A Networked Ping Pong Game

TISHA SIRAZUM MUNIRA, Louisiana State University , USA

A networked game is very good implementation to check the network connectivity and also a popular medium of entertainment. In this article we will describe a simple networked ping pong game and its implementation and evaluate its efficiency and performance. Here the basic code is done using C++ language and Windows OS environment.

CCS Concepts: • **Networked System**; • **Client-Server Game**; • **Windows Operating System**; • **C++ programming language**;

Additional Key Words and Phrases: Multiple Client,One Server-game manager, WinSock2 API

ACM Reference Format:
Tisha Sirazum Munira. 2017. A Networked Ping Pong Game. *ACM Trans. Graph.* 9, 4, Article 39 (March 2017), 4 pages. https://doi.org/0000001.0000001_2

1 INTRODUCTION

Socket programming is a challenging task. It is more challenging to do socket programming in Windows OS using C++ language. WinSock2 API in windows helps to do the Socket programming in Windows environment. For a client-server we need to establish a connection between a server and two clients. Winsock2 with C++ language here used to establish the network connections and C++ language is used to create the gaming environment and all the game logics here as well. Game logic is kept easy here for simplification.

In this article, we will discuss about overall process algorithm as well as the implementations and performance evaluations.

2 ALGORITHM

In our algorithm, at first we install WinSock2 API with Visual Studio. Then we create two console based project: one for server and one for client. Then we set up a connection so that server can be connected with two clients. Then we write code for game within client side and server will be responsible to send the keystrokes by the clients. Client sends their corresponding keystrokes to move the paddle positions to up and down to make a score. The ball starts moving on the random direction and if it is hit by the paddle then it bounces back following the basic physics law. If clients are able to hit the ball with their paddle then they score a point. If one client scores two points then it wins. The overall algorithm is as follows:

Author's address: Tisha Sirazum Munira, Louisiana State University , USA, stisha1@lsu.eduProject: PA-1a (Networked Ping pong game) Instructor: Feng Chen Class: cs7103-au17 LogonID: cs710307.

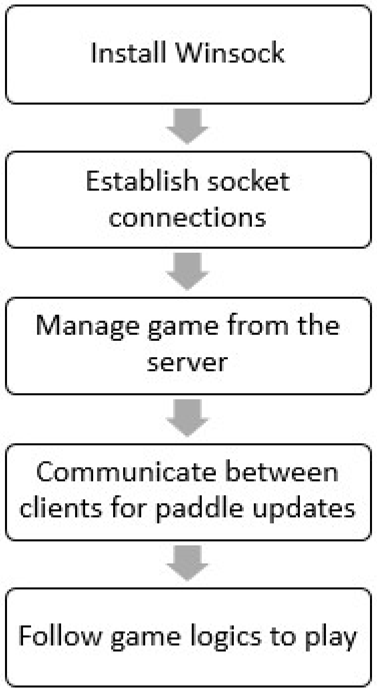

Install Winsock
Establish socket connections
Manage game from the server
Communicate between clients for paddle updates
Follow game logics to play

Fig. 1. Flow chart for the proposed algorithm

3 DESIGN AND IMPLEMENTATIONS

3.1 Client-Server connection establishment

As Algorithm states, at first we downloaded Winsock2 and installed it and attached with the Visual Studio by following the steps described [Microsoft Corporation 2017]. Winsock2 mantains TCP/IP protocol in socket programming [Microsoft 2017]Microsoft Corporation [2017].

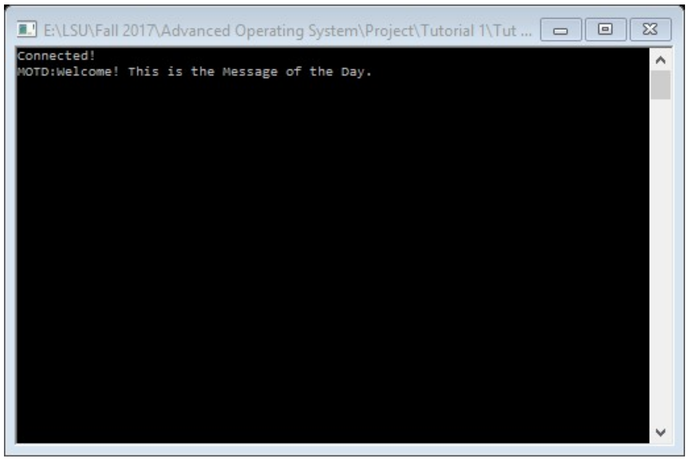
In the very beginning of coding we create two console application based C++ projects on Visual Studio; one for server and one for client.We follow tutorials and its discussions to establish a network connection between client and server [Jpres 2016]. The pseudo source code for client and server is as follows:

