CYBER MANAGEMENT SYSTEM

CYBER MANAGEMENT SYSTEM

A SYSTEM DESIGNED TO MANAGE AN ORGANIZATION'S CYBERSECURITY

AIMS TO ENSURE COMPREHENSIVE AND CONTINUOUS PROTECTION OF AN ORGANIZATION'S NETWORK AND DATA

NEU "FICTIONAL STORY" WITH CYBER MANAGEMENT SYSTEM

NEU HAS MORE THAN 10 CAMPUSES.

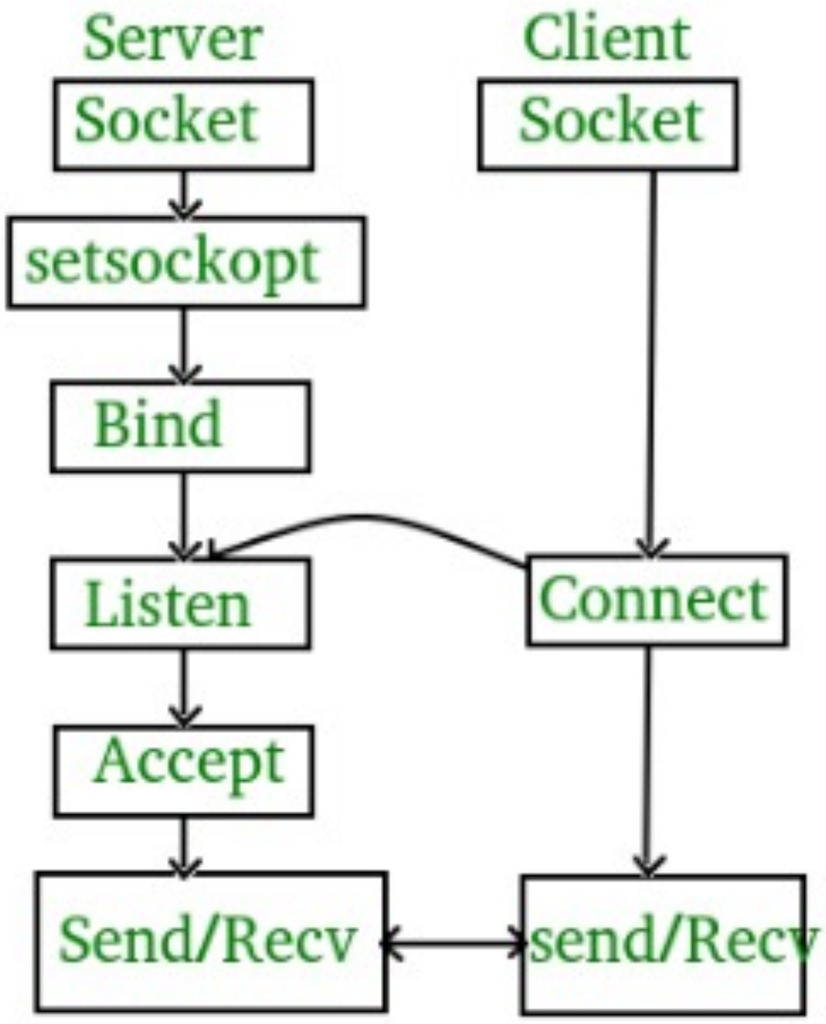
SUPPOSE THAT EACH CAMPUS HAS AN ADMIN TAKING CARE OF CAMPUS CONFIDENTIAL INFORMATION SUCH AS STUDENT INFO, TUITIONS AND SO ON.

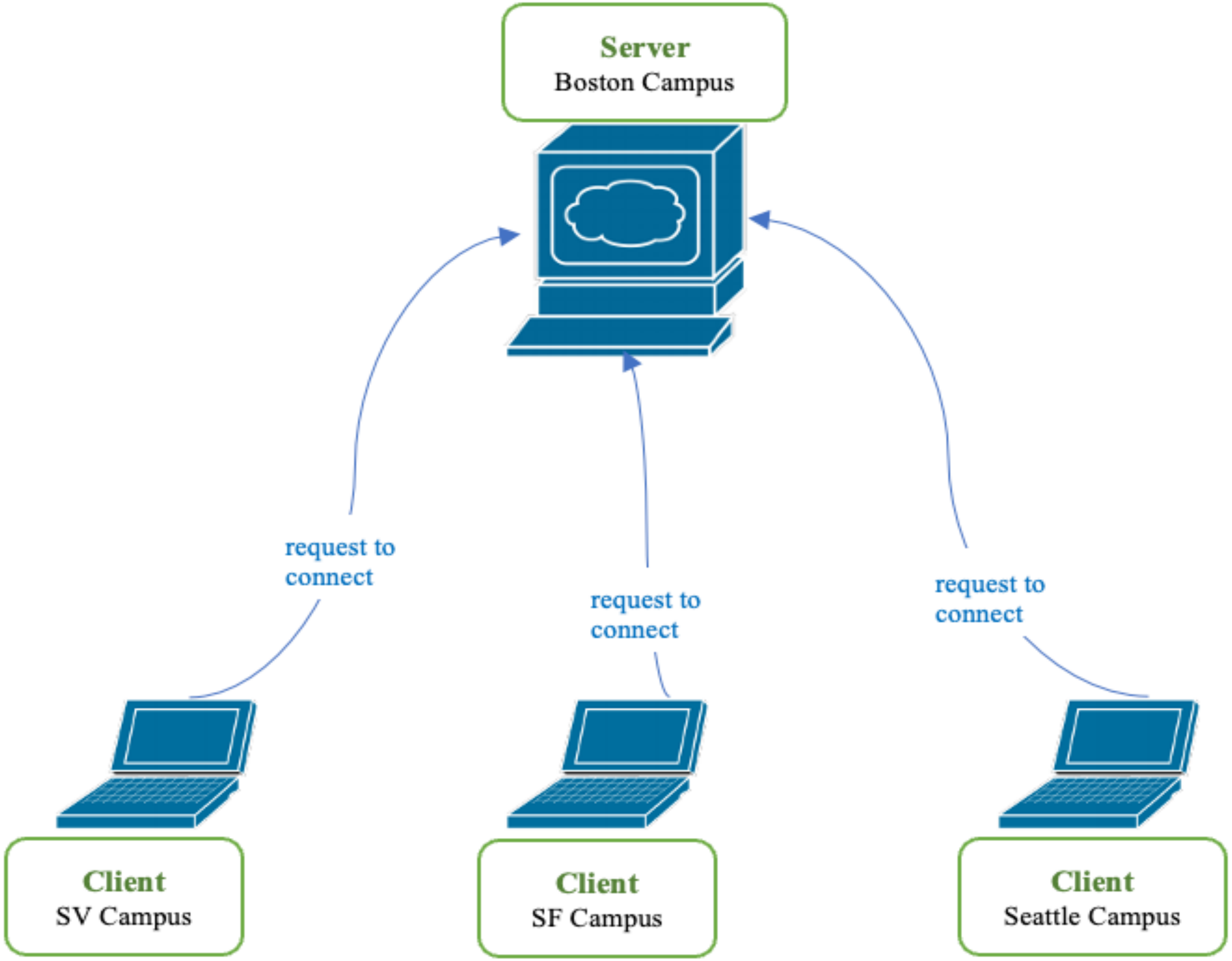
FROM TIME TO TIME, THE ADMIN NEEDS TO SHARE THE CONFIDENTIAL INFORMATION WITH THE MAIN BOSTON CAMPUS.

