CSE156/L Programming Lab 2: Simple File Echo

Serjo Barron
sepbarro
Winter 2024

Program Design (Server):

1. Parse arguments, mainly port number (for server socket)
2. Check if the port is valid
3. Create a server socket and bind it to the port number
4. Run an infinite while loop that does the following:
    a. Receive bytes from the client (when bytes are sent)
    b. Store the received bytes in a buffer (of size 4096)
    c. Send the bytes received back to the client (echo back to the client)
5. The program ends when terminated by the user

Program Design (Client):

1. Parse arguments to get: server IP, server port number, MTU, infile path, and outfile path
2. Check if the server port is valid
3. Check that the MTU is at least 1
4. Call a function called “sendfile()” which does the following:
    a. Open the infile in reading bytes mode, this will be what is sent to the server in packets
    b. Open or create the outfile path in writing bytes mode, this file will have the contents of the infile written to it in segments (whose max is MTU)
    c. Create a socket, and initialize it according to the server IP and port number
    d. Run an infinite while loop:
        i. Read bytes from infile in segments up to MTU
        ii. Send the bytes read to the server using the socket as a connection point
        iii. 10-second timeout begins
        iv. Wait for an ACK sequence number and check it is the same
        v. Receive the bytes echoed back by the server
        vi. If the timeout is reached, we assume packet loss and close the program
        vii. Write bytes read to outfile
    e. Close infile and outfile
    f. Close socket connection
5. Program closes successfully

Program Instructions:

1. Go to the top directory (where the Makefile is) and use the command “make” to create a binary executable file (for both server and client) in the bin directory
2. Change the directory to the bin directory (you will need two terminals open, one for the server and the other for the client)
3. To run the server:
    a. ./myserver <port number>

    b. The port number cannot be well known: 0-1024, but it may be within the range of 1024 - 65535, anything outside of that range is not permitted
4. To run the client:
    a. ./myclient <server IP> <port number> <MTU> <infile path> <outfile path>
    b. The server IP may be “127.0.0.1” which is localhost
    c. The port number cannot be well known: 0-1024, but it may be within the range of 1024 - 65535, anything outside of that range is not permitted (it should also be the same port number as the one used to initialize the server)
    d. The MTU must be at least 1 byte
    e. The infile path must exist before executing the client

Test Cases:

1. Invalid input: argument count
    a. ./myserver
        i. Missing port number (2 expected, 1 given)
    b. ./myclient 127.0.0.1 1024 4096 infile.txt
        i. Missing outfile path (6 expected, 5 given)
2. Invalid input: negative port number
    a. ./myserver -1024
        i. The server socket cannot be bound to a negative port number
3. Invalid input: well-known port number
    a. ./myserver 80
        i. Cannot bind to a well-known port number, these are reserved
4. Invalid input: out-of-range port number
    a. ./myserver 65536
        i. The port number is out of range of dynamic and ephemeral ports (1024-65535)
5. Invalid input: MTU < 1
    a. ./myclient 127.0.0.1 1024 0 infile.txt outfile.txt
        i. If the MTU is less than 1, bytes cannot be read to be echoed back to the client
6. Invalid input: infile path does not exist
    a. ./myclient 127.0.0.1 1024 4096 diri/doesnotexist.txt diro/outfiletxt
        i. The infile path does not exist, attempting fopen() on it will result in a NULL return
7. Invalid input: bad server IP
    a. ./myclient 10.0.0 64 diri/infile.txt diro/outfile.txt
        i. The client will timeout (set to 10 seconds) where packet loss is assumed
8. Valid input: in-range port number
    a. ./myserver 1024
        i. The server is running, port number is within 1024-65535
9. Valid input: small MTU = 1, large infile = 40124774 bytes
    a. ./myclient 127.0.0.1 1024 1 large.txt outfile.txt
        i. Runtime = 25.325s

10. Valid input: medium MTU = 1024, large infile = 40124774 bytes
    a. ./myclient 127.0.0.1 1024 1024 large.txt outfile.txt
        i. Runtime = 1.933s
11. Valid input: large MTU = 65536, large infile = 40124774 bytes
    a. ./myclient 127.0.0.1 1024 65536 large.txt outfile.txt
        i. Runtime = 0.002s