Server.cpp:

```
int main()
{
//WinSock Startup
WSAData wsaData;
WORD DllVersion = MAKEWORD(2, 1);
    //If 0 then an error has occured in the WinSock Startup.
 if (WSAStartup(DllVersion, &wsaData) != 0)
{
MessageBoxA(NULL,"WinSock startup failed","Error",
        MB_OK|MB_ICONERROR);
return 0;
}
//Address that will be binded to listening socket
SOCKADDR_IN addr;
//length of the address (required for accept call)
    int addrlen = sizeof(addr);
addr.sin_addr.s_addr = INADDR_ANY;
addr.sin_port = htons(1111); //Port
addr.sin_family = AF_INET; //IPv4 Socket

//Create socket listen for new connections
    SOCKET sListen = socket(AF_INET, SOCK_STREAM, NULL);
//Bind the address to the socket
    bind(sListen, (SOCKADDR*)&addr, sizeof(addr));
//Places sListen socket in a state in which
    //it is listening for an incoming connection.
    //Note:SOMAXCONN = Socket Oustanding Max Connections
    listen(sListen, SOMAXCONN);

SOCKET newConnection; //Socket to hold the client's connection
newConnection = accept(sListen, (SOCKADDR*)&addr, &addrlen);
//Accept a new connection
    //If accepting the client connection failed
    if (newConnection == 0)
{
std::cout <<"Failed to accept client's connection."<< std::endl;
}
else //If client connection properly accepted
{
std::cout << "Client Connected!" << std::endl;
//Create buffer with message of the day
      char MOTD[256] = "Welcome! This is the Message of the Day.";
 //Send MOTD buffer
        send(newConnection, MOTD, sizeof(MOTD), NULL);
}

system("pause");
return 0;
}
```

Client.cpp:

```
int main()
{
//Winsock Startup
WSAData wsaData;
WORD DllVersion = MAKEWORD(2, 1);
    //if 0 then some error occur
    if (WSAStartup(DllVersion, &wsaData) != 0)
{
MessageBoxA(NULL,"Winsock startup failed","Error",
        MB_OK|MB_ICONERROR);
exit(1);
}
//Address to be binded to our Connection socket
SOCKADDR_IN addr;
    //Need sizeofaddr for the connect function
int sizeofaddr = sizeof(addr);
    //Address = localhost
inet_pton(AF_INET, "127.0.0.1", &(addr.sin_addr.s_addr));
addr.sin_port = htons(1111); //Port = 1111
```


E:\LSU\Fall 2017\Advanced Operating System\Project\Tutorial 1\Tut ...

```
Connected!
MOTD:Welcome! This is the Message of the Day.
```

Fig. 2. Output shows from server side is connected [Jpres 2016]

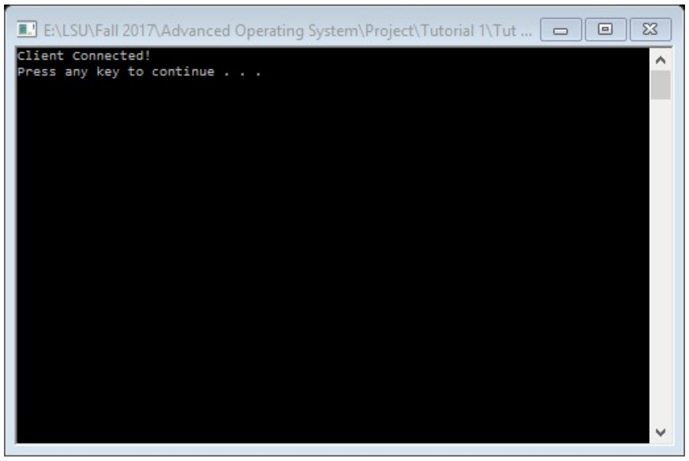
```
addr.sin_family = AF_INET; //IPv4 Socket

//Set Connection socket
    SOCKET Connection = socket(AF_INET, SOCK_STREAM, NULL);
//If unable to connect...
   if (connect(Connection, (SOCKADDR*)&addr, sizeofaddr) != 0)
{
MessageBoxA(NULL, "Failed to Connect","Error",MB_OK|MB_ICONERROR);
return 0; //Failed to Connect
}
std::cout << "Connected!" << std::endl;

char MOTD[256];
 //Receive Message of the Day buffer into MOTD array
    recv(Connection, MOTD, sizeof(MOTD), NULL);
std::cout << "MOTD:" << MOTD << std::endl;
}
```

To check if the connection establishes, we first compile and run the server.cpp and then client.cpp file and got output as fig 2 and fig 3. Thus we know that connection establishes [Jpres 2016].