EMAIL SERVICE IS CONSIDERED NOT SAFE SOLUTION TO TRANSFER CONFIDENTIAL NEU INTERNAL DATA.

WHAT TO DO?

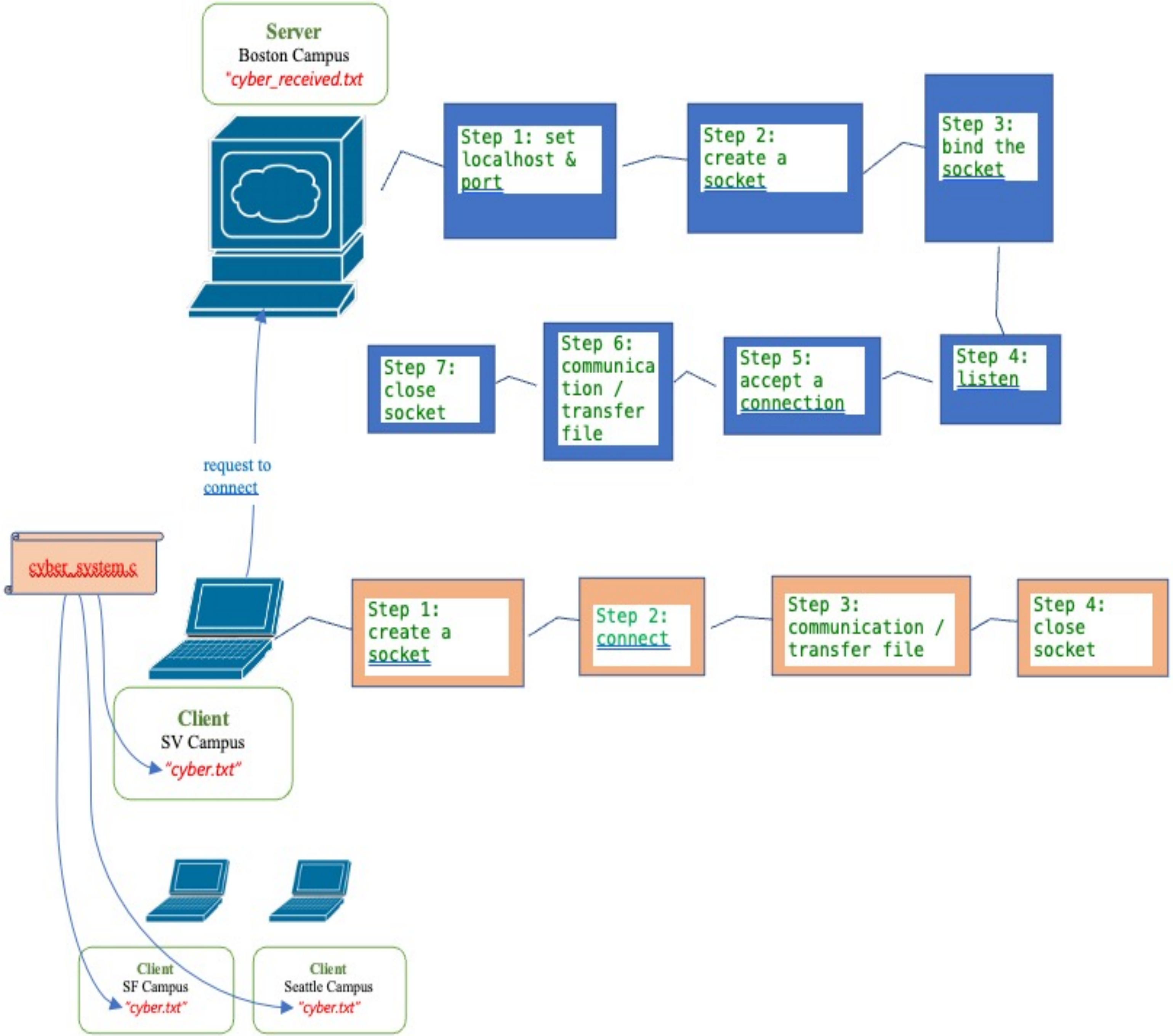
Introduction to Socket Programming





THE SERVER CREATES A SOCKET AND BINDS IT TO A PORT TO LISTEN FOR INCOMING CONNECTIONS FROM CLIENTS.

ON THE OTHER HAND, THE CLIENT SIDE CREATES A SOCKET AND CONNECTS IT TO THE SERVER'S IP ADDRESS AND PORT.



