ROCK PAPER SCISSOR GAME USING SOCKET PROGRAMMING

A COURSE PROJECT REPORT

By
MADHAV DHADWAL (RA2111027010062)
BIKRAM ISHAAN SINGH (RA2111027010082)
TALAT BINTI FIRDOUS (RA2111027010115)

Under the guidance of

Dr. A V KALPANA

In partial fulfillment for the Course

of

18CSC302J - COMPUTER NETWORKS

in DSBS

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND

TECHNOLOGY

Kattankulathur, Chengalpattu District

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report "**Rock Paper Scissor game using Socket programming** " is the bonafide work of **MADHAV DHADWAL (RA2111027010062), BIKRAM ISHAAN SINGH(RA2111027010082), TALAT BINTI FIRDOUS (RA2111027010115),** who carried out the project work under my supervision.

SIGNATURE

Dr. A V Kalpana
Assistant Professor
DSBS
SRM Institute of Science and Technology

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professor Dr.Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our Course project Faculty **Dr. A V KALPANA, Assistant Professor, DSBS,** for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr. M LAKSHMI, Department of Data Science And Business Systems** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

1. ABSTRACT

- Socket is a software structure that provides a two way communication link between two nodes. Sockets are primarily used whenever real time communication needs to be established.
- For instance, chat applications, realtime databases, and online multiplayer games, all use Socket under the hood. One popular example is WhatsApp, the text messaging app that uses Socket for its real time messaging service.
- Rock Paper Scissors is a game played by a single player, But in this project we are going to give a completely new meaning to this game.
- By using the latest programming codes and the ever efficient TCP/IP protocol we have proved it.
- We are spinning the game to be a “Multiplayer game!”

2.INTRODUCTION

1.The client-server programming approach separates information producers from information users in a distributed computing system (servers).

2.A client is a software programme that requests resources from a server, such as web pages or IP addresses.

3.Clients can ask a server for this information whenever they want. Users of information are customers.

4.A server is a programme that gives clients access to resources or information. It must be continuously operational and ready to respond to client requirements.

5.Only server apps and their client counterparts can exchange data. Clients are not in direct contact with one another.

6.Here, we have used python compiler to execute our code.

7. Python is a multiprogramming language and it can be used for game development

Rock paper scissors game is created using no extra libraries, but just socket programming

8.In this game the user gets the first chance to pick either rock, paper, or scissors
Then the computer or other player would pick rock, paper or scissors
The winning conditions will be checked, and the winner would get a point

In the end of the game, the winner will be declared

3.REQUIREMENTS

Requirement Analysis

From the given scenario, we draw the following requirements:

1. Identifying the appropriate hardware which would be used (Pycharm)
2. Users on the internet should be able to access the code .
3. Users on the internet should have access only to the public IP address of the server and not the private IP address.
4. The users in the organization should have full access to the server.
5. TCP/IP Network design with IP addressing
6. Features and configuration required on the hardware with explanation

We need to configure a network design keeping the following requirements in mind.

4. ARCHITECTURE AND DESIGN

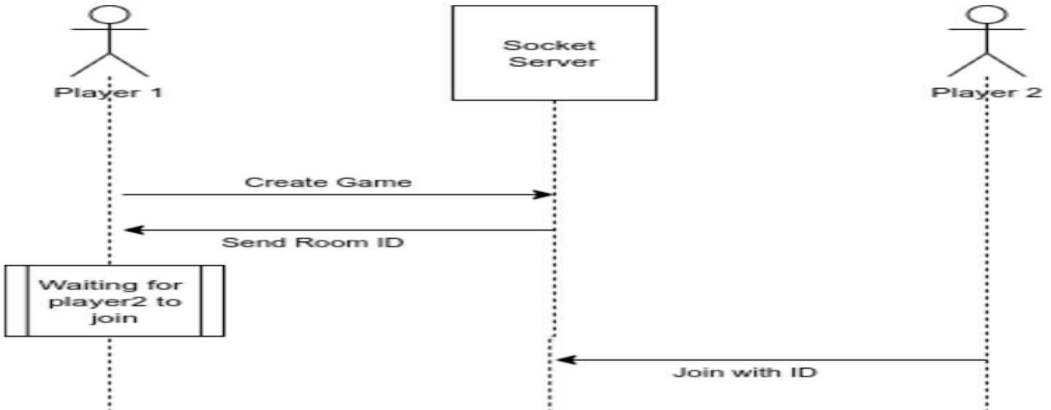
This is a sequence diagram that shows the timeline of events. It's important to understand the flow before we start coding so make sure you understand the diagram completely.

