has transmission speed 77.76Mbps (1/2 of link1). link3 has transmission speed 38.88Mbps (1/2 of link2).

In the graphical interface, you will see 1 packet on link2 for every 2 packets on link1. Similarly, you will see 1 packet on link3 for every 2 packets on link2.

The graph for this configuration has the number of packets received plotted against time for each of node 1, node 2 and node 3. You should see three distinct lines in the graph, corresponding to the number of packets received at each destination node, depending upon the transmission rates of the links.

Recommended delay: 1000

Answers to other FREQUENTLY ASKED QUESTIONS:

1. To load a file from the command line:
For example

```
lab1_exec 3nodes_speed.config
```

or

```
lab1_demo 3nodes_speed.config &
```

2. Any doubts, questions, immediately e-mail durreesi@cis.

3. To dismiss "Help Window", only middle button works, as of now.

{Everytime you click, there is a help window created. So, please be patient after you clicked on HELP the first time. It takes a few seconds to come up.}

4. Ignore the files libcomp.a and dlc_layer.o which are created in your directory after compilation. You can delete them if you wish, but then the makefile will recompile, even if there are no source code changes.

5. A copy of the lab1 handout is in:
/usr/class/cis677/Lab1/lab1.ps

6. drawgraphs: Drawgraphs will work only after you have compiled your program and verified that it works with the graphical interface. All you have to do is type: drawgraphs
{nothing more, nothing less}

The sample graph /usr/class/cis677/Lab1/3nodes_basic.config.ps is there for comparison purpose.

7. lab1_demo cannot be used to produced graphs, since we have disabled the graphical output information.

8. "Check for error" means:
Check if the error bit has been set in the incoming packet from the physical layer { setting of this bit means that error has been detected in the packet by the physical layer }. In this lab, we discard the packet; while in the next lab, we recover from packet errors.

9. To view these .ps files produced by 'drawgraphs', you can use 'ghostview'. For example:

```
ghostview 3nodes_basic.config.ps
```

From the ghostview menu, choose 'Orientation' and set the option to 'Landscape'

To print the graphs, you can use:

```
lpr -d<printer name> 3nodes*.ps
```

o SUBMITTING:

To submit your lab1:

```
submit cis677t1 lab1 file1 file2 ...
```

NOTE: Please do not submit the executable since the file is too large. Submit the remaining things specified in the handout.

o NON-CIS MAJORS and others with disk space problems

You probably don't have enough disk quota to produce the executable from your home directory. The executable requires about 1 Meg of disk space.

So I recommend that if you run into space problems, you use the /tmp directory to compile and run. Copy all your files to /tmp once you have finished editing and then do a make. Remember to copy things back if you need them because /tmp gets deleted periodically.

o What does bcopy() do?

bcopy() copies one string to another character by character.

```
void bcopy(const char *s1, char *s2, int n);
```

bcopy() copies n bytes from the area pointed to by s1 to the area pointed to by s2.

Remember to use addresses for bcopy(), and use A_PDU_SIZE as the number of bytes i.e.,

```
bcopy((char *)&(), (char *)&(), A_PDU_SIZE)
```

Also note that &pdu_from_application is different from &(pdu_from_application->u.a_pdu)

o Why do we have to use bcopy()? Why can't I simply write this?
pdu_to_physical->u.d_pdu.a_pdu = pdu_from_application->u.a_pdu;

Yes, you can use the assignment of structure instead of bcopy(), as long as the compiler supports it. gcc happens to be one. However, not all compilers support assignment of structure. You may rely on gcc for doing the copy for you. But later on when you use another compiler which does not support this feature, you may run into compiling error without realizing what is going on.

To make the code portable, we use bcopy() instead of relying on the compiler. When you write your code, whether lab or real work, you may want to take portability into account.

For those who know memcpy() already, bcopy() does similar job as memcpy(), except that the order of arguments are different.

(revised tlin@cis, 7/17/96)

- o What is "error" field in D_PDU_TYPE for? What do we have to do with it?

The sending DLC knows the packet has no error in it because it is the message that DLC wants to send, so error flag is set to NO by the sending DLC. On the receiving side, the receiving PHY checks CRC to see if any error occurs when the packet passed the link. The receiving PHY sets error flag if any transmission error is detected. The receiving DLC should check the error flag, and passes only correct packets to receiving APP. Packets with error is silently discarded. (Why?)

- o curr_node, next_node

curr_node and next_node in d_pdu
is same as the snode and dnode in the corresponding a_pdu.