Carleton UNIVERSITY
Canada's Capital University

SYSC5709S

Software Development with C

Server/Client Implementation Using TCP Sockets– Developer Manual

Submitted to:
Dr. Cristina Ruiz Martin

Submitted by: Group A

Hitin Sarin-101134217 (Github id- Hitinsarin3091)
Jaspreet Kaur-101148265 (Github id- jaskaur7)
Prerit Sikerwal-101128282 (Github id- preritsingh96)
Saksham Mal-101132131 (Github id- sakky27)

Repository link: https://github.com/jaskaur7/Group_A_SERVER-CLIENT_USING_TCP.git

Problem Statement:

SERVER-CLIENT USING TCP: Socket Programming provides a link for two-way communication between different programs running on the network. It consists of a port number, IP address which helps in identifying where the data needs to be sent to and thus establishing a connection between client and server. The main purpose of our project is to design and implement a web server and client system that can be used to communicate by the means of transfer of data in the form of string. The communication is established using TCP sockets. The server will echo the message or instructions that client needs. A connection between for a single client and server will be established.

Additionally, a multithreading concept will be implemented in case there is more than one client that is more than one requests to the server. Thus, whenever this occurs there will be different threads created (depending on the queue size specified) to process different requests at a time and thus there will be no waiting time and would be more responsive.

File and Folder Structure Organization

1. bin : Bin folder consists of all the executable code for the SERVER-CLIENT application consisting of:

- client
- serverMulti
- serverSingle
- test_client
- test_server

2. build : build folder consists of all the object files:

- client.o
- command.o
- queue.o
- serverMulti.o
- serverSingle.o
- test_client.o
- test_server.o

3. doc : All the documents regarding the project is stored in the doc folder and the folder consists of following documentations:

- DeveloperManual and Software design

• user_manual
• installation_file

4. include : include folder consists of all header files of the project**:**

• client.h
• common.h
• queue.h
• server_single.h
• server_multi.h

5. src : src folder consists of source code for the SERVER-CLIENT application consisting of:

5.1 server-singlethread
    • server.c
5.2 server-mulithread
    • server.c
5.3 client
    • client.c

6. lib: It contains the file for implementing the queue.