3.2 Ping Pong game

To make a ping pong game we do not use any graphics library or coding, rather we make a simple game environment completely based on C++ code. For this we follow tutorials of [NVitanovic 2016]. We modify the logic according to the project requirement. Here, after colliding with the paddles ball moves on the random direction. After each hit ball as well as paddles reset to their initial position. Whenever a player scores 3 points then it wins. Key 'w' is for left player paddle up movement, 's' is for down and 'i' is for right player paddle up and 'k' for down. Key 'q' is for quiting the game [NVitanovic 2016]. draw() and logic()[NVitanovic 2016] functions pseudo source code are as follows:

```
void Draw()
{
for (int i = 0; i < width + 2; i++)
cout << "\xB2";
cout << endl;
```



```
Client Connected!
Press any key to continue . . .
```

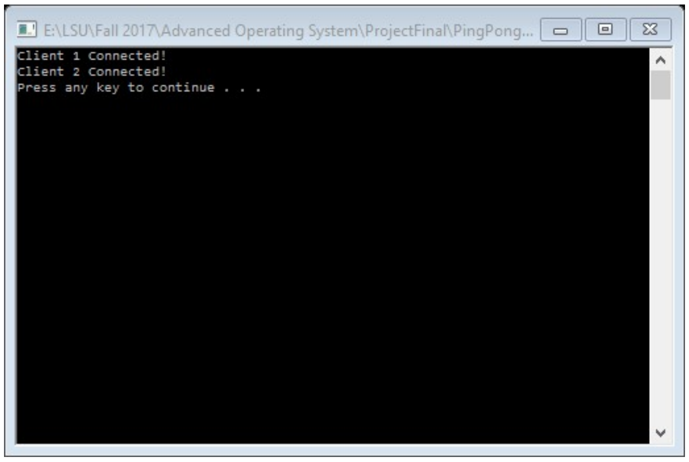
Fig. 3. Output shows from client side is connected
[Jpres 2016]

```
for (int i = 0; i < height; i++)
{
for (int j = 0; j < width; j++)
{
if (j == 0)
cout << "\xB2"; //characters for boundery

if (ballx == j && bally == i)
cout << "O"; //ball
else if (player1x == j && player1y == i)
cout << "\xDB"; //player1
else if (player2x == j && player2y == i)
cout << "\xDB"; //player2

else if (player1x == j && player1y + 1 == i)
cout << "\xDB"; //player1
else if (player1x == j && player1y + 2 == i)
cout << "\xDB"; //player1
else if (player1x == j && player1y + 3 == i)
cout << "\xDB"; //player1

else if (player2x == j && player2y + 1 == i)
cout << "\xDB"; //player1
else if (player2x == j && player2y + 2 == i)
cout << "\xDB"; //player1
else if (player2x == j && player2y + 3 == i)
cout << "\xDB"; //player1
else
cout << " ";

if (j == width - 1)
cout << "\xB2";
}
}

for (int i = 0; i < width + 2; i++)
cout << "\xB2";
cout << endl;
```

```
cout << "Score 1: " << score1 << endl <<
        "Score 2: " << score2 << endl;
if (score1 == 2 && score1 > score2) {
cout << "Player 1 wins!";

ball->Reset();
player1->Reset();
player2->Reset();
}

else if (score2 == 2 && score2 > score1) {
cout << "Player 2 wins!";
ball->Reset();
player1->Reset();
player2->Reset();
}
}
void Logic()
{
     //get positions for ball and paddles
//left paddle
for (int i = 0; i < 4; i++)
if (ballx == player1x + 1)
if (bally == player1y + i)
ball->changeDirection((eDir)
                    ((rand() % 3) + 4));

//right paddle
for (int i = 0; i < 4; i++)
if (ballx == player2x - 1)
if (bally == player2y + i)
ball->changeDirection((eDir)
                    ((rand() % 3) + 1));

//bottom wall
if (bally == height - 1)
ball->changeDirection(ball->getDirection() ==
            DOWNRIGHT ? UPRIGHT : UPLEFT);
//top wall
if (bally == 0)
ball->changeDirection(ball->getDirection() ==
            UPRIGHT ? DOWNRIGHT : DOWNLEFT);
//right wall
if (ballx == width - 1)
ScoreUp(player1);
//left wall
if (ballx == 0)
ScoreUp(player2);
}
```

