

## Project 3, CSci 330, Fall 2006

*Due: Dec 1, 2:10pm. Value: 60 pts.*

Your assignment is to experiment with conflict resolution schemes and to write a summary of what you found, including an English description of your best algorithm and statistics regarding its success. Your paper should present experimental results comparing your algorithm to at least the following two simple conflict resolution protocols:

- When a program requests that a message be sent, the station attempts to send the message immediately.
- When a program requests that a message be sent, the station waits until it believes the network is idle, and then it sends the message immediately. When the network appears idle already, there is no wait.

The second approach is obviously better than the first. But it is also obviously suboptimal: During the time that a long message is on the network, several stations may get requests to send messages, and they'll all try to dump them as soon as the long message completes. All those messages will be lost.

To help with your simulation, you can download a Java program from the Web page for simulating a network at the physical layer. Right now, it implements the first of the protocols above. While you are welcome to modify all parts of the program (and indeed you are welcome to write your own simulation program), this program has been designed so that you need only to modify the `Station` class to implement your protocols and to modify the constants in `Main` to configure the simulation behavior.

The simulation posted on the Web makes several important simplifying assumptions.

- The only message type involved is a broadcast to all other computers on the network. This removes all concerns about which station is supposed to receive the message.
- All events occur at discrete time steps. You can think of each step corresponding to one microsecond.
- The network is in a simple star configuration: If one station begins to send a message at time  $T$ , then all other stations begin to receive the message at time  $T + L$ , where  $L$  is the network's latency. If the message has length  $\ell$  and the throughput is  $r$ , then the sending station completes sending the message at time  $T + \ell/r$ , and the receiving station(s) complete receiving the message at time  $T + L + \ell/r$ .
- If more than one message exists on the network at any instant, then all the messages on the network at that instant fail. (This is not entirely realistic: If the same station sends two messages in succession, they do not necessarily interfere with each other.) Stations are not notified about message failures.
- For each station, the probability that it may be requested to send a message is a fixed probability for each simulation step; but when there is already a message that the station is to send, and that message is pending or not yet complete, the probability is 0. The message requests at a station occur entirely independently of what is happening at other stations.
- Message lengths are uniformly distributed between the minimum and maximum lengths.
- When a station is to send a message, the entire message must be sent whole; it cannot be broken into smaller frames. (Rather than loosening this restriction, a reasonable approximation to loosening it can be done by adjusting the message size and the number of messages requested.)
- Once the station begins to send a message, it is committed to sending the message in its entirety; it cannot be canceled midway through (as would be appropriate when a conflict is detected).

The primary product of your project is a paper outlining your approach(es) and experimental results concerning their behavior. Of course, the experimental results should cover a reasonable spectrum of scenarios. You should also e-mail me the simulation code for the protocol you feel performs best.