DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

CS531P – COMPUTER NETWORKS

CIA 3-Component 2

TCP implementation to build a two-player game Tic-Tac-Toe

Submitted by

DHARATHI VENKATESH - 1960414

Department of Computer Science and Engineering

School of Engineering and Technology,

CHRIST(Deemed to Be University),

Kumbalagodu, Bangalore - 74.

November 2021

INDEX

1. INTRODUCTION

A client server architecture is a computing model where the client requests services and the server provides the requested services. It operates on the concept of distributed computing systems where all nodes work individually. It is a type of shared computing network in which numerous remotely linked devices send several requests to a centralised system known as a server. Examples of Client-Server architecture include email servers, file servers and web servers which handle multiple websites.

TCP/IP protocol makes use of socket programming where sockets are used to establish a connection between different devices. Once the connection is established a socket descriptor (a task specific number) is used to uniquely identify that connection.

A program code in C language was written to implement the TCP/IP protocol and build the two-player game Tic-Tac-Toe. In this two-player game, one player is assigned to use the symbol ‘x’ the other to use the symbol ‘o’ and a grid with 9 boxes in a grid is drawn. Each player gets their turn after the other person has finished marking their symbol in the grid. The player with symbol ‘x’ gets the first chance and then the other player and this goes on until one person wins or the match is a draw. In order to win, the player should have marked his/symbol three times consecutively either in one row, column or a diagonal. Here the two players are the Client and the Server.

2. About TCP

Transmission control protocol allows application programs and computing devices to communicate over a network. It's purpose is to send packets across the internet and ascertain that data and messages are delivered successfully over networks. TCP is added over internet protocol for reliable transmission of packets. Before transmitting data, it first sets up a connection between source and destination, ensuring that it remains active until communication begins. It then breaks the data into smaller packets while maintaining data integrity throughout.

For establishing a connection both endpoints must have a unique IP address and desired port for information transfer. The client first sends a SYN(synchronize) packet with a unique number to the server. The number ensures data integrity(transmission of entire data in the correct order).

If the server receives the SYN packet it agrees to a connection and sends a SYN-ACK(synchronize-acknowledgement) packet to the client, which includes server's sequence number and client's sequence number+1.

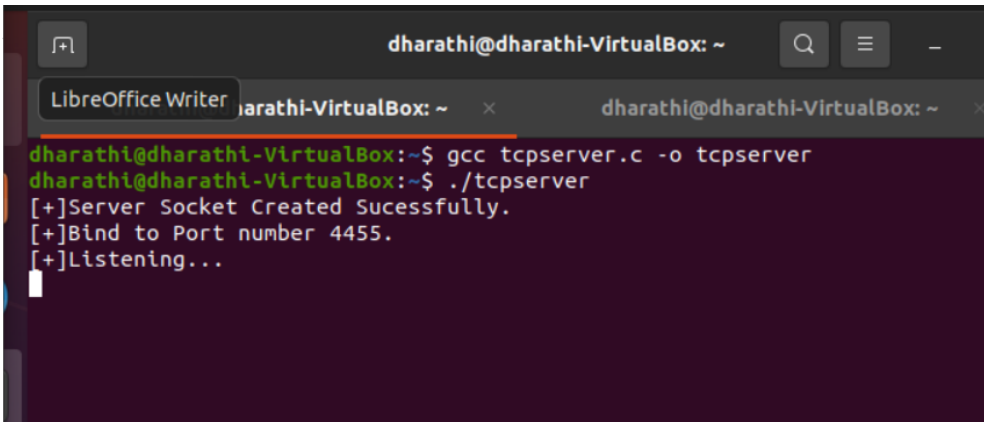
After the client receives the SYN-ACK packet, it sends an ACK(Acknowledgement) packet with the sequence number of server +1 and at the same time starts transferring the information.

Advantages of TCP over UDP:

1) TCP gives guarantees that a packet will reach the destination without any duplication and the order of data will be the same. On the other hand UDP does not guarantee that data will reach its destination. it does not give guarantee that data will be in the same order and it also does not give guarantee that data will reach its destination without any duplication.
2) TCP is a reliable protocol but UDP is an unreliable protocol.
3) Data transmission is more dependable on TCP than UDP.

