# COMP 7005

# Assignment 1 Design Document

**By:** Derek Wong

**Instructor:** Aman Abdulla

**Due:** 9AM on October 7th, 2020

## Overview

Implement a simple client-server application using the TCP/IP protocol suite that comes with the Berkeley Sockets API. The server will listen for connections from clients, once established it will respond to file transfer commands from the clients. The client will initiate a connection to remote server and issue commands to server to either send a file or receive a file. Server information as well as commands will be supplied as command line arguments to the client. Two commands, **GET** or **SEND**, are specified to either request a named file from the server or to put a specified file to a remote server.

## Requirements

The application has the following constraints:

1. Any language of choice
2. Server will listen on and use port 7005, the control channel, for connection and requests
3. When GET/SEND commands are received, the server will initiate a data connection from port 7006, the data channel, to port X (i.e., 4612) on the client machine to either send or receive a file

## Design

- The client/server application will be written in C
- Client socket will bind to port 4612 when facilitating data transfer
- File transferred will be simple text files

### Client

1. Client connects to server using command line specified server information
2. Client sends request (e.g., GET/SEND) specified from command line to server on control channel
3. Client either send a file to the server (send.txt) or get a file from the server (get.txt)

### Server

1. Server listens until connections are made from clients with a request
2. Server either sends a file to client (get.txt) or gets a file from the client (send.txt)

## TCP/IP Client Server Model

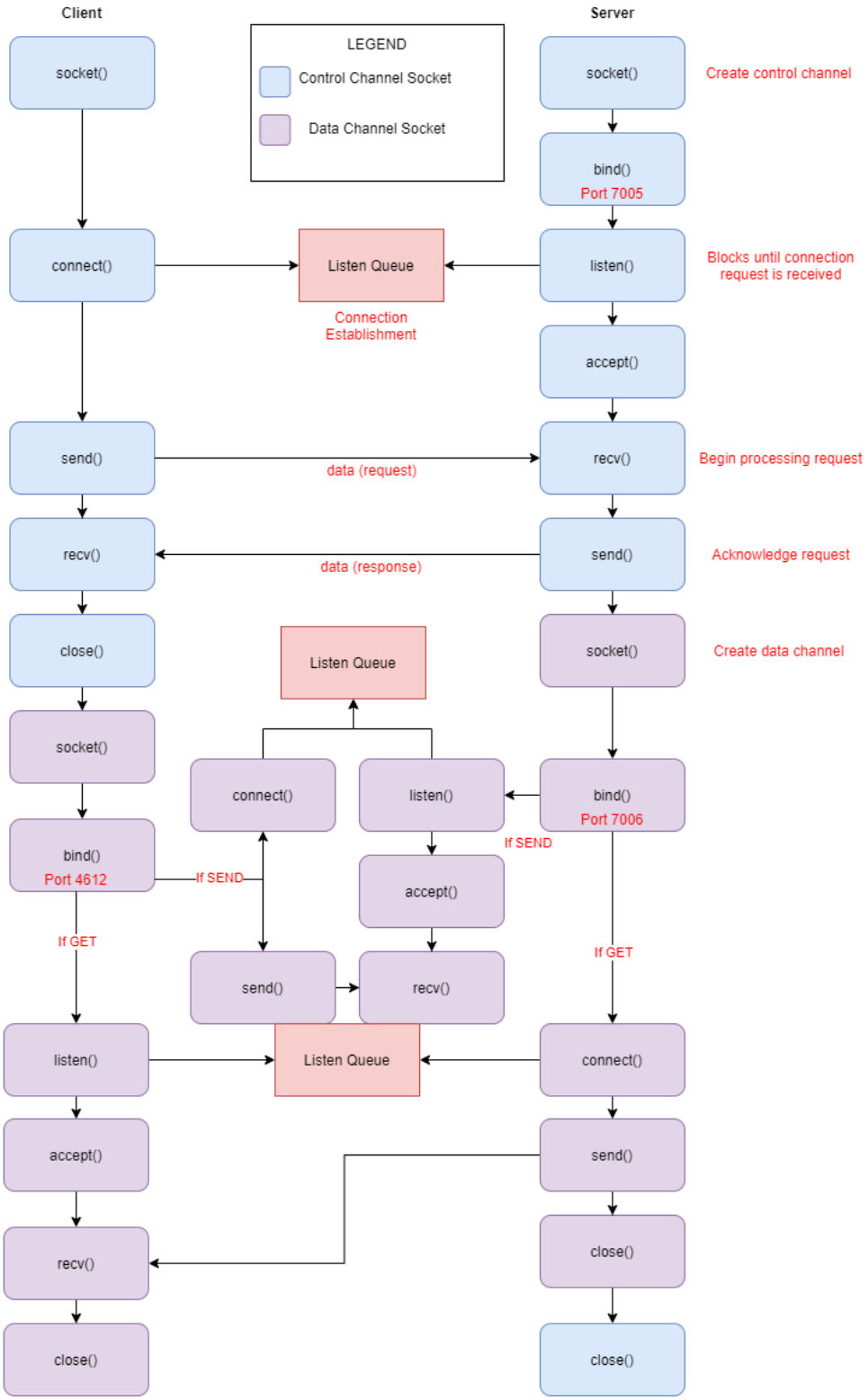
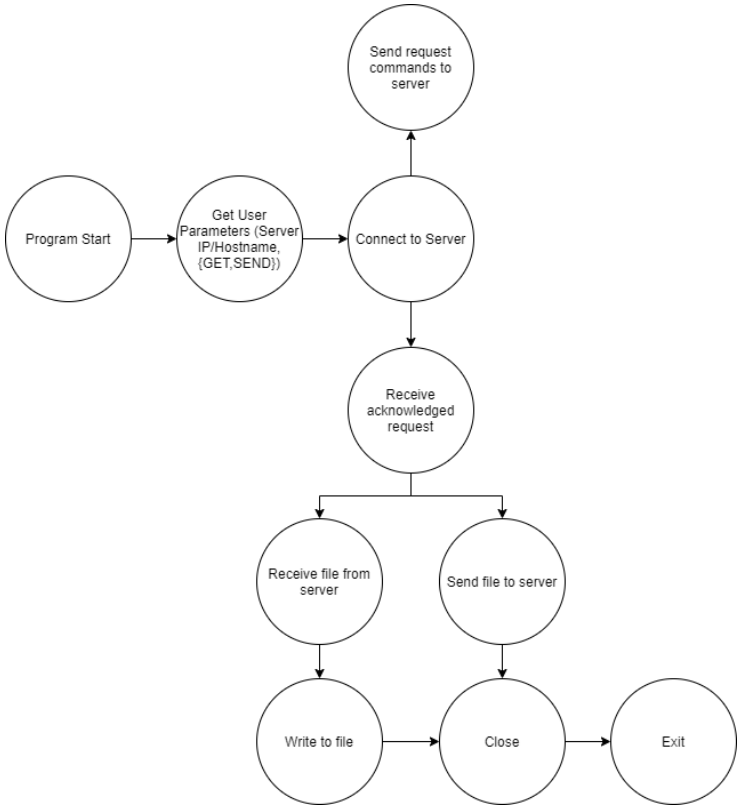