Server
Boston Campus
"cyber_received.txt

Step 1: set localhost & port

Step 2: create a socket

Step 3: bind the socket

Step 4: listen

Step 5: accept a connection

Step 6: communication / transfer file

Step 7: close socket

request to connect

cyber_system.c

Step 1: create a socket

Step 2: connect

Step 3: communication / transfer file

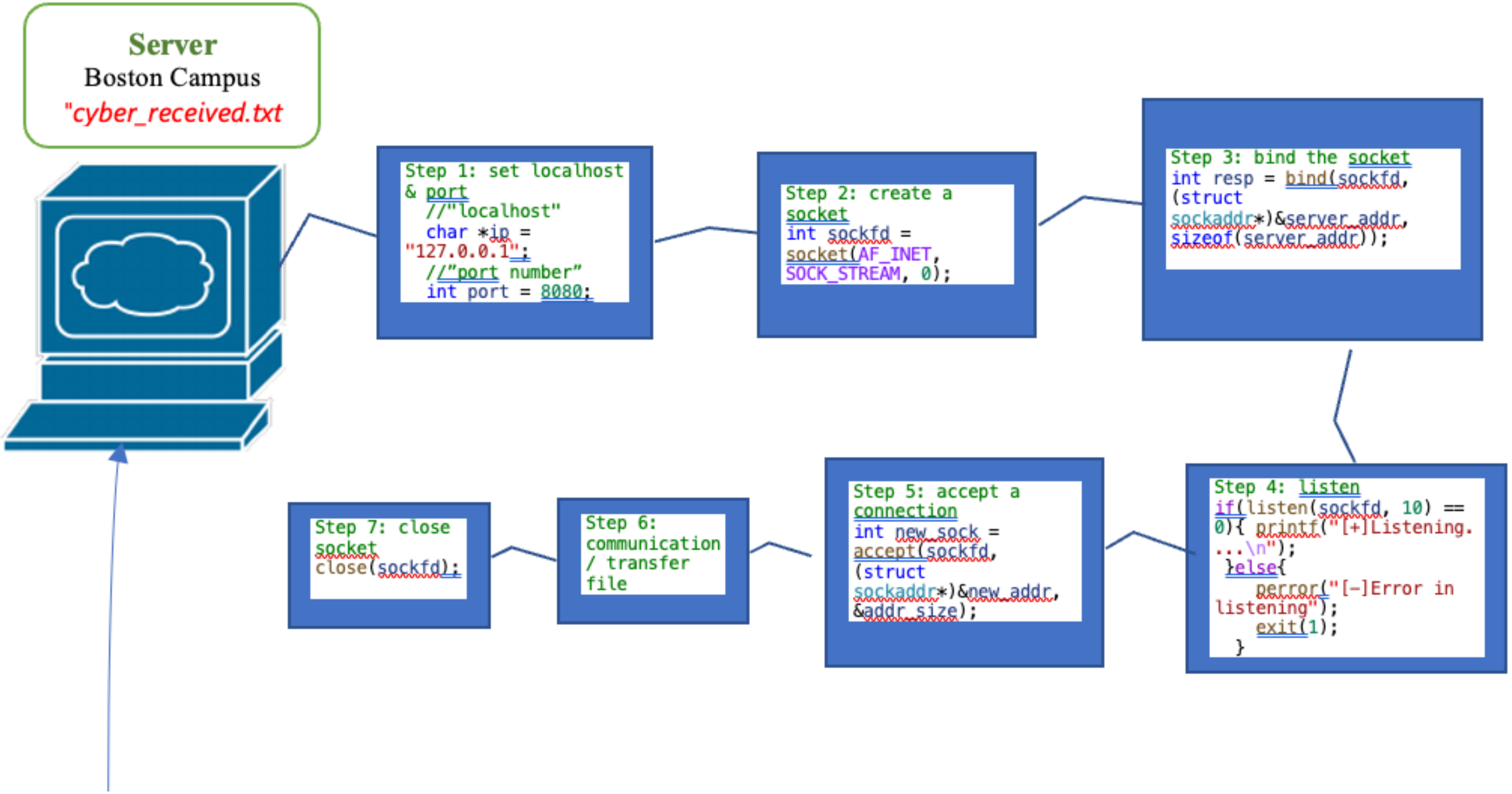
Step 4: close socket

Client
SV Campus
"cyber.txt"

Client
SF Campus
"cyber.txt"

Client
Seattle Campus
"cyber.txt"

DEMO:
NEU INTERNAL DATA TRANSMISSION PROCESS USING SOCKET PROGRAMMING



```
Step 1: set localhost
& port
  //"localhost"
  char *ip =
"127.0.0.1";
  //"port number"
  int port = 8080;
```

```
Step 2: create a
socket
int sockfd =
socket(AF_INET,
SOCK_STREAM, 0);
```

```
Step 3: bind the socket
int resp = bind(sockfd,
(struct
sockaddr*)&server_addr,
sizeof(server_addr));
```

```
Step 4: listen
if(listen(sockfd, 10) ==
0){ printf("[+]Listening.
...\n");
 }else{
    perror("[-]Error in
listening");
    exit(1);
  }
```

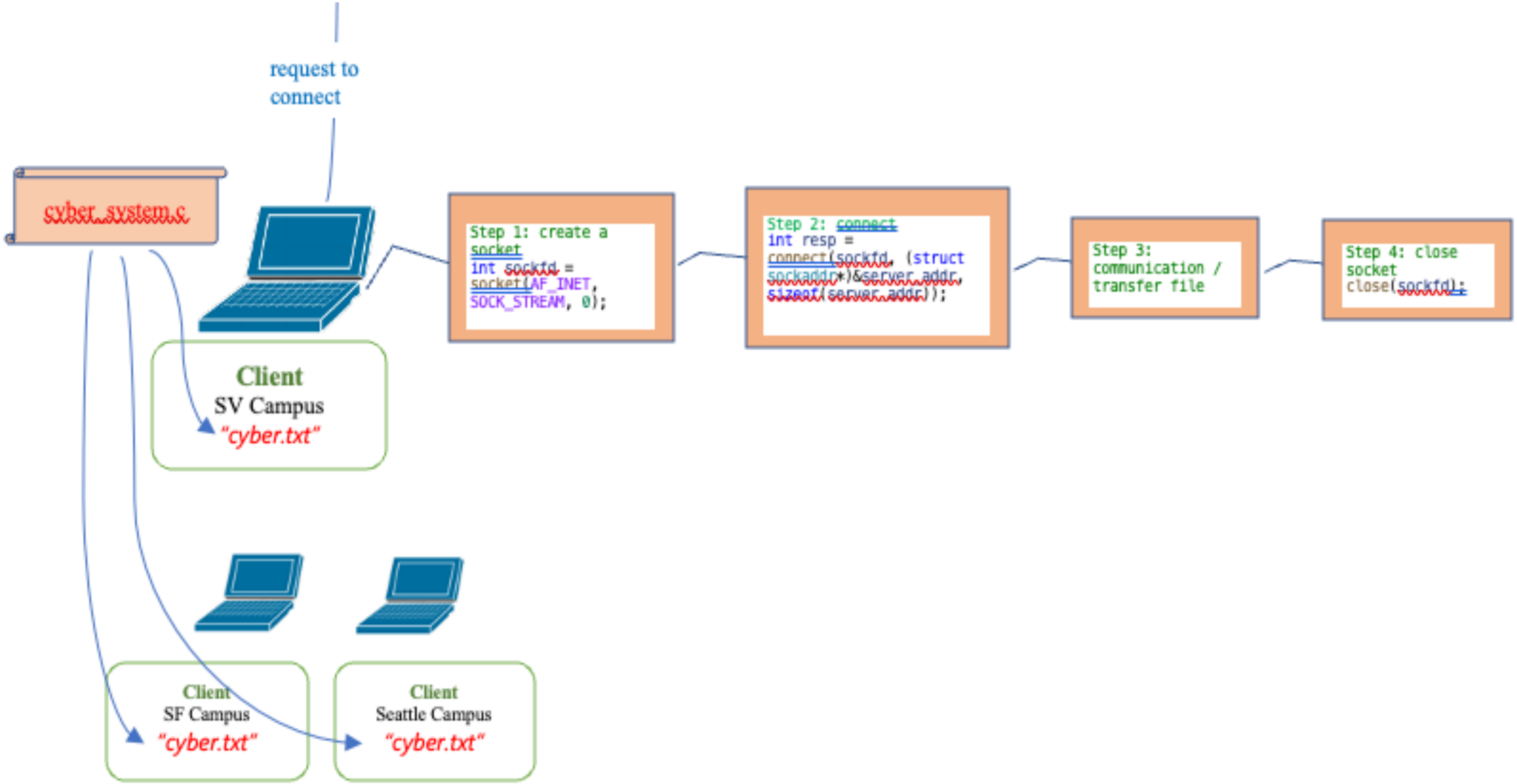
```
Step 5: accept a
connection
int new_sock =
accept(sockfd,
(struct
sockaddr*)&new_addr,
&addr_size);
```

```
Step 6:
communication
/ transfer
file
```

```
Step 7: close
socket
close(sockfd);
```