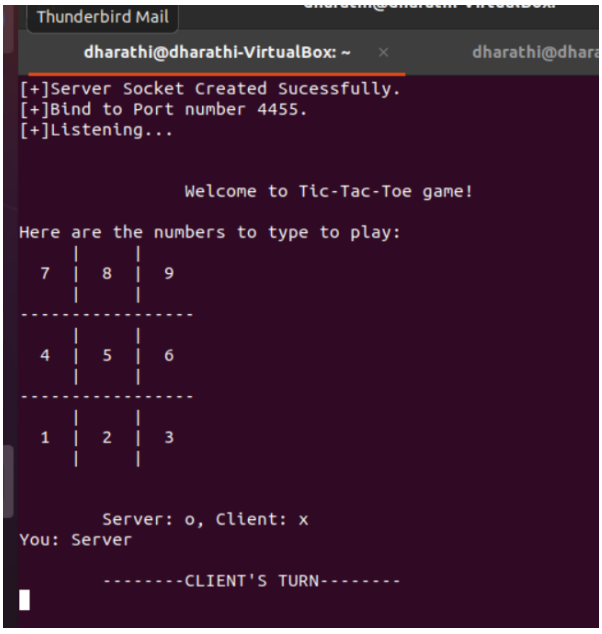
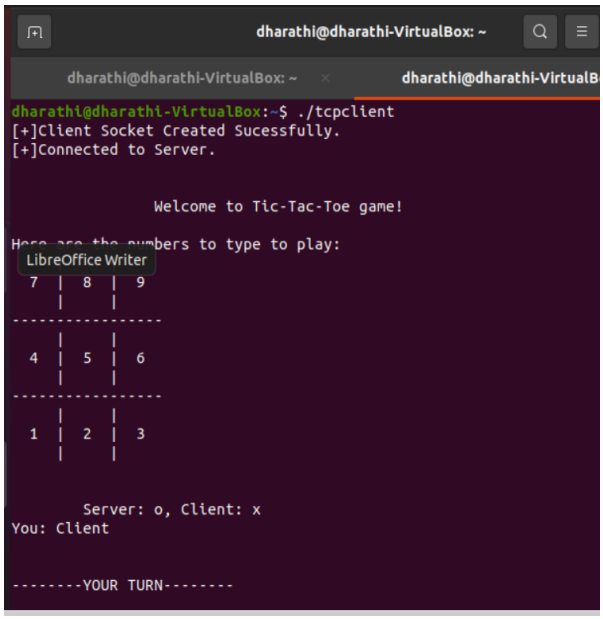
3. OUTPUT

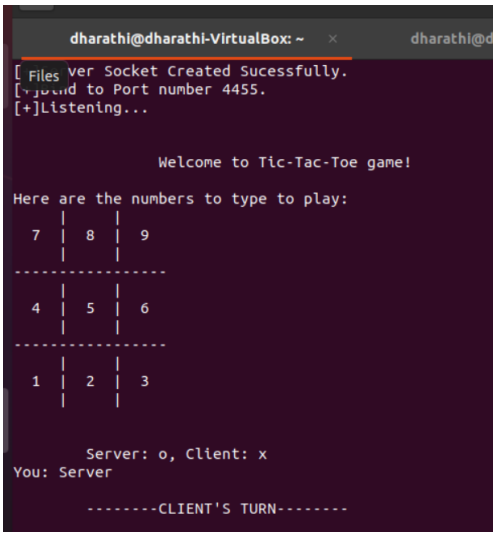
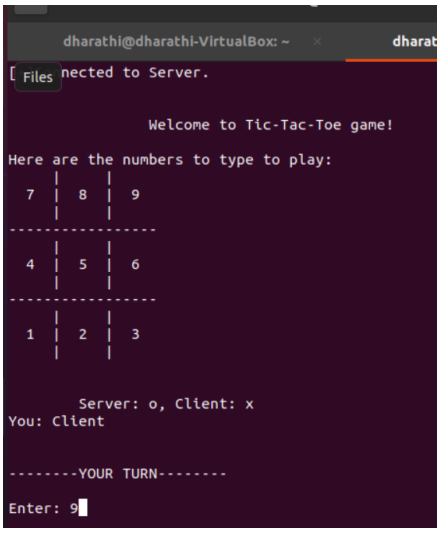
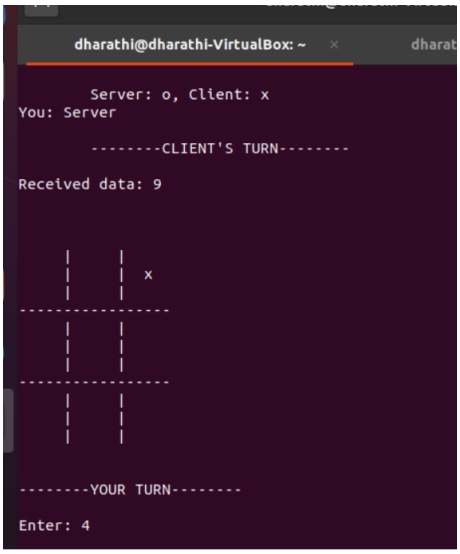
Server waiting for Client to connect (i.e, execution of client side code):