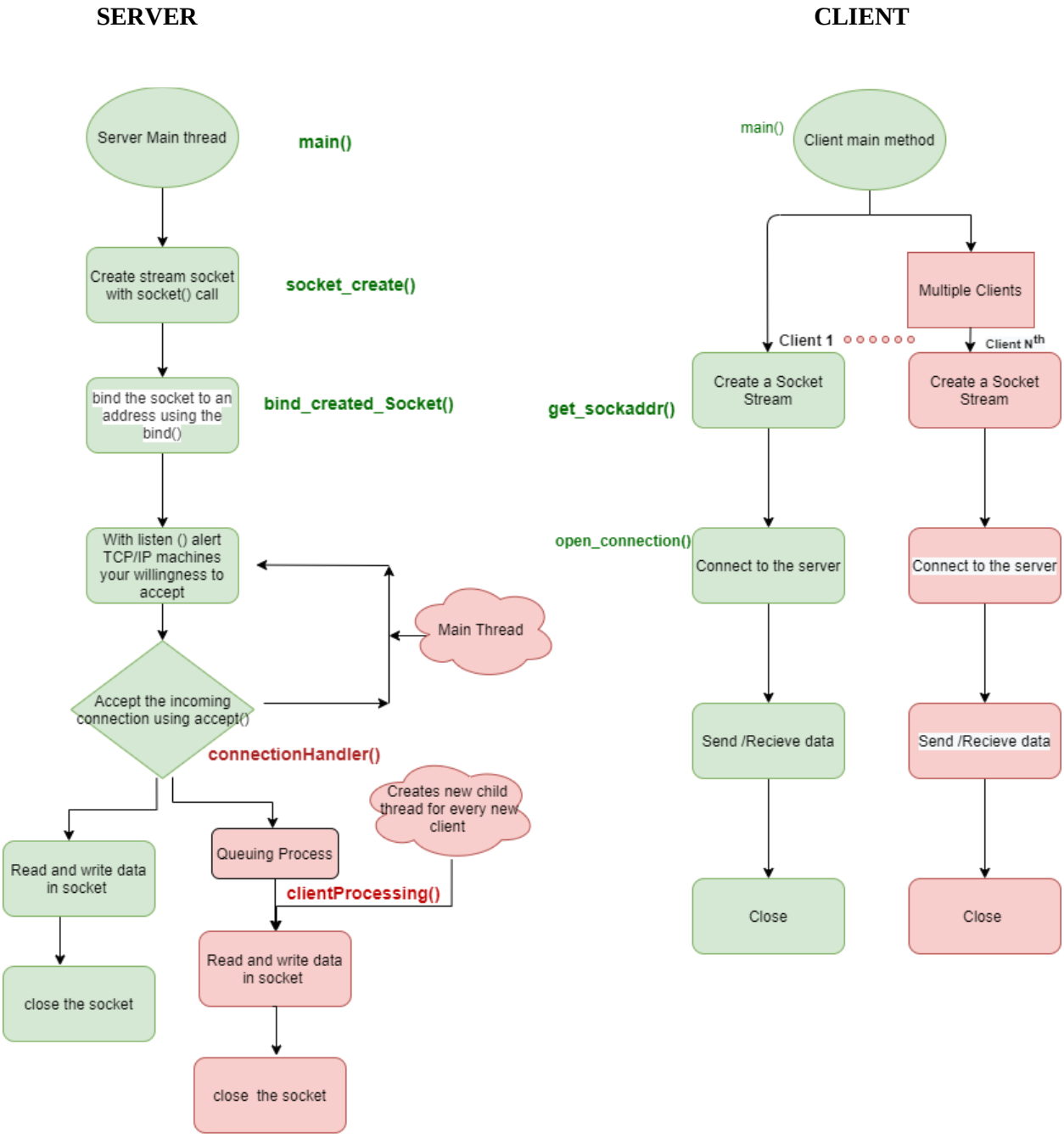
• queue.c

7. test : test folder consists of the source folder for test scenarios regarding the project :

7.1 src
    • test_client.c
    • test_server.c

Software Design:

Project was completed in two releases with each release consisting of different functions as shown below. GREEN color is for release one and RED color for release two.