Read through the points below if you are not familiar with Sequence Diagrams.

- Time flows downward.
- Arrows represent events. The start of an arrow denotes the Emitter, the end of an arrow denotes the listener. For example, Player1 emits the Create Game Event and the Socket Server listens to this event.
- A rectangular box denotes waiting/processing time.

First, Player1 emits the create Game Event that the server acknowledges and responds back with a room ID.

Once, player2 emits join Game Event with the same room ID, the server then detects that both players have joined the room and shares the other player's info with each other.



Finally, once both of the players have made their choice, the server processes the result and sends it back to both players.

5. IMPLEMENTATION

1.Client enters their name and then gives a choice for rock,paper,scissors choice

2.By executing the server class the user will be prompt to specify a port number which has to be an integer value strictly greater than zero and less than or equal to 65535.

3.Alternatively “0” is accepted for the standard port (1337) .

4.After submitting the script runs a little validation and checks if the value is within the defined range (0 > x <= 65535) and sets the port value.

5.So far no further validation is implemented, we're not verifying for example if the selected port is reserved or just busy at the moment.

6.If no exception is thrown the program dumps a status message to inform the user, that the server is running on the specified port number.

Input		Result	
Client 1	Client 2	Client 1	Client 2
R	R	Draw	Draw
S	S	Draw	Draw
P	P	Draw	Draw
R	S	You Win	You Lose
S	R	You Lose	You Win
R	P	You Lose	You Win
P	R	You Win	You Lose
S	P	You Win	You Lose
P	S	You Lose	You Win