After connection:

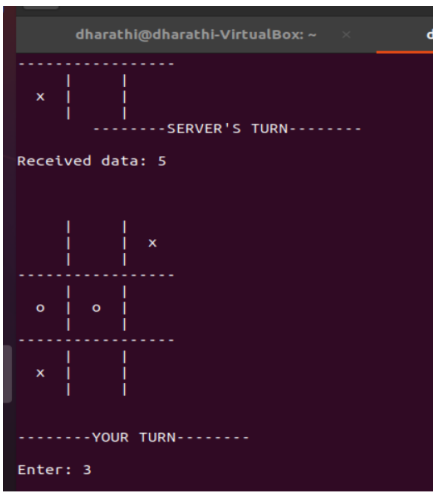
Case 1: Server wins the match

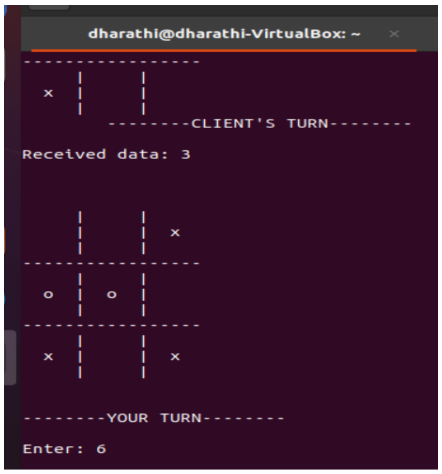
SERVER SIDE	CLIENT SIDE
	

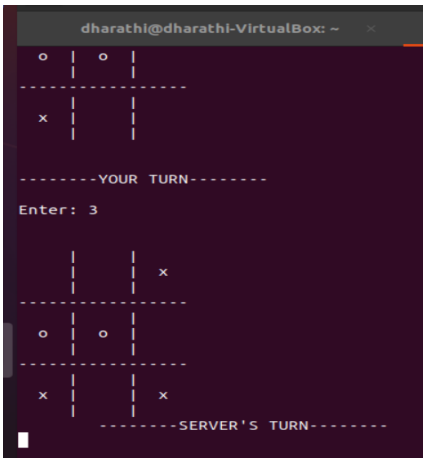
GAME 2	GAME 2
	
	

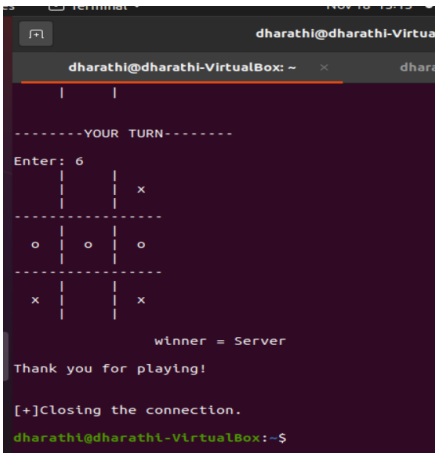
dharathi@dharathi-VirtualBox: ~
Files
ver Socket Created Sucessfully.
nd to Port number 4455.
[+]Listening...
Welcome to Tic-Tac-Toe game!
Here are the numbers to type to play:
7 | 8 | 9
4 | 5 | 6
1 | 2 | 3
Server: o, Client: x
You: Server
--------CLIENT'S TURN--------
Received data: 9
x
--------YOUR TURN--------
Enter: 4