DETAILS ON

SERVER.C FILE



```
Step 1: create a
socket
int sockfd =
socket(AF_INET,
SOCK_STREAM, 0);
```

```
Step 2: connect
int resp =
connect(sockfd, (struct
sockaddr*)&server_addr,
sizeof(server_addr));
```

```
Step 3:
communication /
transfer file
```

```
Step 4: close
socket
close(sockfd);
```

DETAILS ON

CLIENT.C FILE

ONCE SERVER AND CLIENT CONNECTS,

WHAT HAPPENS NEXT…

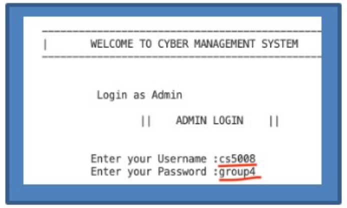
SCENARIO ONE: CLIENT(SV CAMPUS) SENDS "CYBER.TXT" FILE GENERATED BY THE "CYBER_SYSTEM.C" PROGRAM TO THE SERVER(BOSTON CAMPUS);

SERVER(BOSTON) SYNC THE TEXT FILE AND THEN CONVERTED THE RECEIVED DATA TO "CYBER_RECEIVED.TXT".

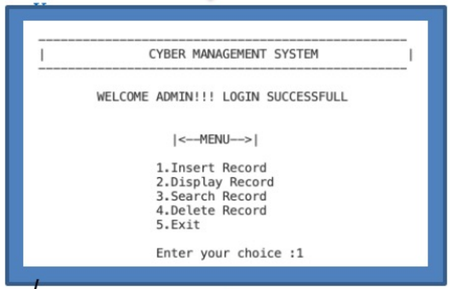
SCENARIO TWO: ON THE TOP OF SCENARIO ONE, ONCE CONNECTED, SERVER(BOSTON) FIRST "**CHATS**" WITH CLIENT(SV) SIDE A LITTLE BIT, BEFORE TRANSFERRING THE FILE.

DETAILS ON CYBER_SYSTEM.C FILE

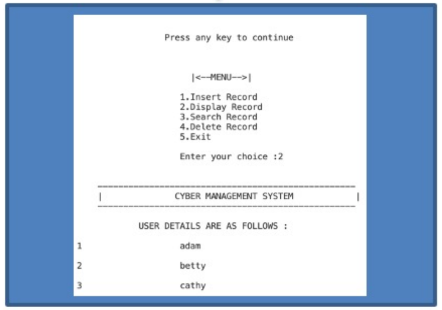
cyber_system.c



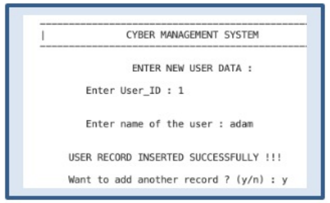
```
 ----------------------------------------------------------
|        WELCOME TO CYBER MANAGEMENT SYSTEM
 ----------------------------------------------------------


          Login as Admin

                 ||    ADMIN LOGIN    ||


        Enter your Username :cs5008
        Enter your Password :group4
```



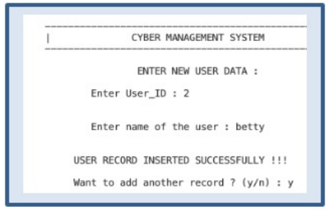
```
 ------------------------------------------------------------
|                 CYBER MANAGEMENT SYSTEM                   |
 ------------------------------------------------------------

          WELCOME ADMIN!!! LOGIN SUCCESSFULL


                    |<--MENU-->|

                  1.Insert Record
                  2.Display Record
                  3.Search Record
                  4.Delete Record
                  5.Exit

                  Enter your choice :1
```



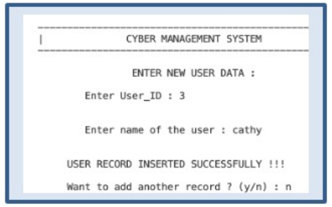
```
              Press any key to continue



                    |<--MENU-->|

                  1.Insert Record
                  2.Display Record
                  3.Search Record
                  4.Delete Record
                  5.Exit

                  Enter your choice :2


   -------------------------------------------------------
  |              CYBER MANAGEMENT SYSTEM                  |
   -------------------------------------------------------

            USER DETAILS ARE AS FOLLOWS :

1                 adam

2                 betty

3                 cathy
```