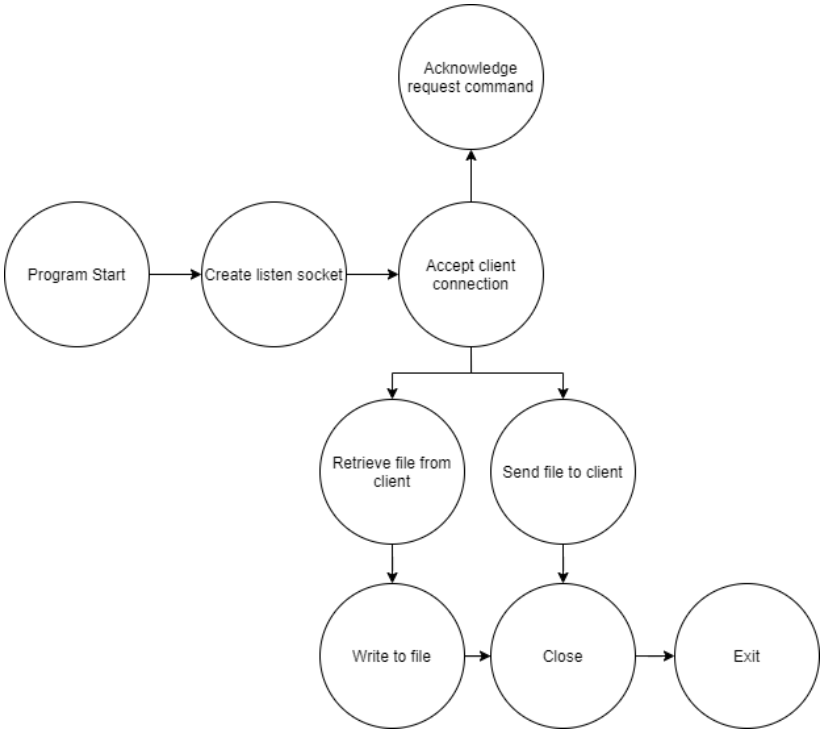

**Figure 1 –** TCP/IP Client Server model showing the socket functions flow in the client/server application.

# State Diagram

## Client




## Server



## Pseudocode

```
////CLIENT////

get_user_param:
        get server ip from user
        get command request from user


connect_to_server:
        create client control channel socket
        set socket options
        bind address to socket
        connect to server


send_request:
        send client request to server
        receive acknowledged request
        close client control channel socket


process_request:
        create socket
        set socket options
        bind address to socket
        if request is GET:
                listen for server connection
                accept server connection
                receive file data
                write to file
                close server data channel socket
                close client data channel socket
        if request is SEND:
                connect to server data channel socket
                open file
                send file data
                close client data channel socket


exit application
```

```
////SERVER////

create_listen_socket:
        create server control channel socket
        bind address to socket
        listen for client connection


while(server is running):
        accept client connection
        receive request from client
        send acknowledged request to client
        go to process_request


process_request:
        create server data channel socket
        bind data channel socket
        if request is GET:
                connect to client data channel socket
                oepn file
                send file data
                close server data channel socket
        if request is SEND:
                listen for client connection
                accept client connection
                receive file data
                write to file
                close client data channel socket
                close server data channel socket


exit application
```