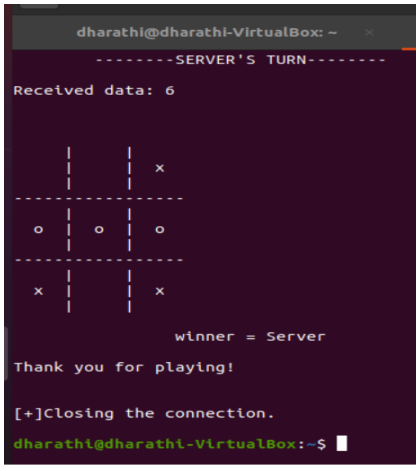
dharathi@dharathi-VirtualBox: ~
Files
nected to Server.
Welcome to Tic-Tac-Toe game!
Here are the numbers to type to play:
7 | 8 | 9
4 | 5 | 6
1 | 2 | 3
Server: o, Client: x
You: Client
--------YOUR TURN--------
Enter: 9
--------SERVER'S TURN--------
Received data: 4
x
o
--------YOUR TURN--------
Enter: 1





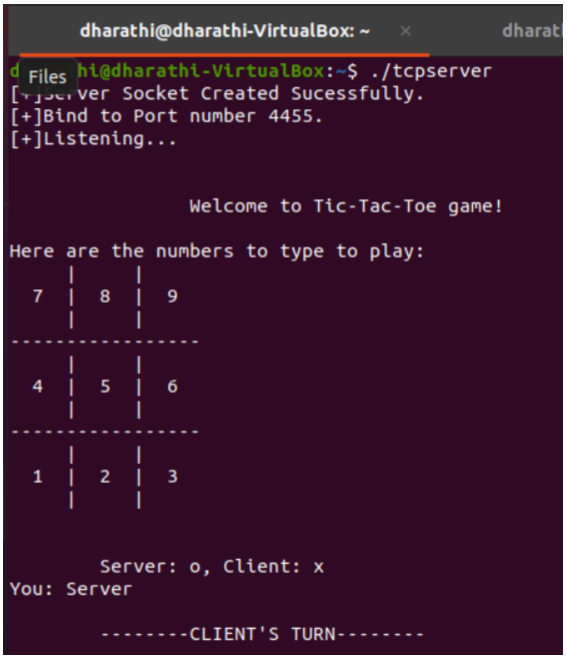


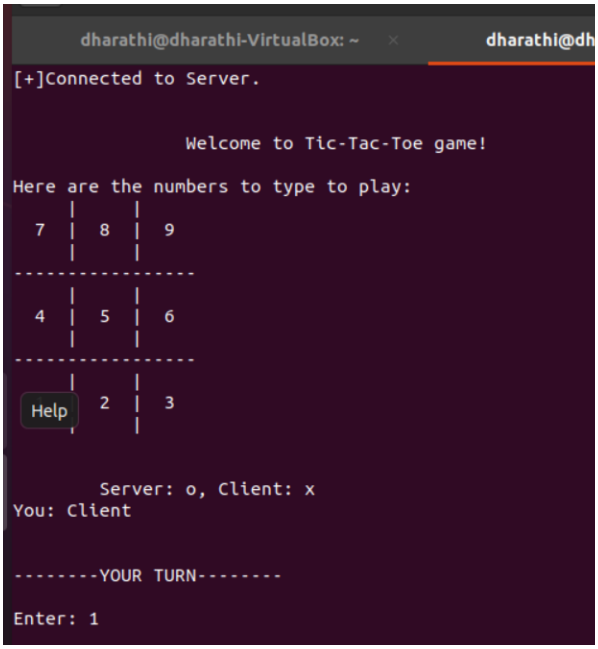


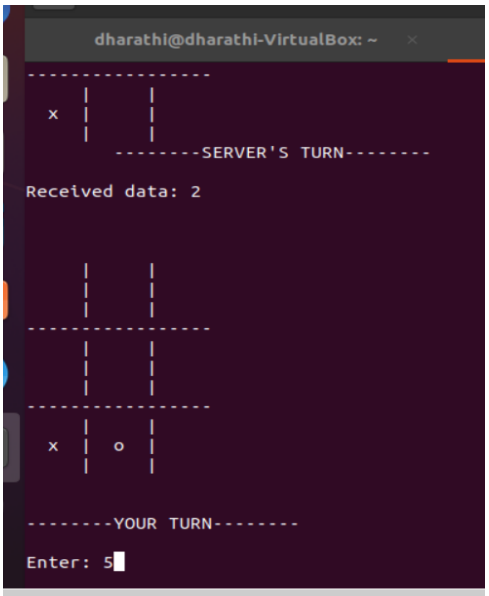


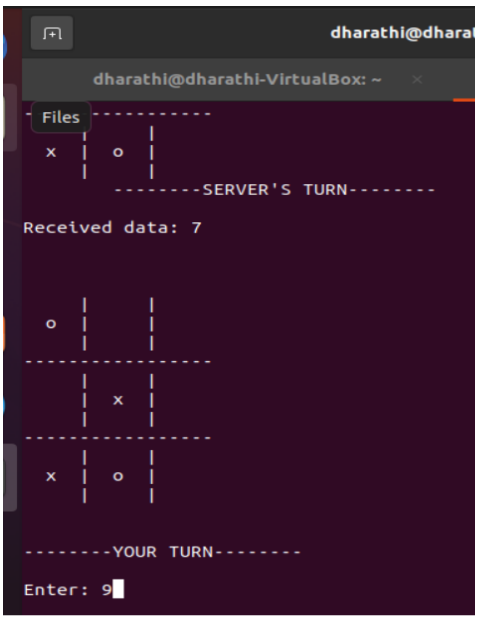
Case 2: Client wins the match:

SERVER SIDE	CLIENT SIDE



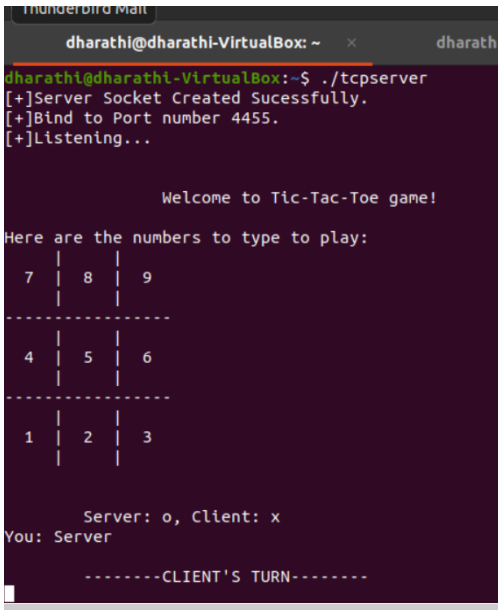






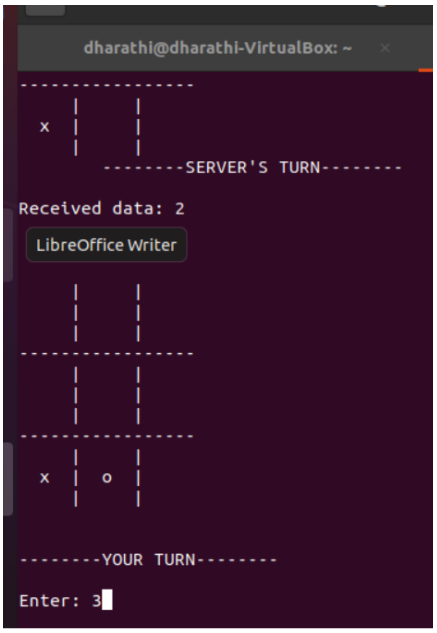
Case 3: Draw match

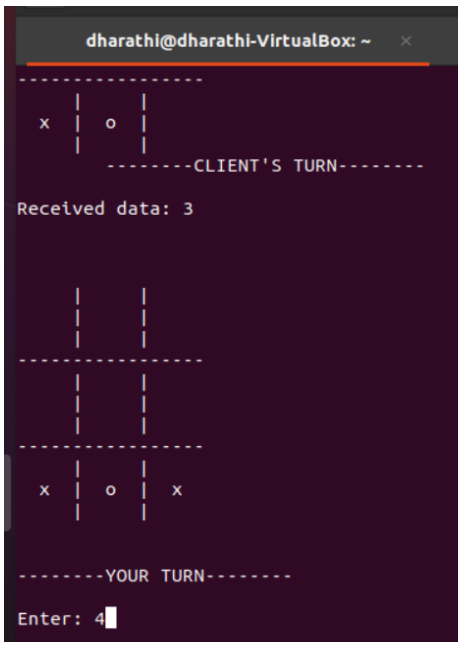
SERVER SIDE	CLIENT SIDE


dharathi@dharathi-VirtualBox: ~
dharathi@dharathi-VirtualBox:~$ ./tcpserver
[+]Server Socket Created Sucessfully.
[+]Bind to Port number 4455.
[+]Listening...
Welcome to Tic-Tac-Toe game!
Here are the numbers to type to play:
7 | 8 | 9
4 | 5 | 6
1 | 2 | 3
Server: o, Client: x
You: Server
--------CLIENT'S TURN--------


dharathi@dharathi-VirtualBox: ~
[+]Connected to Server.
Welcome to Tic-Tac-Toe game!
Here are the numbers to type to play:
7 | 8 | 9
4 | 5 | 6
1 | 2 | 3
Server: o, Client: x
You: Client
--------YOUR TURN--------
Enter: 1


dharathi@dharathi-VirtualBox: ~