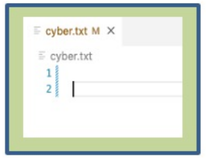
```
 ----------------------------------------------------------
|               CYBER MANAGEMENT SYSTEM
 ----------------------------------------------------------

                 ENTER NEW USER DATA :

        Enter User_ID : 1


        Enter name of the user : adam


     USER RECORD INSERTED SUCCESSFULLY !!!

     Want to add another record ? (y/n) : y
```

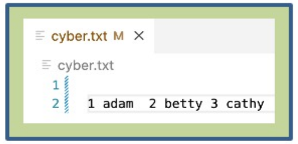


```
 ----------------------------------------------------------
|               CYBER MANAGEMENT SYSTEM
 ----------------------------------------------------------

                 ENTER NEW USER DATA :

        Enter User_ID : 2


        Enter name of the user : betty


     USER RECORD INSERTED SUCCESSFULLY !!!

     Want to add another record ? (y/n) : y
```



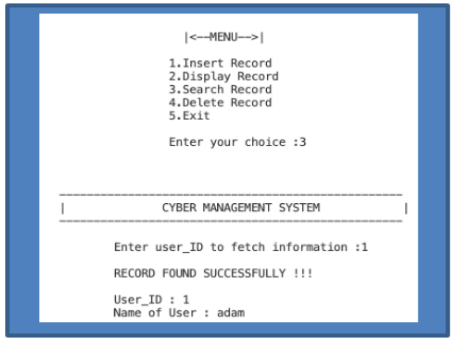
```
 ----------------------------------------------------------
|               CYBER MANAGEMENT SYSTEM
 ----------------------------------------------------------

                 ENTER NEW USER DATA :

        Enter User_ID : 3


        Enter name of the user : cathy


     USER RECORD INSERTED SUCCESSFULLY !!!

     Want to add another record ? (y/n) : n
```



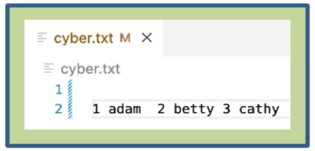


```
 1 adam   2 betty  3 cathy
```

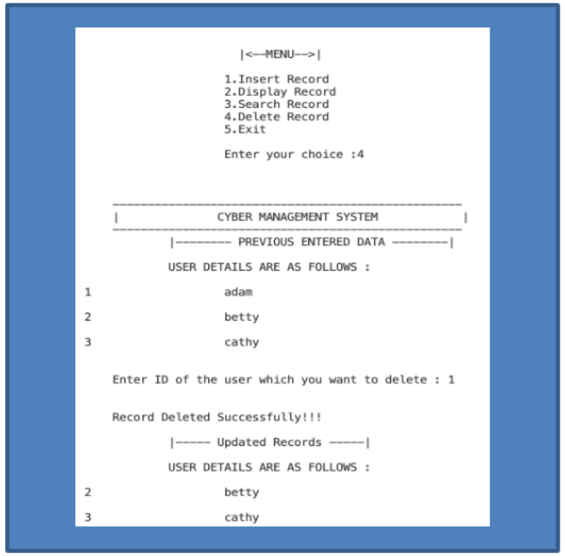
cyber_system.c



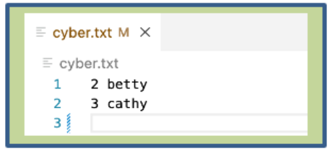
```
                    |<--MENU-->|

                  1.Insert Record
                  2.Display Record
                  3.Search Record
                  4.Delete Record
                  5.Exit

                  Enter your choice :3



   -----------------------------------------------------
  |               CYBER MANAGEMENT SYSTEM               |
   -----------------------------------------------------

          Enter user_ID to fetch information :1

          RECORD FOUND SUCCESSFULLY !!!

          User_ID : 1
          Name of User : adam
```



```

 1 adam   2 betty 3 cathy
```



```
                    |<--MENU-->|

                  1.Insert Record
                  2.Display Record
                  3.Search Record
                  4.Delete Record
                  5.Exit

                  Enter your choice :4



   -----------------------------------------------------
  |               CYBER MANAGEMENT SYSTEM               |
   -----------------------------------------------------
         |-------- PREVIOUS ENTERED DATA --------|

         USER DETAILS ARE AS FOLLOWS :

1                 adam

2                 betty

3                 cathy

   Enter ID of the user which you want to delete : 1


   Record Deleted Successfully!!!

         |----- Updated Records -----|

         USER DETAILS ARE AS FOLLOWS :

2                 betty

3                 cathy
```



```
2 betty
3 cathy

```

DETAILS ON

CYBER_SYSTEM.C FILE

DEMO

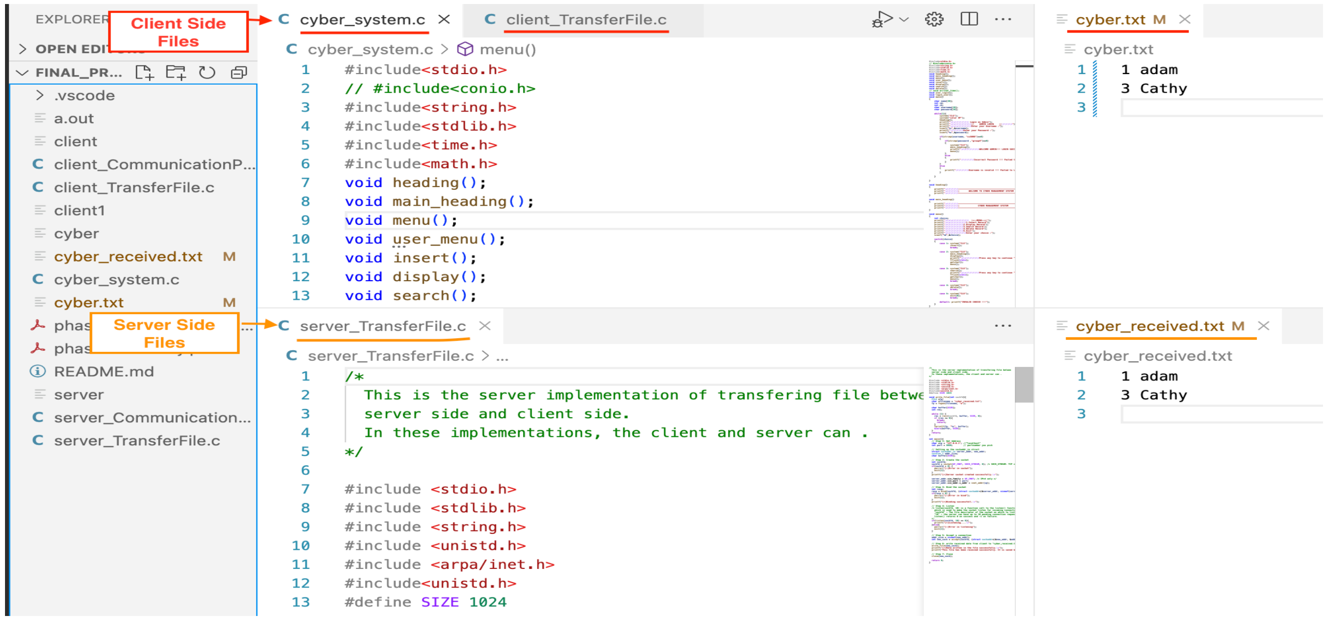
Overview of Scenario one:

