CSC / ECE 573 Internet Protocols, Fall 2005

Project Part II

Due Date

- Tuesday, October 25, at 11:45pm

Instructions

- Put your name (both names if joint with another person), "Project, Part II ", and date at the top of the first page.
- Do not plagiarize; that means, do not copy content from any source without permission from the instructor, and if permitted, acknowledge the source.
- Submit this part to both persons' lockers if doing jointly. They will be identical in this case, and have the name of both persons at the top of page 1.

Description

DO NOT COPY CODE FROM SOMEONE ELSE'S IMPLEMENTATION OF THE PROTOCOL!

In Part I of the project you picked a protocol to implement. In this part, you will begin implementation, testing, and documentation.

Your plan of implementation and testing should be done in stages, like any non-trivial programming project. At the end of each step, you should test what you've got and make sure it works before going on to the next step. For your project, you should make a list of protocol functions, putting them in order of functionality (basic to advanced), and divide into the 3 stages (corresponding to parts II, III, and IV of this project) that you will implement. Shown below is the beginning of a list for TCP, just to give you the idea. (If you are doing TCP, you do not have to use this list; it is only an example.) Your list contains not just the functions, but how you plan to test them as well (as you can see in the example below).

1. Write a program that creates a UDP datagram, transmits with some fixed payload, and can receive the datagram and interpret the payload value correctly.
2. Extend this program to read a file, transmitting it one UDP datagram at a time in fixed size blocks, receiving it at the destination and writing the blocks as they are received into a file (blocks may be missing, out of order, or duplicated, but you're not worrying about that at this point. Also don't worry about two-way transmission with acknowledgments at this point). Compute block length at the receiver from the length of the UDP data received.
3. Define the TCP header. Correctly set port and destination numbers (same as in UDP datagram), header length (no options at this stage), reserved field, and checksum. Set Sequence Number, Acknowledgment Number, Reserved, Control Flags, Window Size, and Urgent Pointer all to 0's for now. Send the file as above, but before each data block, insert this application-layer TCP header. Verify at the receiver you can interpret the header contents correctly and that the checksum is computed properly.
4. Implement the Sequence Number field based on the bytes transmitted, verify that this is computed (with wraparound) and received properly. Again, don't worry about missing, out of order, or duplicate segments at this point.
5. For each datagram received, send an Acknowledgment TCP segment back, containing no data, but with the proper values for Acknowledgment Number, ACK Flag, and other fields mentioned above. Verify you can receive and interpret this Acknowledgment segment correctly and that the Acknowledgment Number field is implemented correctly. Note that at this point each endpoint is interleaving sending and receiving, so communication is bidirectional. (Note: Make the receive operation non-blocking, or you will have a Stop-and-Wait flow control mechanism, which is not right!)
6. Continue in this fashion for State Machine implementation (connection establishment and termination), Flow Control (Window Size advertisements), Sliding Windows, RTT estimation and RTO calculations, timeout and retransmission, etc. etc.

Begin the document that you will turn in at the end of the project. For parts III and IV, you will simply extend and bring up to date this document, so it is a "work in progress". Quality of writing and completeness count; formatting and length do not. The document contents are:

1. Title of project, names of partners, date, "Project Part II"
2. Protocol chosen to implement
3. Platform and programming language used
4. List of functions, as above, with test steps.
5. Indicate which steps you completed in stage 1 (this part of the project).
6. Also indicate what protocol functions will *not* be implemented, or implemented only if a miracle happens and there is more time than you expect :-)
7. *--Write the description of the extension you picked, in Internet Draft style, and indicate where and in what RFC you would put this extension (if RFCs were allowed to be updated, which they aren't - once standardized, they are "archival' documents).*
8. *--Instructions for installing, compiling and executing (or interpreting) your code. What OS, what release of compiler / JRE, and/or libraries are needed, and any makefiles, switches etc. that need to be created or set.*
9. *--Instructions for running your software, including documentation of parameters if there are any.*
10. Results of testing each function (can be summarized if output is voluminous). Performance is worth measuring as well as correctness / completeness, particularly for TCP transfers.
11. References you used (RFC's, books, websites, etc.)
12. Listing of source code. There should be enough comments to follow the code with modest effort. It is not necessary to put
    ```
    /* add 1 to the count */
    ```
    as a comment for the instruction
    ```
    cnt++;
    ```
    ; let's be realistic. Just make it possible to follow the flow of execution and purpose of each module, and puzzle out any particularly inscrutable or cryptic code. You don't need to explain the operation of the state machine, sliding windows, circular buffers, etc. in your comments - take it for granted the reader knows all that stuff. Just say things like
    ```
    /* state machine transitions implemented below, conforms to TCP
    /* standard except for .... */
    ```
    , for instance.

Submit this document, current through the parts you have implemented and tested, for fulfillment of Part II of the project. The items in italics above can be represented by just stubs or headings, with no content at this time. It is possible to fulfill this part of the project in 4 single-spaced pages, not counting source code or the italicized items.

REMINDER: DO NOT COPY CODE FROM SOMEONE ELSE'S IMPLEMENTATION OF THE PROTOCOL!

Created on October 7 , 2005
Last Modified October 7, 2005
Maintained by Douglas S. Reeves