Files
Server: o, Client: x
You: Server
--------CLIENT'S TURN--------
Received data: 1
x
--------YOUR TURN--------
Enter: 2


dharathi@dharathi-VirtualBox: ~
x
--------SERVER'S TURN--------
Received data: 2
LibreOffice Writer
x | o
--------YOUR TURN--------
Enter: 3


dharathi@dharathi-VirtualBox: ~
x | o |
--------CLIENT'S TURN--------
Received data: 3
x | o | x
--------YOUR TURN--------
Enter: 4

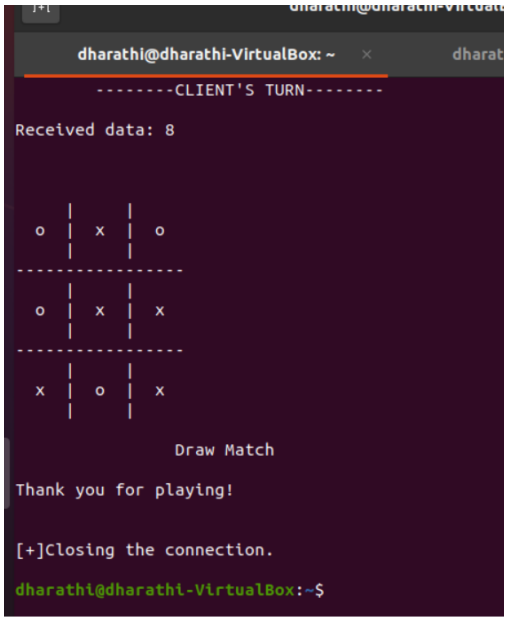

dharathi@dharathi-VirtualBox: ~
x | o | x
--------SERVER'S TURN--------
Received data: 4
o |
x | o | x
--------YOUR TURN--------
Enter: 6

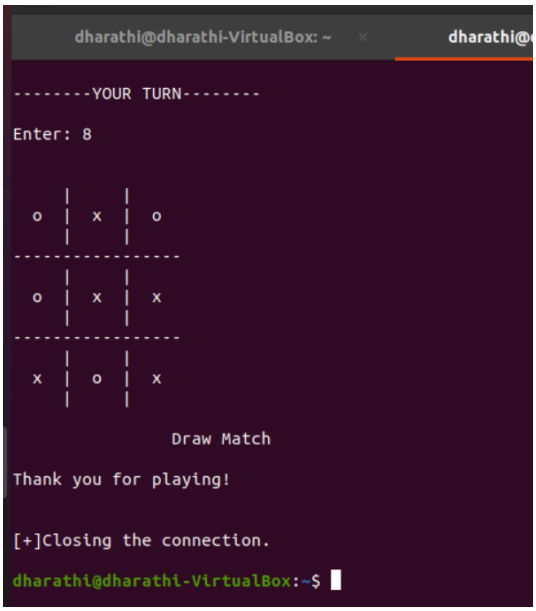

dharathi@dharathi-VirtualBox: ~
x | o | x
--------CLIENT'S TURN--------
Received data: 6
o | | x
x | o | x
--------YOUR TURN--------
Enter: 9