CLIENT SIDE:

1. "CYBER_SYSTEM.C" FILE: FOR NEU SV CAMPUS ADMIN TO MANAGE STUDENT INFO, GENERATE "CYBER.TXT" FILE IN THE LOCAL.
2. "CLIENT_TRANSFERFILE.C" FILE: SOCKET PROGRAMMING FILE FOR CONNECTING TO SERVER AND THEN TRANSFER "CYBER.TXT" TO SERVER ONCE CONNECT.
3. "CYBER.TXT" FILE: STORES THE STUDENT DATA GENERATED BY RUNNING "CYBER_SYSTEM.C" FILE.

SERVER SIDE:

1. "SERVER_TRANSFER.C" FILE: SOCKET PROGRAMMING FILE FOR CONNECTING NEU SV CAMPUS SIDE AND THEN RECEIVE "CYBER.TXT" DATA AND REWRITE IT TO "CYBER_RECEIVED.TXT" FILE.
2. "CYBER_RECEIVED.TXT": COPY OF THE DATA IN "CYBER.TXT" SENT BY NEU SV CAMPUS ADMIN(CLIENT SIDE).



Client Side Files

Server Side Files

```
#include<stdio.h>
// #include<conio.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
void heading();
void main_heading();
void menu();
void user_menu();
void insert();
void display();
void search();
```

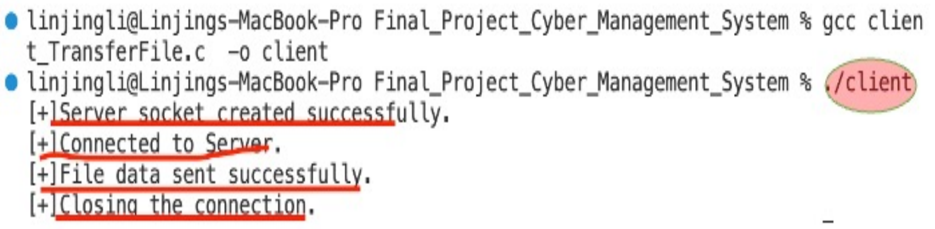
cyber.txt

```
1 adam
3 Cathy
```

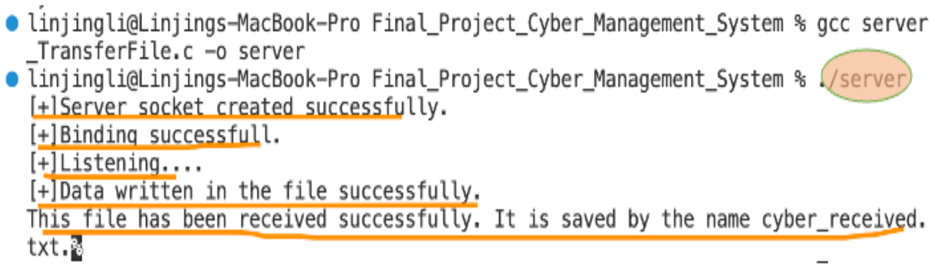
```
/*
 This is the server implementation of transfering file betwe
 server side and client side.
 In these implementations, the client and server can .
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include<unistd.h>
#define SIZE 1024
```

cyber_received.txt

```
1 adam
3 Cathy
```



```
linjingli@Linjings-MacBook-Pro Final_Project_Cyber_Management_System % gcc clien
t_TransferFile.c  -o client
linjingli@Linjings-MacBook-Pro Final_Project_Cyber_Management_System % ./client
[+]Server socket created successfully.
[+]Connected to Server.
[+]File data sent successfully.
[+]Closing the connection.
```



```
linjingli@Linjings-MacBook-Pro Final_Project_Cyber_Management_System % gcc server
_TransferFile.c -o server
linjingli@Linjings-MacBook-Pro Final_Project_Cyber_Management_System % ./server
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening....
[+]Data written in the file successfully.
This file has been received successfully. It is saved by the name cyber_received.
txt.
```

