CS 39006: Networks Lab
Lab Test

Basic Rules:

You need to follow the rules strictly:

- You can use the tutorial on *Beej's Guide to Network Programming* during the lab test.
- You can download your own codes for previous assignments, and build the program over it.
- You can use Linux man pages.
- No other resources including Internet will be provided.
- Total time: 2 Hours
- Consulting with others will result in disqualification from the lab test.
- You can consult with the TAs and with the instructors for any doubt.

Problem Statement:

In this assignment you have to implement a client-server program using C/C++ Socket programming API.

Basic Objectives:

Our objective is to compare the time required to transfer data using two well known resource sharing policies used in the Internet – bandwidth division multiple access and time division multiple access.

For this you need to implement one server and one client over stream socket API. The server is simply a file transfer server that forwards a file when a client requests for the same. However, the policy varies depending on how it handle requests from multiple clients.

So you need to implement two different variations of the file transfer server. One is *Bandwidth Division Multiple Access File Transfer (BDMAFT)* sever and another one is *Time Division Multiple Access File Transfer (TDMAFT)* server.

Whenever the BDMAFT server receives requests from multiple clients, it handles the requests in parallel. That is, for every client request, it simply fork a child process and the child process handles the file transfer. Therefore, if there are three child process running in parallel, then the total bandwidth is subdivided among those three child processes.

For TDMAFT server, multiple client file download requests are handled in a sequential round robin fashion. Whenever a new request comes from a client, the request is kept in a write queue. Assume that there are three such requests in the write queue. Then the requests are processed using a sequential round robin fashion with a predetermined time slice. This can be implemented using a `select()` system call.

Server Requirements (BDMAFT):

- Multiple connections are handled using `fork()` system call.
- Whenever a file download request comes from a client, the server prints the following in the terminal
  - Client ID
  - File Name
  - File size
  - MD5 checksum of the file (this can be done using *`md5sum`* system command – check *`man md5sum`*)
  - Start time of the file download (check `clock()` function in `time.h`)
  - End time of the file download

Server Requirements (TDMAFT):

- Multiple connections are handled using `select()` system call.
- Whenever a file download request comes from a client, the server prints the following in the terminal
  - Client ID
  - File Name
  - File size
  - MD5 checksum of the file (this can be done using *`md5sum`* system command – check *`man md5sum`*)
  - Start time of the file download (check `clock()` function in `time.h`)
  - End time of the file download
- The time slice for round robin scheduling is 4 seconds.

Client Requirements:

- The client requests for a file from the server.
- There should be single client for both BDMAFT server and TDMAFT server.
- The clients need to print the following in their respective terminals
  - Client ID
  - File name
  - File size

    - Download progress (10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100%)
    - MD5 checksum of the file after the download is over.

Test Cases:

- There should be at-least three clients running in parallel.
- Use buffer size = 1024 bytes
- For every write in the socket, wait for 2s before the next write. For this add a `sleep(2)` after the `write()` function call. This is required to make the output tractable, otherwise the file transfer will be very fast.
- You'll be given three test files – test1.pdf, test2.pdf and test3.pdf.