CSCI 4273/5273: Network Systems
Fall 2014
Programming Assignment 4: Extra Credit
Due Date: 12/05/2014

You may earn up to 5% extra credit for your final grade by extending your programming assignment 4. In order to earn these extra credit, you must first successfully complete programming assignment 4. Your extra credit assignment will not be graded if you do not submit a working version (Determined by the TA) of programming assignment four. There will be separate links for submitting your programming assignment and your extra credit work.

Team: You may work in teams of up to two students on this extra credit assignment.

Extend your implementation of the two network models by incorporating *segmentation and reassembly* functionality in IP. For the purpose of this assignment, assume that an IP packet can be atmost 200 bytes long including the IP header. Use the following format for IP header:

hlp	Ident	Flags	Offset	Data Length

Here the *hlp* and *Data Length* fields are same as in Programming Assignment Four. The *Other Info* field has been divided into three subfields: *Ident*, *Flags* and *Offset*. You will use these subfields for implementing IP fragmentation and reassembly. *Ident* field is 16 bits long, *Flags* field is 3 bits long, and *Offset* field is 13 bits long. Recall that *Ident* is used to identify the original message that is fragmented, *Flags* field is used to identify whether a fragment is the last fragment of the original message, and *Offset* field is used to identify the offset of the fragment from the beginning of the original message. Also, recall that the offset is represented as the number of octets. Since, you are using UDP to simulate the network, there is no guarantee that all fragments will arrive at the destination. Discard the fragments at the destination after 15 ms. Use your Event Scheduler library for this.

Applications

Implement one application (one thread) per every top-level protocol (four applications in total) on both hosts. These applications repeatedly execute the following code 100 times:

create a message (message size: see note below)
send message

- the *application thread writes to the send pipe of the application-level protocol in process-per-protocol models*
- *the application thread invokes the send() function of the application-level protocol in process-per-message model*

sleep for a few milliseconds

Use 10 different message sizes (in bytes) sent in the following order: 50, 300, 100, 350, 150, 200, 75, 250, 125, 400.

Evaluation

Measure the total time required to complete all four applications (100 messages sent by each of the four applications from two hosts and all 400 messages received by each host) under three different experimental setups: (1) Both hosts are running process-per-protocol implementations; (2) Both hosts are running process-per-message implementations; and (3) One host is running process-per-protocol implementation and the other host is running process-per-message implementation.

Assignment Submission

1. Submission deadline is Friday, December 05, midnight. No late submissions will be allowed, unless there is a valid excuse.
2. Submit a single zip file via the submission link for Extra Credit on Moodle. This link is separate from the submission link for programming assignment four. Your zip file must contain all your source code including all three libraries, Makefile and README.
3. DO NOT include any object files in your submission.
4. In the README file, provide the following information: Your name; instructions on how to compile and run your program; current status of your program: whether it compiles or not, known bugs/limitations/unusual features, what parts of the program work, etc.; any other information that will be useful in grading your program. In addition, include the times you measured for the three scenarios.