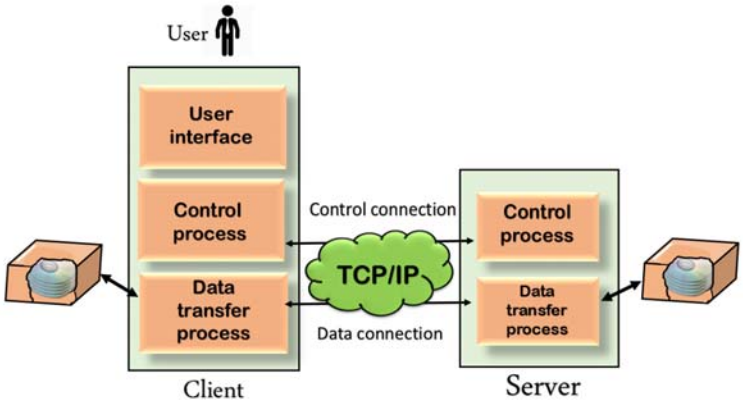
IT305 LAB – 06 Socket Programming II

Implement a fast multi-threaded File Transfer Program

FTP is an Internet standard protocol for transferring files between network nodes. RFC-959 defines the FTP protocol specifications. At the very basic level, FTP specifies setting up of two TCP connections – a *control* connection and a *data* connection. *Control* connection is used to pass control information (commands, responses, errors etc.) and the *Data* connection is used to transfer data.



A fast implementation of the protocol generally relies on the following:

1. Used multi-threading to handle each client and thematically different tasks separately. Threads can be pre-created and managed to improve the response time even further. It is assumed that you have worked with multi-threaded application design and implementation. You may review the POSIX pthread library functions **here**.

2. Multiple data channels may be set up for transferring data chunks in parallel. In many situations, this improves the performance.

3. Small file transfers (threads handling these) may be prioritized to improve the response time for small file transfers.

4. Some of the file chunks for each file can be cached in the memory and the threads responsible for secondary storage to main memory data chunk transfer can be optimized to further improve the response time.

Exercise

PART-I

1. Implement a TCP sockets based file transfer protocol that (a) creates a control and a data channel, (b) implement **DIR** and **GET** *filename* (with *response* and *error* handling) on the control channel, and (c) implement the data transfer logic on the data channel.
2. Prepare one of the 3 PCs (per group) to host the server, and the other two machines to run multiple clients each. Associate a folder with a number of files with the server.

3. Measure the time to transfer a large file from a client.
4. Increase the number of clients simultaneously transferring files from the server and measure the completion time as a function of *number of clients*.

PART-II

1. Now implement the same protocol using the multithreading and enhancements mentioned in the section above and measure the completion time as a function of number of clients.

Submissions

1. Describe the design of your multi-threaded application with the relevant implementation details. Include the server and client set up and completion time measurements.
2. Source codes for the server and the client.