6. Code

Client side:

```
import pygame
from network import Network
import pickle
pygame.font.init()

width = 700
height = 700
win = pygame.display.set_mode((width, height))
pygame.display.set_caption("Client")


class Button:
    def __init__(self, text, x, y, color):
        self.text = text
        self.x = x
        self.y = y
        self.color = color
        self.width = 150
        self.height = 100

    def draw(self, win):
        pygame.draw.rect(win, self.color, (self.x, self.y, self.width, self.height))
        font = pygame.font.SysFont("comicsans", 40)
        text = font.render(self.text, 1, (255,255,255))
        win.blit(text, (self.x + round(self.width/2) - round(text.get_width()/2), self.y + round(self.height/2) -
round(text.get_height()/2)))

    def click(self, pos):
        x1 = pos[0]
        y1 = pos[1]
        if self.x <= x1 <= self.x + self.width and self.y <= y1 <= self.y + self.height:
            return True
        else:
            return False


def redrawWindow(win, game, p):
    win.fill((128,128,128))

    if not(game.connected()):
        font = pygame.font.SysFont("comicsans", 80)
        text = font.render("Waiting for Player...", 1, (255,0,0), True)
        win.blit(text, (width/2 - text.get_width()/2, height/2 - text.get_height()/2))
    else:
        font = pygame.font.SysFont("comicsans", 60)
        text = font.render("Your Move", 1, (0, 255,255))
        win.blit(text, (80, 200))

        text = font.render("Opponents", 1, (0, 255, 255))
        win.blit(text, (380, 200))

        move1 = game.get_player_move(0)
        move2 = game.get_player_move(1)
        if game.bothWent():
```

```
            text1 = font.render(move1, 1, (0,0,0))
            text2 = font.render(move2, 1, (0, 0, 0))
        else:
            if game.p1Went and p == 0:
                text1 = font.render(move1, 1, (0,0,0))
            elif game.p1Went:
                text1 = font.render("Locked In", 1, (0, 0, 0))
            else:
                text1 = font.render("Waiting...", 1, (0, 0, 0))

            if game.p2Went and p == 1:
                text2 = font.render(move2, 1, (0,0,0))
            elif game.p2Went:
                text2 = font.render("Locked In", 1, (0, 0, 0))
            else:
                text2 = font.render("Waiting...", 1, (0, 0, 0))

        if p == 1:
            win.blit(text2, (100, 350))
            win.blit(text1, (400, 350))
        else:
            win.blit(text1, (100, 350))
            win.blit(text2, (400, 350))

        for btn in btns:
            btn.draw(win)

    pygame.display.update()


btns = [Button("Rock", 50, 500, (0,0,0)), Button("Scissors", 250, 500, (255,0,0)), Button("Paper", 450, 500,
(0,255,0))]
def main():
    run = True
    clock = pygame.time.Clock()
    n = Network()
    player = int(n.getP())
    print("You are player", player)

    while run:
        clock.tick(60)
        try:
            game = n.send("get")
        except:
            run = False
            print("Couldn't get game")
            break

        if game.bothWent():
            redrawWindow(win, game, player)
            pygame.time.delay(500)
            try:
                game = n.send("reset")
            except:
                run = False
                print("Couldn't get game")
                break

            font = pygame.font.SysFont("comicsans", 90)
            if (game.winner() == 1 and player == 1) or (game.winner() == 0 and player == 0):
                text = font.render("You Won!", 1, (255,0,0))
            elif game.winner() == -1:
```

```
            text = font.render("Tie Game!", 1, (255,0,0))
        else:
            text = font.render("You Lost...", 1, (255, 0, 0))

        win.blit(text, (width/2 - text.get_width()/2, height/2 - text.get_height()/2))
        pygame.display.update()
        pygame.time.delay(2000)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
            pygame.quit()

        if event.type == pygame.MOUSEBUTTONDOWN:
            pos = pygame.mouse.get_pos()
            for btn in btns:
                if btn.click(pos) and game.connected():
                    if player == 0:
                        if not game.p1Went:
                            n.send(btn.text)
                    else:
                        if not game.p2Went:
                            n.send(btn.text)

    redrawWindow(win, game, player)

def menu_screen():
    run = True
    clock = pygame.time.Clock()

    while run:
        clock.tick(60)
        win.fill((128, 128, 128))
        font = pygame.font.SysFont("comicsans", 60)
        text = font.render("Click to Play!", 1, (255,0,0))
        win.blit(text, (100,200))
        pygame.display.update()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                run = False
            if event.type == pygame.MOUSEBUTTONDOWN:
                run = False

    main()

while True:
    menu_screen()
```

Server side:

```
import socket
from _thread import *
import pickle
from game import Game

server = "10.11.250.207"
port = 5555
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    s.bind((server, port))
except socket.error as e:
    str(e)

s.listen(2)
print("Waiting for a connection, Server Started")

connected = set()
games = {}
idCount = 0


def threaded_client(conn, p, gameId):
    global idCount
    conn.send(str.encode(str(p)))

    reply = ""
    while True:
        try:
            data = conn.recv(4096).decode()

            if gameId in games:
                game = games[gameId]

                if not data:
                    break
                else:
                    if data == "reset":
                        game.resetWent()
                    elif data != "get":
                        game.play(p, data)

                    conn.sendall(pickle.dumps(game))
            else:
                break
        except:
            break

    print("Lost connection")
    try:
        del games[gameId]
        print("Closing Game", gameId)
    except:
        pass
    idCount -= 1
    conn.close()



while True:
    conn, addr = s.accept()
    print("Connected to:", addr)

    idCount += 1
    p = 0
    gameId = (idCount - 1)//2
    if idCount % 2 == 1:
        games[gameId] = Game(gameId)
        print("Creating a new game...")
```

```
    else:
        games[gameId].ready = True
        p = 1


    start_new_thread(threaded_client, (conn, p, gameId))
```

