CS5220 Project 2: Socket-based ARQ Protocol Programming

Note: You can choose to do the project in C++ instead of C, but you have to use the same lower-level socket api that is available to c (i.e. socket.h etc). And, you will provide all the scripts/instructions for building it, so running it will be exactly the same as if it were C.

Attachments:

TCP_Client-1.c
TCP_Server.c
SIGCOMM10-DataCenterTCP-2.pdf
UDP_EchoClient.c
UDP_EchoServer.c

Reading: Chapter 6.1, The Transport Service (pp 497-506). And, Chapter 3.3 of the textbook for sequencing, ACKs, and retransmission mechanisms.

Linux servers to use: blanca.uccs.edu, crestone.uccs.edu, shavano.uccs.edu, and windom.uccs.edu. Use your own UCCS credentials to access the Linux servers. You need to connect to VPN first if you are accessing them from outside of UCCS!

Part 1. A file server in TCP (3 pts)

The source code of both client-side and server-side are downloadable; see attached TCP Client in C and TCP Server in C for reference. Current client code displays the text file on the standard output. Modify client-side code so that a binary file, transferred from the server, will be saved as a file in the local directory. Make both sides running on the Linux computers (by remote access). Note that those Linux servers are sharing the same file system, you need to create a new directory (using your username) to save the received file at the client-side.

Server-side: TCP_server
Client-side: TCP_client server_hostname filename

TCP_client is the executable program. server_hostname is the hostname of the machine you use as the server (please use **windom**). the filename is the file that the client wants to read from the server. We assume that the server has the requested file (you should store several PDF files there). Use the attached PDF file to test your program. You may use binary I/O in the file operations. You may also want to save the transmitted file in a different directory due to the use of a shared file system in the Linux servers.

For self-checking, you should be able to open the test PDF file on the client machine after the TCP transmission is completed. Note that you may use 2XYZ as the port number in your program, in which XYZ is the last 3 digits of your student ID. Doing so would avoid that multiple programs/processes use the same port for communication.

Part 2. A file server in UDP with ARQ flow control (7pts)

Modify the source code of 3.1 using UDP sockets (instead of TCP sockets). Make both sides running on the Linux servers (by remote access). See attached UDP Client in C and UDP Server in C for reference. Then, enhance the reliability of your code by implementing two ARQ algorithms; stop-and-wait (3 **pts**), and Selective Repeat (**4 pts**). Refer to Section 3.4 of the textbook for sequencing, ACKs, and retransmission mechanisms. The communication should be reliable, assuring that the client got whole file correctly. Since UDP indeed is indeed quite reliable for communications in a LAN, you need to use a random generator/function to "purposely" drop X% of data blocks BEFORE they are delivered from the server-side. Each UDP segment should be less than 4KB so that you can have enough segments to experience a few segment losses.

Server-side: UDP_server loss_probability protocol_type

Client side: UDP_client server_hostname file_name protocol_type

Parameter loss_probability is the dropping probability (X) as a command-line input of the server code. Parameter protocol_type is the ARQ type implemented (1 for Stop-and-Wait, 2 for Selective Repeat). server_hostname is the hostname of the machine you use as the server (e.g., use **windom** server). filename is the file that the client wants to read from the server. Clearly, we assume that the server has the requested file. Use the attached PDF file to test your programs. You should be able to open the test PDF file on the client machine correctly. You may save the transmitted file in a different directory due to the use of a shared file system in our Unix/Unix computers. Again, you may use 2XYZ as the port number in your program, in which XYZ is the last 3 digits of your student ID. Doing so would avoid that multiple programs/processes use the same port for communication.

What to turn in

You will do the project individually. You can discuss with your classmates, but you must code and implement the project by yourself (no copy from your classmates).

For this project you will upload Canvas one ZIP file (in the name of P2_Yourlastnames) that contains all source code, together with a technical report specifying 1) how to compile and execute your program; 2) the self-testing results of your program and any important feature you want to specify. If your program does not work, please notify it clearly! The technical report should contain the trace of the flow control for the lost frames (i.e., Seq#, ACK#, Retransmission#).

I may check your project during class time by remotely accessing the Linux servers.

You may search the Internet for more helpful information on socket-based network programming.