dharathi@dharathi-VirtualBox: ~
x | o | x
--------SERVER'S TURN--------
Received data: 9
| o
o | | x
x | o | x
--------YOUR TURN--------
Enter: 5


dharathi@dharathi-VirtualBox: ~
x | o | x
--------CLIENT'S TURN--------
Received data: 5
o
o | x | x
x | o | x
--------YOUR TURN--------
Enter: 7

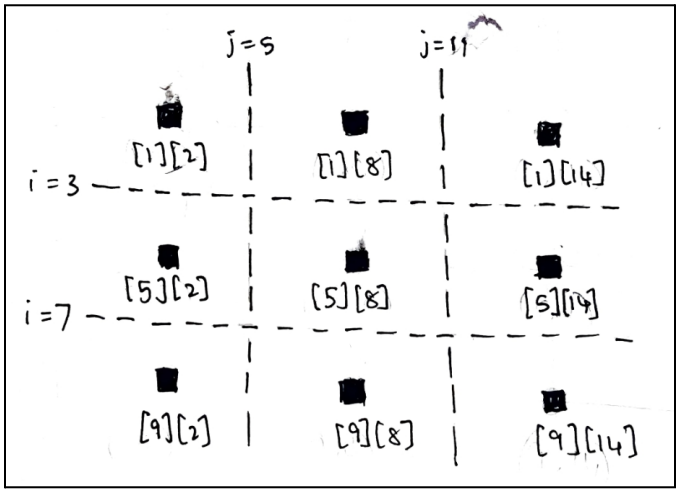

dharathi@dharathi-VirtualBox: ~
x | o | x
Rhythmbox
-----SERVER'S TURN--------
Received data: 7
o | o
o | x | x
x | o | x
--------YOUR TURN--------
Enter: 8


dharathi@dharathi-VirtualBox: ~
--------CLIENT'S TURN--------
Received data: 8
o | x | o
o | x | x
x | o | x
Draw Match
Thank you for playing!
[+]Closing the connection.
dharathi@dharathi-VirtualBox:~$


dharathi@dharathi-VirtualBox: ~
--------YOUR TURN--------
Enter: 8
o | x | o
o | x | x
x | o | x
Draw Match
Thank you for playing!
[+]Closing the connection.
dharathi@dharathi-VirtualBox:~$

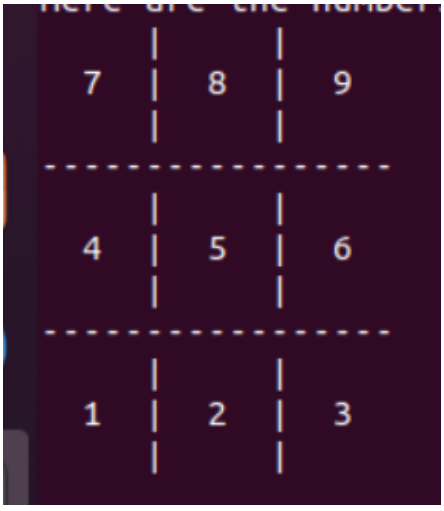
4. PROGRAM CODE EXPLANATION

To build the Tic-Tac-Toe game a very simple approach was implemented by using 2D arrays/matrices. In the program code written above there are 2 matrices used. The first matrix is used to hold the numbering of positions that the players have to enter according to which the symbol has to be marked. This matrix is named ‘grid’. The other matrix ‘grid1’ stores the moves made by the players and initially contains blank spaces in all the positions where the players can enter the symbols. Both these matrices are based on the base architecture of as shown below:



The boxes indicate that the symbols can be entered in those positions and below each box is the index position ([i][j]) of that position.

