UNIVERSITY OF DELAWARE
DEPARTMENT OF COMPUTER & INFORMATION SCIENCES

CISC 450 / CPEG 419: Computer Networks I

Fall Semester, 2019 Professor: Adarsh Sethi

Programming Project 1
Due Date: 11 pm on Friday October 11, 2019

This project is a group assignment; you should work on it with your group partner.
Collaboration with or help from anyone except your group partner is not permitted.
Only one submission per group is required.

Important note: Before you start this project, read the accompanying document *Project 1 Help,* which contains important and useful information about this project.

In this project, you will implement a client and a server which use TCP to transmit a file from the server to the client. Both the client and the server must run on the course VM *cisc450.cis.udel.edu*. You may select a port number for the server that does not conflict with port numbers chosen by other students in the course. For convenience, both the hostname and the server port number may be hardcoded into both the client and server programs, but this should be done in such a way that they are easy to change.

The server starts by waiting for a TCP connection request from the client.

The client starts by prompting the user to enter the name of the file to be transferred. The client then establishes a TCP connection with the server. After the connection is established, the client sends the filename to the server. The server reads the file and sends it to the client in a series of packets as described below. The client receives the file and stores it with the name *out.txt*. When the file transfer is complete, both the client and the server terminate execution.

The server constructs packets by reading lines one at a time from the input file. Each line in the input file contains a sequence of printable characters (no control characters, etc.), with no more than 80 characters on a line. The "newline" character read from the file at the end of each line is also transmitted in the packet and is included within the limit of 80 characters per line. The server transmits each line to the client in a separate packet. The client receives the packets and puts their data into distinct lines in the output file.

The format of a data packet is shown in the figure below:

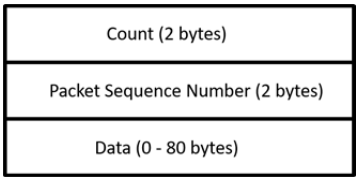

Figure 1: Format of a data packet

Each data packet contains a 4-byte long header followed by a number of data characters. The header contains 2 fields, each of length 16 bits (2 bytes) as shown in Figure 1. You must convert the values in these fields into the network byte order when they are transmitted, and convert them back to host byte order when they are received.

The first field of the header is a count of the number of data characters in the packet. This value must be in the range 0 through 80. If the count is 0, then there are no data characters in the packet. The second field of the header is called the packet sequence number. Each packet transmitted by the server is assigned a sequence number starting with 1 and incremented by 1 for each packet.

When the server is finished transmitting all the lines in the data file, it will send a special last packet signifying "End of Transmission" (EOT). This packet will have a Count of 0 and no data characters. It will have a Sequence Number that is the next sequence number that would have been used if this were a valid data packet. The server program can terminate once this packet has been transmitted. When the client receives the EOT packet, it closes the output file and also terminates.

The filename is sent by the client to the server using the same format as the data packet shown above. In this packet, the data characters will contain the filename, but there will not be any newline character at the end. The Count will be the number of data characters in the packet (i.e. length of the filename), and the sequence number will be 0.

The transmission of each packet will be done using two TCP *send* operations, one to send the header and the second to send the data characters. You may use two separate buffers for this purpose. Similarly, the receiver will use two TCP *receive* operations to receive the header and the data bytes. See the *Project 1 Help* document for more guidance on how to send and receive these two parts of the packet.

Output of your program

At specific places in both your client and server programs, you must print out specific messages. The symbol "*n*" below refers to the sequence number of the transmitted or received packet, and the symbol "*c*" below refers to the count (number of data bytes) in the transmitted or received packet.

The messages to be printed are:

When a new data packet numbered *n* is sent by the server:
Packet n transmitted with c data bytes
When a data packet numbered *n* is received by the client:
Packet n received with c data bytes
When the "End of Transmission" packet is sent:
End of Transmission Packet with sequence number n transmitted with c data bytes
When the "End of Transmission" packet is received:
End of Transmission Packet with sequence number n received with c data bytes

At the end, before terminating execution, the following statistics should be printed. Do not include the last special "End of Transmission" packet in the count of data packets in these statistics. Also do not include the first packet sent by the client with the filename.

For server:

Number of data packets transmitted

Total number of data bytes transmitted (this should be the sum of the count fields of all transmitted packets)

For client:

Number of data packets received

Total number of data bytes received (this should be the sum of the count fields of all received packets)

Submission

The files *test1.txt* and *test2.txt* in the directory */usa/sethi/networks/proj1* on *cisc450.cis.udel.edu* are sample input files that may be used by you to test your programs.

When you are ready to do your project submission, create two scripts using the Unix *script* command. One script will show the execution of the client and the other script will show the execution of the server. Show the transfer of the file *test2.txt* when you prepare these scripts. The server script should contain a long listing of the directory that contains your files (using *ls -l*), should show them being compiled, then another long listing (using *ls -l*) of the directory after compilation is complete, and then show the execution of the server program including the program outputs. The client script should show a long listing (using *ls -l*) of the directory after compilation has been completed in the server script and then show the execution of the client program including the program outputs. Finally, in the client script, do a *diff* on the input and output files. When you are finished, exit both scripts.

In your project directory, create a *ReadMe* text file with the following information: The names of the two students in the group who worked on this project, the names of the source files, the names of the files that contain the executables, and the names of the files containing the scripts with the test runs.

Submit a zipped copy of your project directory. This directory should include all your original source files, the executables, the input and output data files, the scripts generated by you for your test run, and the *ReadMe* file.

Grading

Your programs will be graded on correctness, proper output, readability, and documentation, as specified in the project rubric. Points for documentation will not be awarded lightly; we will be looking for meaningful variable and function names, good use of comments, good modular structure with appropriate use of functions, good programming style, and proper indentation. A major part of this code will be re-used in Project 2, so using good software engineering principles will help you.

Deadline for submission: Friday October 11, 11 pm. The course's standard late policy for assignments will apply.