【2021 Advanced Computer Networks Homework 6】

Rules:

1. Please finish this homework under Ubuntu 20.04 OS system.
2. You must use C language to complete this homework with makefile.
3. You are not allowed to copy the other's homework. Otherwise, you will fail this course.
4. Compress this assignment to a zip file, and name it "StudentID_TCPIP_HW6.zip" (e.g., M063040001_TCPIP_HW6. zip) and submit it to the Cyber University system http://cu.nsysu.edu.tw/ before the deadline.
5. If you have any question, please send email to TA at net_ta@net.nsysu.edu.tw or drop by Room EC5018 in the duration of 11:00 to 17:00. However, TA will not help you to debug program.
6. You have to deeply understand what your program do because TA will ask you about your program during demo.
7. Deadline: 2021/12/31 23:59. No delay is accepted. If you have any trouble, please notify us inadvance by email.

Request:

Section 61.13.1 noted that an alternative to out-of-band data would be to create two socket connections between the client and server: one for normal data and one for priority data. Write client and server programs that implement this framework. Here are a few require:

Create two hosts using the mininet, one host executes the server program and the other host executes the client program. The server needs some way of knowing which two sockets belong to the same client. One way to do this is to have the client first create a listening socket using an ephemeral port (i.e., binding to port 0). After obtaining the ephemeral port number of its listening socket (using getsockname()), the client connects its“normal” socket to the server’s listening socket and sends a message containing the port number of the client’s listening socket. The client then waits for the server to use the client’s listening socket to make a connection in the opposite direction for the “priority” socket. (The server can obtain the client’s IP address during the accept() of the normal connection.) Implement some type of security mechanism to prevent a rogue process from trying to connect to the client’s listening socket. To do this, the client could send a cookie (i.e., some type of unique message) to the server using the normal socket. The server would then return this cookie via the priority socket so that the client could verify it. In order to experiment with transmitting normal and priority data from the client to the server, you will need to code the server to multiplex the input from the two sockets using select() or poll() (described in Section 63.2).