The ‘grid’ matrix holding all the positions:



By default we assign the Client player with the symbol ‘x’ and the Server player with the symbol ‘o’. The game can be split into rounds. Each round starts with the Client typing the input followed by the Server. If a winner is found at any point in the game, the message stating the winner will be printed and the execution is stopped. In the worst case scenario, when it is a ‘Draw Match’ the rounds can go up to 5 rounds and in the 5th round only the Server types the input. The checking for the winner starts only in the third round as there must be a minimum of “three consecutive symbols” for a player to win.

Socket variable at Server’s side: newSocket
Socket variable at Client’s side: clientSocket

Code for socket connection in Server’s side:

```
int sockfd;
struct sockaddr_in serverAddr;

int newSocket;
struct sockaddr_in newAddr;

socklen_t addr_size;
char sen[2];
char rec[2];
```

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
printf("[+]Server Socket Created Sucessfully.\n");
memset(&serverAddr, '\0', sizeof(serverAddr));

serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(PORT);
serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

bind(sockfd, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
printf("[+]Bind to Port number %d.\n", 4455);

listen(sockfd, 5);
printf("[+]Listening...\n");

newSocket = accept(sockfd, (struct sockaddr*)&newAddr, &addr_size);
```

Code for socket connection in Client’s side:

```
int clientSocket;
struct sockaddr_in serverAddr;
char sen[2];
char rec[2];
```

```
clientSocket = socket(PF_INET, SOCK_STREAM, 0);
printf("[+]Client Socket Created Sucessfully.\n");

memset(&serverAddr, '\0', sizeof(serverAddr));
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(PORT);
serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

connect(clientSocket, (struct sockaddr*)&serverAddr, sizeof(serverAddr)

printf("[+]Connected to Server.\n");

printf("\n\n\t\tWelcome to Tic-Tac-Toe game!\n\n");
printf("Here are the numbers to type to play:\n");
```

Code for sending data from Server's side:

```
send(newSocket,sen,strlen(sen),0);
```

Code for sending data from Client's side:

```
send(clientSocket,sen,strlen(sen),0);
```

Code for receiving data from Server's side:

```
recv(newSocket,rec,2,0);
```

Code for receiving data from Client's side:

```
recv(clientSocket,rec,2,0);
```

Closing the socket after communication in Server's side:

```
close(newSocket);
exit(0);
```

Closing the socket after communication in Client's side:

```
close(clientSocket);
exit(0);
```

Methods/Functions used:

- **check():** Checks for a winner of the game and prints it. In case all the grids are filled and there is no winner “Draw Match” is printed.
- **modifyx():** Modifies ‘grid1’ with the new ‘x’ value that the player has entered the position for.
- **modifyo()::** Modifies ‘grid1’ with the new ‘o’ value that the player has entered the position for.
- **printgrid():** Prints ‘grid1’.

5. CONCLUSION

The main learning of our project was to understand the practical working of an TCP/IP protocol. A good understanding of the basics of networking can help us to understand, modify and identify the errors. All the rounds of the Tic-Tac-Toe game were coded individually instead of being looped in order to completely understand the communication between the Server and the Client that takes place in every round, thus resulting in redundant and long program code. In order to use the most basic concepts of programming 2D arrays/matrices were used to store the moves made by each player. To build the grid and to check for the winner there is a requirement for the programmer to have the skill of analyzing patterns.

Learning outcomes obtained from this project:

- Understanding of the working of TCP/IP protocol.
- Understanding of Socket programming and the use of ports.
- In depth understanding of how the control flows among the Server and the Client when they communicate with each other.
- Improvement of skills like pattern analysis and building of code in C language.

Major drawback of the project implementation: long and redundant code.

6. REFERENCES

1. For the basic coding of the TCP/IP protocol:
   https://www.youtube.com/watch?v=hptViBE23fI

2. For advantages of TCP over UDP
   https://www.wowza.com/blog/udp-vs-tcp

3. For about TCP
   https://www.ionos.com/digitalguide/server/know-how/introduction-to-tcp/
   https://www.techtarget.com/searchnetworking/definition/TCP
   https://www.fortinet.com/resources/cyberglossary/tcp-ip

4. For introduction
   https://www.easytechjunkie.com/what-is-socket-programming.htm
   https://www.jigsawacademy.com/blogs/cyber-security/what-is-client-server-architecture/