Game side:

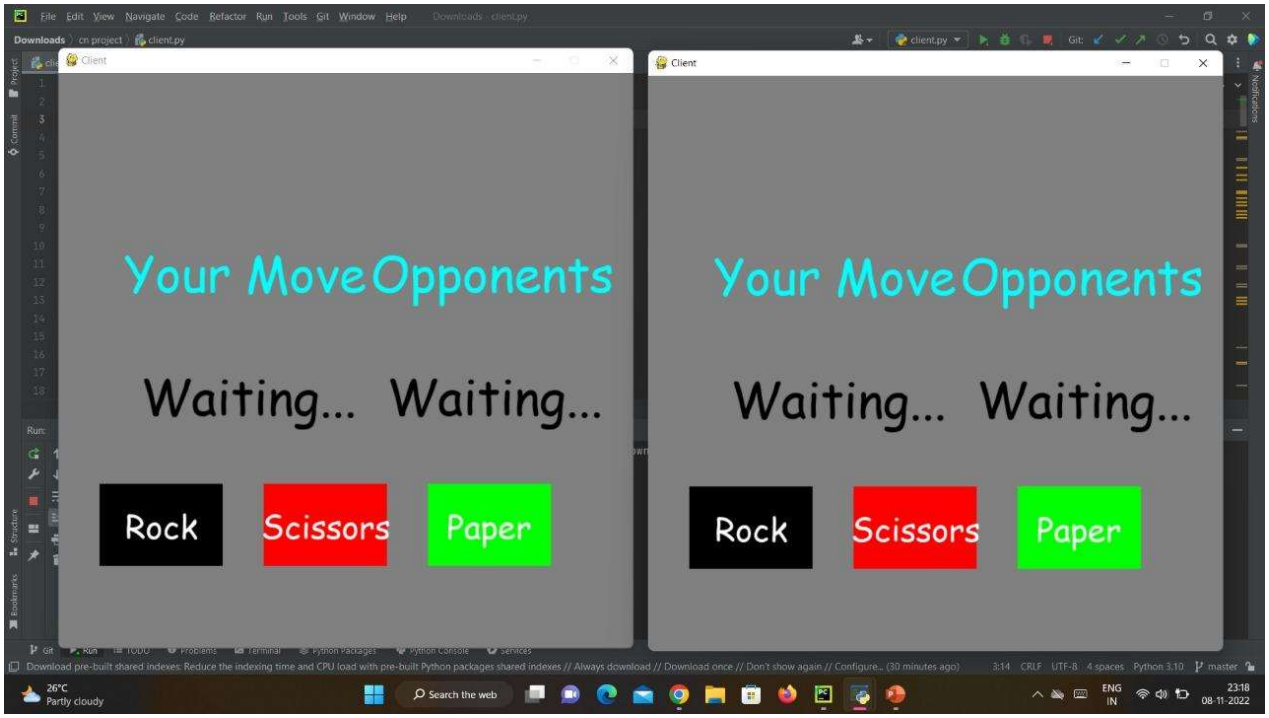
```
class Game:
    def __init__(self, id):
        self.p1Went = False
        self.p2Went = False
        self.ready = False
        self.id = id
        self.moves = [None, None]
        self.wins = [0,0]
        self.ties = 0

    def get_player_move(self, p):
        """
        :param p: [0,1]
        :return: Move
        """
        return self.moves[p]

    def play(self, player, move):
        self.moves[player] = move
        if player == 0:
            self.p1Went = True
        else:
            self.p2Went = True

    def connected(self):
        return self.ready

    def bothWent(self):
        return self.p1Went and self.p2Went

    def winner(self):

        p1 = self.moves[0].upper()[0]
        p2 = self.moves[1].upper()[0]

        winner = -1
        if p1 == "R" and p2 == "S":
            winner = 0
        elif p1 == "S" and p2 == "R":
            winner = 1
        elif p1 == "P" and p2 == "R":
            winner = 0
        elif p1 == "R" and p2 == "P":
            winner = 1
        elif p1 == "S" and p2 == "P":
            winner = 0
        elif p1 == "P" and p2 == "S":
            winner = 1
```

