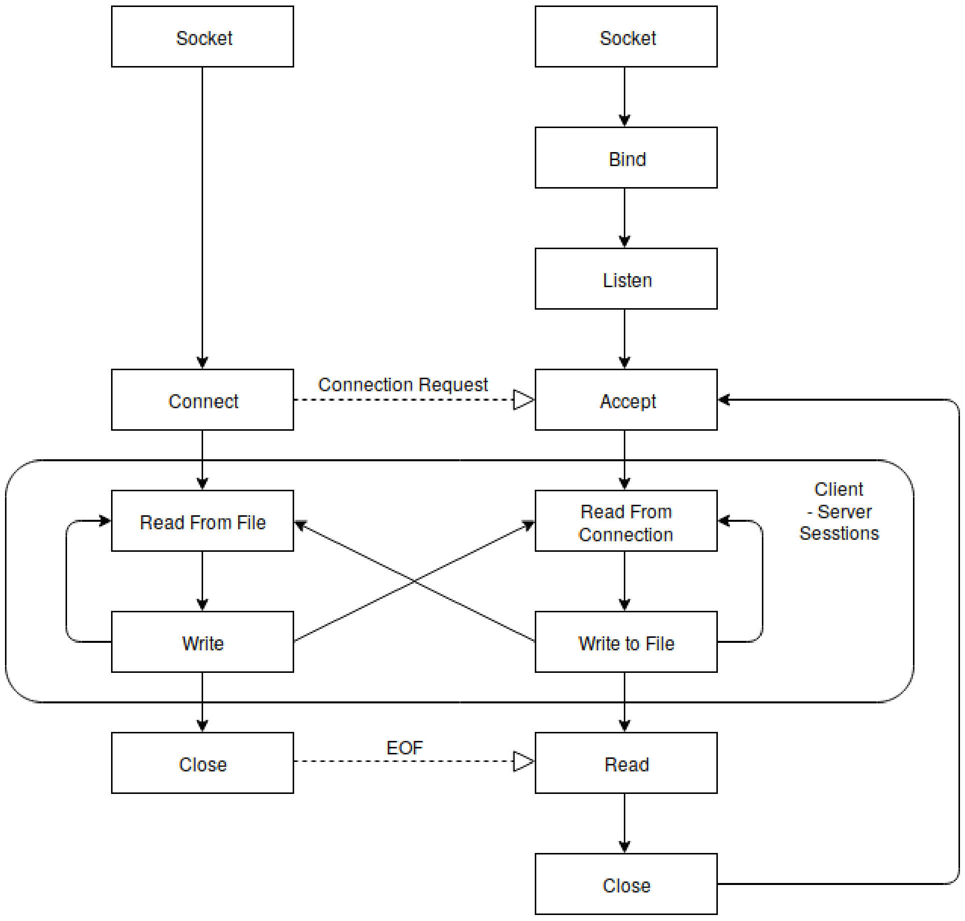
Report for the practical work 1

Group 2

February 2019

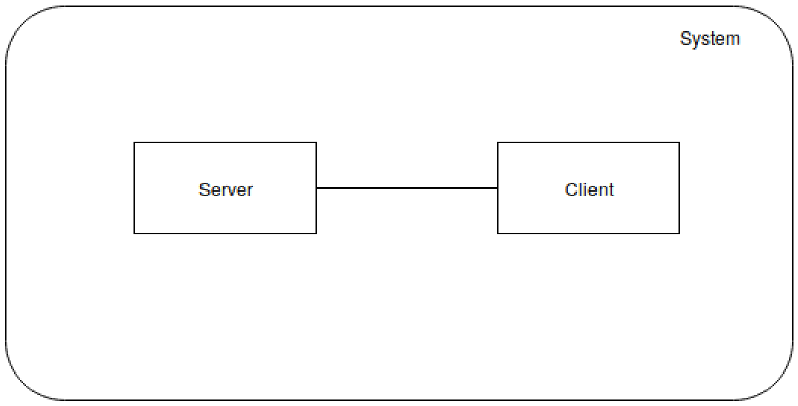
Protocol:

We design our protocol based on the client server chat system using tcp socket with some modification over the server / client session we create a buffer of 8192 bit in the server to be the temporary place for the file to be stored and then assemble it into the whole file



System:

Our system is a very simple system of just 2 machine connect to each other through a cable. one of them act as a server to receive the file and the other machine act as the client to send the file, the system can expand to more client to connect and send file to the server.