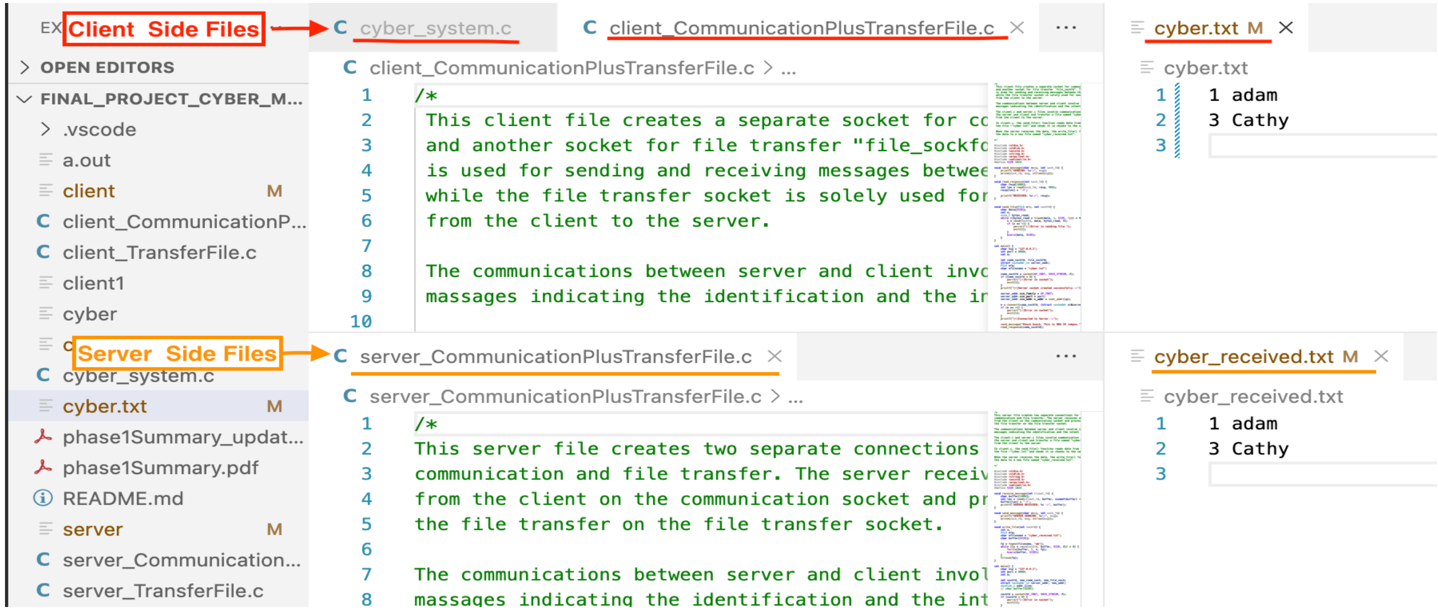
Overview of *Scenario two:*

CLIENT SIDE:

1. "CYBER_SYSTEM.C" FILE: FOR NEU SV CAMPUS ADMIN TO MANAGE STUDENT INFO, GENERATE "CYBER.TXT" FILE IN THE LOCAL.
2. " CLIENT_COMMUNICATIONPLUSTRANSFERFILE.C" FILE: SOCKET PROGRAMMING FILE TO CONNECT TO SERVER, AFTER CONNECTION, FIRST CHAT WITH SERVER,THEN TRANSFER "CYBER.TXT" TO SERVER.
3. "CYBER.TXT" FILE: STORES THE STUDENT DATA GENERATED BY RUNNING "CYBER_SYSTEM.C" FILE.

SERVER SIDE:

1. "SERVER_COMMUNICATIONPLUSTRANSFERFILE.C" FILE: SOCKET PROGRAMMING FILE FOR CONNECTING CLIENT - NEU SV CAMPUS SIDE, ONCE CONNECTED, FIRST CHAT WITH CLIENT AND THEN RECEIVE "CYBER.TXT" DATA AND REWRITE IT TO "CYBER_RECEIVED.TXT" FILE.
2. "CYBER_RECEIVED.TXT": COPY OF THE DATA IN "CYBER.TXT" SENT BY NEU SV CAMPUS ADMIN(CLIENT SIDE).

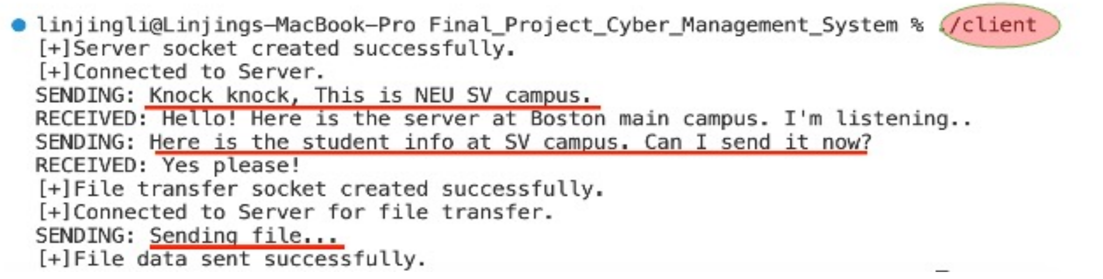


```
/*
This client file creates a separate socket for cc
and another socket for file transfer "file_sockfc
is used for sending and receiving messages betwee
while the file transfer socket is solely used for
from the client to the server.

The communications between server and client invc
massages indicating the identification and the ir
```

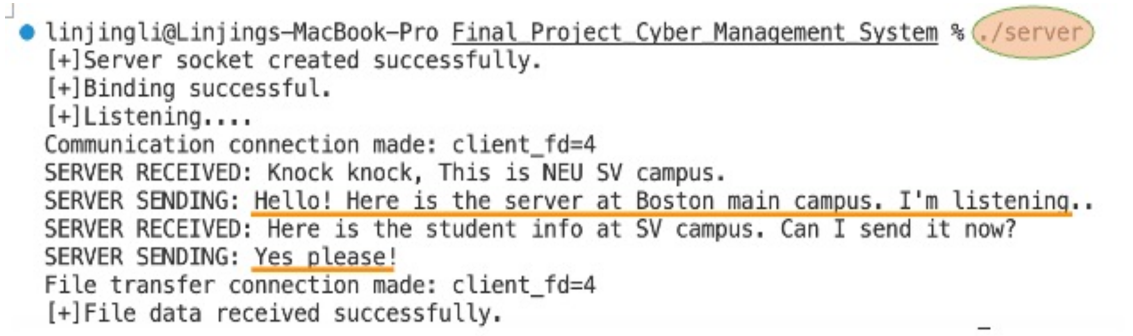
```
1 adam
3 Cathy
```

```
/*
This server file creates two separate connections
communication and file transfer. The server receiv
from the client on the communication socket and pr
the file transfer on the file transfer socket.

The communications between server and client invol
massages indicating the identification and the int
```

```
1 adam
3 Cathy
```