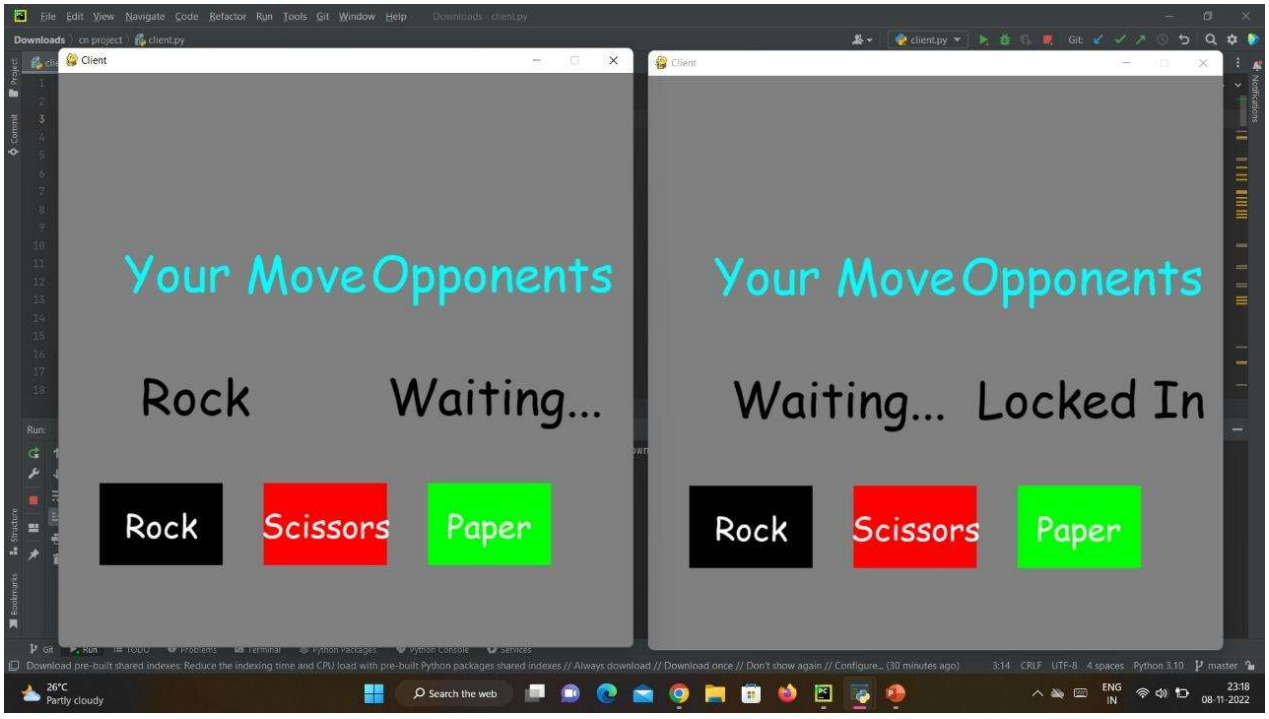
```
        return winner

    def resetWent(self):
        self.p1Went = False
        self.p2Went = False
```