How:

We implement the file transfer using the raw socket on the language C/C++. First create a socket to connect two machine with the default function of socket socket(). The number of port using to connect is chosen by the user using the function htons() but the server and the client have to have the same port in order to connect as long as it not the same as the port use for other purposes. The client use function connect() to establish the connection to the server. On the server their is a function accept() use to accept the connection from the client.
After this the two machine is connected and ready to send and recieve file.

In the client side next we open the file using the function open() but in the client we set the permission to read the file only.
Inside the while function, we read the file into the buffer then send the file through the connection, because depend on the size of the file, if the file is bigger than the size of the buffer then the file will be split into many file to send.

Code: (on client side)

```
#include <stdio.h >
#include <stdlib.h >
#include <string.h >
#include <sys/types.h >
#include <sys/socket.h >
#include <netdb.h >
#include <fcntl.h >

int main(int argc, char* argv[]) {
    int so;
    struct sockaddr_in ad;
    char buff[8192];
    socklen_t ad_length = sizeof(ad);
    struct hostent *hep;

    int serv = socket(AF_INET, SOCK_STREAM, 0);

    hep = gethostbyname(argv[1]);
    memset(&ad, 0, sizeof(ad));
    ad.sin_family = AF_INET;
    ad.sin_addr = *(struct in_addr *)hep->h_addr_list[0];
    ad.sin_port = htons(9001);

    connect(serv, (struct sockaddr *)&ad, ad_length);
    int fileToBeSended = open("test.jpg", O_RDONLY);

    while(1) {
            int n = read(fileToBeSended, buff, sizeof(buff));
            if (n <=0) break;
            write(serv, buff, n);
    }
    close(fileToBeSended);
    return 0;
}
```

On the server side, after connecting to the client the server will fork the process to start recieving file from the multiple clients.
Function open() is used to create a file with the name that we want with the option to create the file and to write to it.
Then create a integer type n to use as a placeholder to accept the buffer of the file from the client, after the n is 0 mean that all part of the file has been sent then we have a function write() to write to the file name that we given above.
For some system the file when sent from the client to the server may have no permission at all so no one can view it, so we use the command system() to execute the linux command chmod to change the file permission in order to be able to read the file.
After all of that then close the connection.

Code: (on server side)

```
#include <stdio.h >
#include <stdlib.h >
#include <string.h >
#include <sys/types.h >
#include <sys/socket.h >
#include <netdb.h >
#include <fcntl.h >

int main() {
    int ss, cli, pid;
    struct sockaddr_in ad;
    socklen_t ad_length = sizeof(ad);
    char buff[8192];

    // create the socket
    ss = socket(AF_INET, SOCK_STREAM, 0);

    // bind the socket to port 9001
    memset(&ad, 0, sizeof(ad));
    ad.sin_family = AF_INET;
    ad.sin_addr.s_addr = INADDR_ANY;
    ad.sin_port = htons(9001);
    bind(ss, (struct sockaddr *)&ad, ad_length);

    // then listen
    listen(ss, 0);
        while (1) {
    // an incoming connectionint n
    cli = accept(ss, (struct sockaddr *)&ad, &ad_length);

    pid = fork();
    if (pid == 0) {
```

```
    // I’m the son, I’ll serve this client
    printf(”client connected \n”);
    int fileSended = open(”fileReceived.jpg”, O_CREAT — O_WRONLY);
    while (1) {
    int n = read(cli, buff, sizeof(buff));
    printf(”n is :%d\n”, n);
    if (n <= 0) {
    system(”chmod 755 fileReceived.jpg”);
    break;
    }
    write(fileSended, buff, n);
    }
    return 0;
    }
    else {
    // I’m the father, continue the loop to accept more clients
    continue;
    }
    }
    // disconnect
    close(cli);
    return 0;
  }
```

Who does what:
- Duc does the code for the connection of the 2 machine.
- Nam write the code for the server side.
- Huy write the code for the client side.
- Trung write the document.