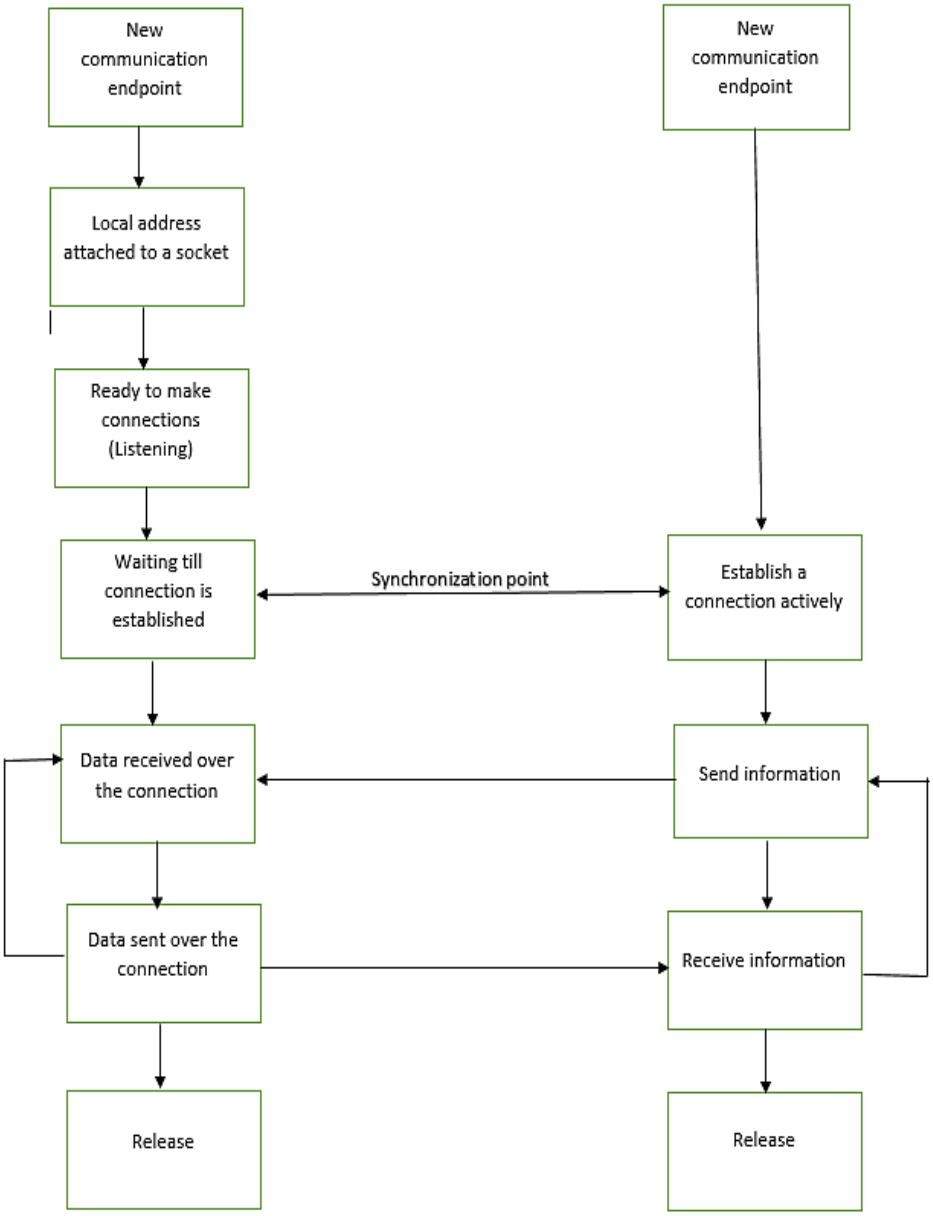


Flow Diagrams:

Release 1 Implementation of TCP protocol

In first release our main aim would be to implement TCP protocol which will provide us a reliable platform to transfer data strings between a server and a client. Since we know that TCP is an end to end protocols, we will also use Sockets here which are helpful as file descriptors for establishing network communication.

Basic flow diagram for the complete process of Release 1.



Server

Functions:

1. **main()**

In this function all the functions related to the working of server end such as creating & binding socket, listening for client request, accepting the request and transfer of data string will be called.

Input parameter : int argc, char **argv []

Return value expected: 0

2. **socket_create ()**

The function is used to create a socket and once it is created it has to specify the domain and type of socket. The function returns an entry as small integer into file descriptor table and it

Input parameter : af_net , sock_strem

Return value expected: -1 if it could not create socket

3. **bind_created_socket()**

Using this function we will bind the socket to a specific address, but the socket will be still in closed state. If the binding gets failed it will show an error message. The hsocket will be created by the function SocketCreate() with AF_INET protocol address and SOCK_STREAM, which is a type of network socket or interprocess communication socket that provides connection-oriented services without any limit and organized mechanisms for opening and closing of the connection.

Input parameters: hsocket, port number

Return value expected: 1

After this server goes in listening phase actively looking for client request.

Client

Functions:

1. **main()**

In this function all the functions related to the working of client end such as creating & binding socket, connection to remote server and transfer of data string will be called.

Input parameter : int argc, char *argv []

Return value expected: 0

2. **get_sockaddr()**

The function get_sockaddr converts the server's address and port into a form usable to create a socket.

Input parameter : struct addrinfo hints, const char* hostname, const char *port
Return value expected: 0