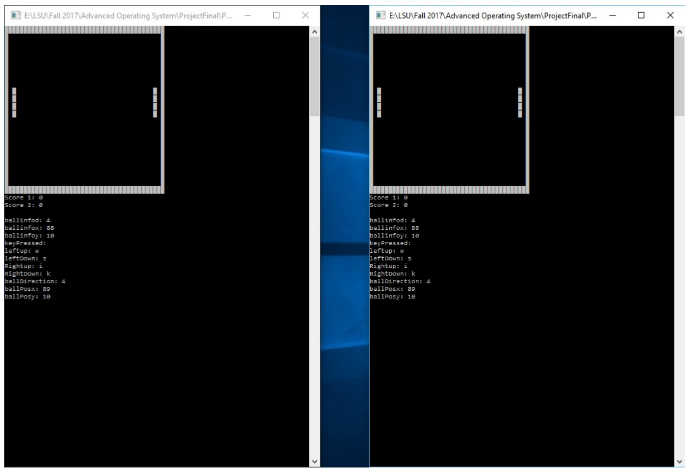
3.3 Ping Pong with client-server

The requirement of the project is that two clients play the game online and a server connects the clients and manages the game. For this we add the pingpong game code to both the server and client. SOMAXCONN = Socket Outstanding Max Connections in Server.cpp allows multiple clients but we need only one client.cpp file. Here we bound the number of clients to two by limiting the loop to two. More than two could be connected but not be able to play the game.


E:\LSU\Fall 2017\Advanced Operating System\ProjectFinal\PingPong...

```
Client 1 Connected!
Client 2 Connected!
Press any key to continue . . .
```

Fig. 4. Final Output shows from server side


E:\LSU\Fall 2017\Advanced Operating System\ProjectFinal\P...
E:\LSU\Fall 2017\Advanced Operating System\ProjectFinal\P...

```
Score 1: 0
Score 2: 0

ballinfod: 4
ballinfox: 88
ballinfoy: 10
keyPressed:
leftup: w
leftDown: s
Rightup: i
RightDown: k
ballDirection: 4
ballPosx: 89
ballPosy: 10
```

```
Score 1: 0
Score 2: 0

ballinfod: 4
ballinfox: 88
ballinfoy: 10
keyPressed:
leftup: w
leftDown: s
Rightup: i
RightDown: k
ballDirection: 4
ballPosx: 89
ballPosy: 10
```

Fig. 5. Output shows from two clients parallel

Here client gets keystroke from the players to play their corresponding sides game i.e, Client1 plays as Player1 and can only move the left sided paddle not the opposite one and vice versa for client2. Clients presses their keystrokes to move the paddle and server receives them and check whether it is from the allotted clients and sends it to all clients including the sender client.Then again client receives it from the server and pass it to the game manager through Play() function. Play() passes it to the Input() function and changes the paddle position accordingly. In the Input() function ball-direction is also chosen randomly. So Input() then passes the ball-direction and its position and the corresponding paddle position to both Draw() and Logic() to draw the GUI part and playing logic accordingly. The whole program is run for 35 cycles and then it restarts. This is done to verify the network connection and to check if it is working correctly. In this project most importance is given to the networking part.The Final output is as Fig 4-5.

4 EVALUATIONS AND CONCLUSIONS

In this article, we develop a networked ping pong game. Here we use TCP/IP protocol by using WinSock2 API. One of the limitations of this work is one client is always lag behind as it requires some time to send the changes through network server and update clients accordingly. So it has a latency issue.If clients presses the keys more frequently then this latency increases more to send the keys and run the Play() functions accordingly and thus makes the problem less efficient. To make it look more efficient keystrokes are put slowly. Another limitation is that the ping pong game out put is always blinking as it uses system clear function and for low processing speed of computers. NVitanovic [2016].

ACKNOWLEDGMENTS

The authors would like to thank Dr. Feng Chen of Louisiana State University for providing specifications about the application scenario and guidelines.

REFERENCES

Jpres. 2016. *C++ WinSock Networking Tutorial.* https://www.youtube.com/watch?v=AjG4jcVn6QE&t=711s.
Microsoft 2017. *Windows Sockets 2 (Windows) - MSDN - Microsoft.* Microsoft. https://msdn.microsoft.com/en-us/library/windows/desktop/ms740673(v=vs.85).aspx.
Microsoft Corporation 2017. *Getting Started with Winsock.* Microsoft Corporation. https://msdn.microsoft.com/en-us/library/windows/desktop/ms738545(v=vs.85).aspx.
NVitanovic. 2016. *C++ Tutorial 21 - Simple Pong Game.* https://www.youtube.com/watch?v=y8QL62SDlcQ.

Received February 2016; revised March 2016; final version June 2016; accepted July 2016