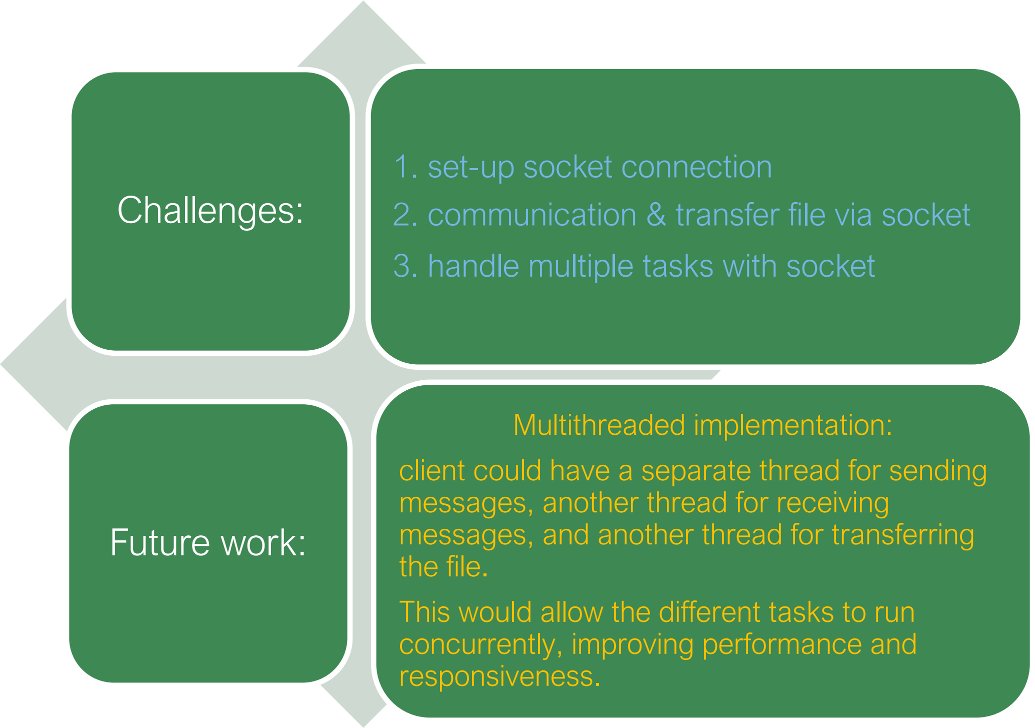


```
linjingli@Linjings-MacBook-Pro Final_Project_Cyber_Management_System % ./client
[+]Server socket created successfully.
[+]Connected to Server.
SENDING: Knock knock, This is NEU SV campus.
RECEIVED: Hello! Here is the server at Boston main campus. I'm listening..
SENDING: Here is the student info at SV campus. Can I send it now?
RECEIVED: Yes please!
[+]File transfer socket created successfully.
[+]Connected to Server for file transfer.
SENDING: Sending file...
[+]File data sent successfully.
```



```
linjingli@Linjings-MacBook-Pro Final_Project_Cyber_Management_System % ./server
[+]Server socket created successfully.
[+]Binding successful.
[+]Listening....
Communication connection made: client_fd=4
SERVER RECEIVED: Knock knock, This is NEU SV campus.
SERVER SENDING: Hello! Here is the server at Boston main campus. I'm listening..
SERVER RECEIVED: Here is the student info at SV campus. Can I send it now?
SERVER SENDING: Yes please!
File transfer connection made: client_fd=4
[+]File data received successfully.
```

Challenges & Future work



Challenges:

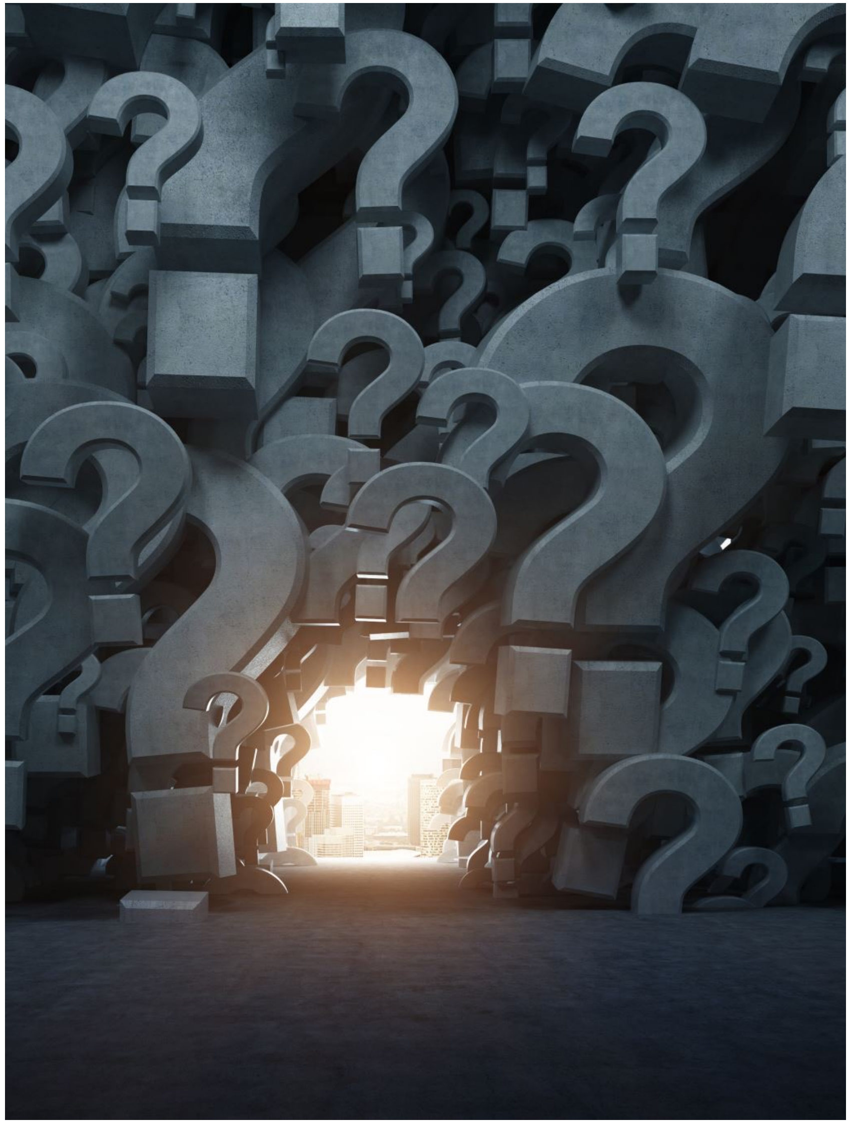
1. set-up socket connection
2. communication & transfer file via socket
3. handle multiple tasks with socket

Future work:

Multithreaded implementation:

client could have a separate thread for sending messages, another thread for receiving messages, and another thread for transferring the file.

This would allow the different tasks to run concurrently, improving performance and responsiveness.



Conclusion

LEARNT SOCKET PROGRAMMING

UNDERSTOOD THE FUNDAMENTALS OF NETWORKING, SUCH AS IP ADDRESSES, PORTS, AND PROTOCOLS

EXPLORED CROSS-PLATFORM APPLICATION, DISTRIBUTED SYSTEMS