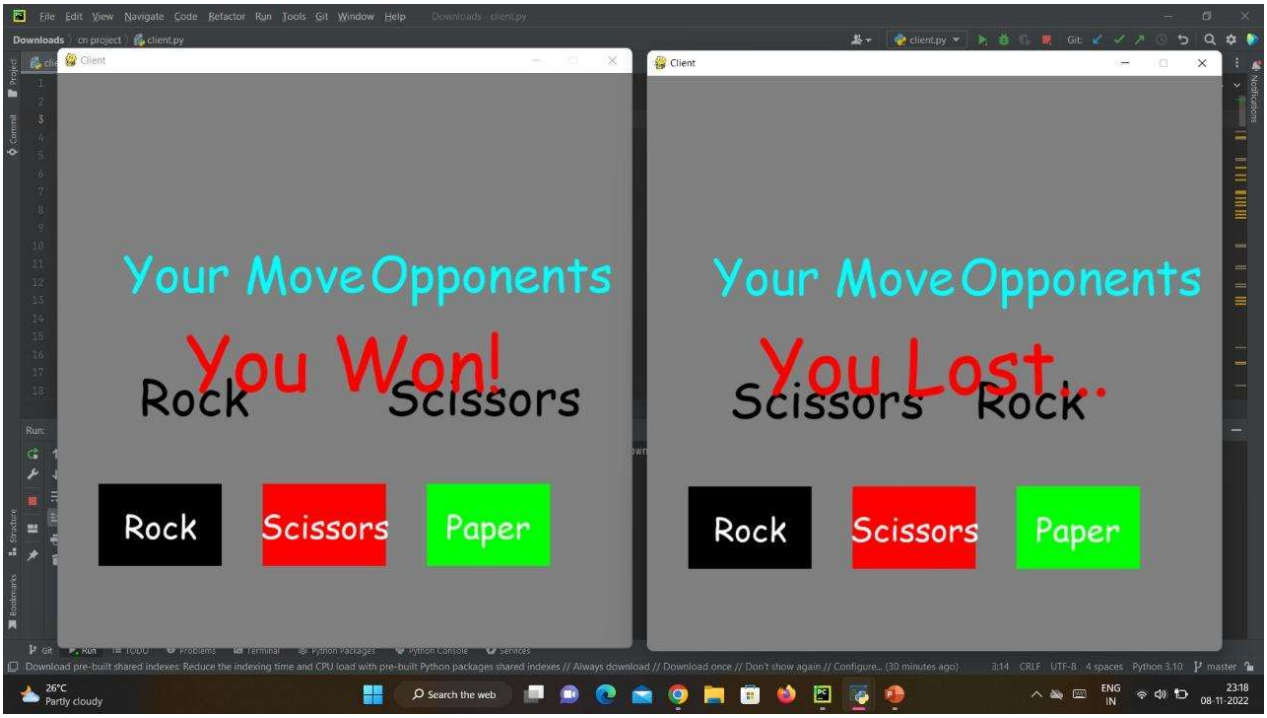
Network side:

```
import socket
import pickle


class Network:
    def __init__(self):
        self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server = "10.11.250.207"
        self.port = 5555
        self.addr = (self.server, self.port)
        self.p = self.connect()

    def getP(self):
        return self.p

    def connect(self):
        try:
            self.client.connect(self.addr)
            return self.client.recv(2048).decode()
        except:
            pass

    def send(self, data):
        try:
            self.client.send(str.encode(data))
            return pickle.loads(self.client.recv(2048*2))
        except socket.error as e:
            print(e)
```

7. Experiment Result & Outputs






Client
Your Move Opponents
You Won!
Rock
Scissors
Rock
Scissors
Paper
Client
Your Move Opponents
You Lost...
Scissors
Rock
Rock
Scissors
Paper

8. CONCLUSION & FUTURE ENHANCEMENT

As depicted through the above screenshots our model creates a back and through system which can be used to play the 'Rock, Papers, Scissors' game using servers as its source. We have used TCP server connections to perform the task. The package was designed in such a way that future modifications can be done easily. The following conclusions can be deduced from the development of the project:

i. Automation of the entire system improves efficiency.
ii. It provides a friendly GUI which proves to be better when compared to existing systems.
iii. It is easy to use.

From these points we can conclude that this could be an efficient system for playing games and because of its simplicity, can easily be built further on in multiple ways.

9. REFERENCES

1. **Section.io** :
https://www.section.io/engineering-education/rock-paper-scissor-online/
2. Stackoverflow
3.Ajith singh- Socket programming with python