3. **open_connection()**

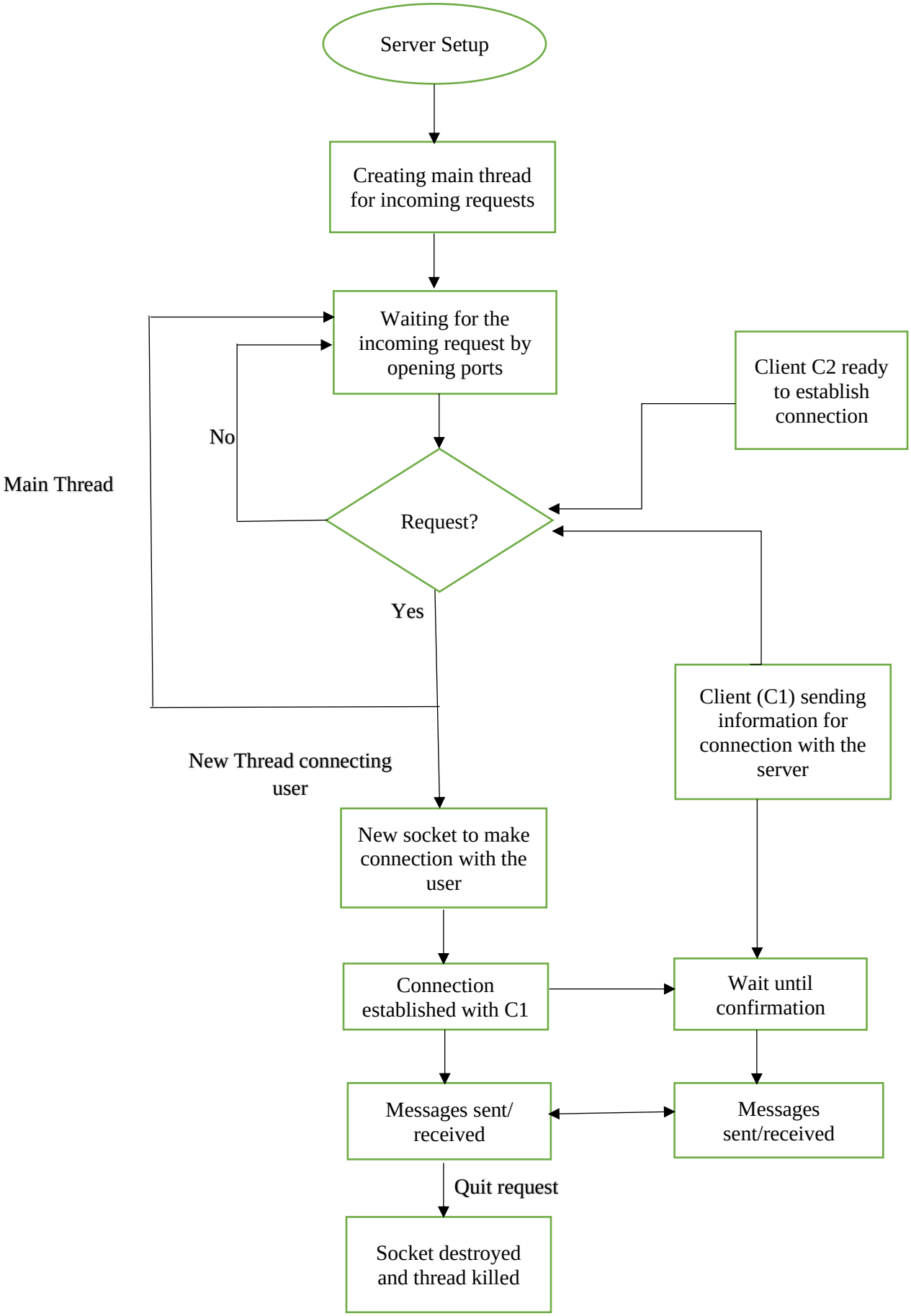
This function will be used to establish a connection with the remote server.

Input parameter : int sockfd ,addr_list

Return value expected: 1

Release 2 Multi-Threading

In release 1, we will be establishing connection between only one server and one client. But this could be problematic in many situations when there are several clients requesting data transfer. In such case we need to modify our Server that is implementation of multithreaded server so that it can handle multiple client requests at a time. In multithreading, every time when a new client requests for data exchange. Server creates different child threads depending on the size specified and allocate it to the client thus making sure that different clients are served together and thus there is no waiting time and better performance.


Server Setup
Creating main thread for incoming requests
Waiting for the incoming request by opening ports
Client C2 ready to establish connection
Main Thread
No
Request?
Yes
Client (C1) sending information for connection with the server
New Thread connecting user
New socket to make connection with the user
Connection established with C1
Wait until confirmation
Messages sent/ received
Messages sent/received
Quit request
Socket destroyed and thread killed

Server: In this server will have extended functionalities to solve the purpose of multithreading concept where a server can serve multiple clients at a time.

Functions:

1. **main()**

   In this function all the functions related to the working of server end such as creating & binding socket, listening for client request, accepting the request, handling the request,creating new threads for each request and transfer of data string will be called.

   Input parameter : int argc, char **argv []

   Return value expected: 0

2. **connectionHandler()**

   This function plays a key role in multi-threading process. This function handles connection for each client. Each thread is bound to work using its own handler. Once a new handler is created, the thread that generates it uses it to send messages to the message queue, and execute them when they come out of the message queue.

   Return value expected: void

3. **clientProcessing()**

   The server receives data from the client, read the string and gives it back to the client and then closes the connection.

   Input parameter : connfd

   Return: integer value for client disconnected and recieving failure

4. **queue_add(), getQueue()**

   The functions which locks the queue, fetches if the client connection is present in the queue in order to establish the connection and if got the element it passes it back and unlocks the queue again.

   Input parameter : value (The integer value of pthread_mutex must be initialised)

   Return: integer value for waiting connection error and adding queue

5. **Queue related functions**
   struct Queue* createQueue() ,int isFull(),int isEmpty(),enqueue(),dequeue()

   These are the functions used for array implementation of queue.It includes function to create a queue of given capacity, functions used for adding and deletings items from queue.