SUDOKU SOLVER USING CLIENT SERVER MODEL

Report submitted to the SASTRA Deemed to be University
as the requirement for the course

CSE302: COMPUTER NETWORKS

Submitted by

NARRA CHANDANA
124003065 , B.Tech - CSE

December 2022

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled "**SUDOKU SOLVER USING CLIENT SERVER MODEL**" submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **Shri. Narra Chandana (Reg. No.124003065,CSE**) during the academic year 2022-23, in the School of Computing

Project Based Work *Viva voc*e held on _____

Examiner - I **Examiner - II**

List of Figures

Abbreviations

HTTP-HyperText Transfer Protocol

IP -Internet Protocol

GUI-Graphical User Interface

TCP - Transmission control Protocol

Abstract

Sudoku is a logic based,combinational number-placement puzzle. In classic sudoku, the objective is to fill a 9*9 grid with digits so that each column, each row, and each 3*3 grid contain all the digits from 1 to 9.

The web application is designed in such a way that it takes the input of unsolved sudoku from the client, and the input data is sent to the server. Client side web application is built by python GUI frameworks as they are helpful for the user to communicate easily.

The input data is sent to the server from the client using HTTP protocols at the application layer and TCP protocols at the transport layer. Data is made into data packets and transferred via the Internet.TCP protocols are used to acknowledge every data packet.

The application to solve sudoku is developed in python and lies in the server, takes the data as input, processes it and the output is produced. Simultaneously the user will be solving the sudoku.User's answer is cross checked with the output and the response is given whether the user entered numbers are correct or not through GUI.Else if user wants the complete solved sudoku,the complete output produced in the server is sent to the client.

If the user completely solves the sudoku, a message congratulating the user will be displayed, along with the time taken by the user to solve the sudoku. If the user enters an invalid sudoku, a message will be displayed showing that the entered sudoku is invalid. This data is transferred to the client via the same protocols used during request calls.

KEYWORDS:

- **Client server**
- **TCP**
- **HTTP**
- **Python**
- **Game solver**

Table of Contents

CHAPTER 1
INTRODUCTION

1.1Proposed work:

The Sudoku solver uses TCP/IP protocol to establish a connection between client and server.This application uses a single server and connects up to 3 clients. Any connections which exceed the limit won't be connected to the server.

When a player enters the sudoku to be solved and clicks done, the application on the client side sends the data to the server. Once the data is received by the server it will solve the given question and will return the solution of the sudoku that is entered by the client.

The application now allows you to play the sudoku game interactively. If any number is entered in such a way that mismatches the solution, the mismatched number will be shown in red color whereas, If the number entered is correct ,it is shown in black color.

Once the player finishes the game and if the answer is correct a message will be displayed congratulating the player. Player can also see the solution if he wants to, by clicking on the solution button. The server does the computations required to solve the sudoku. The server does not store or use any information of the client.

1.2Proposed Protocols Description:

TCP:

An application can exchange data over a network using the Transmission Control Protocol (TCP), a standard that outlines how to start and sustain a network conversation. TCP interacts with the Internet Protocol (IP), which establishes the rules for how computers exchange data packets.

- TCP is a reliable protocol.
- TCP will ensure that the data reaches the required destination in the same order it was sent

HTTP:

In the network protocol stack, HTTP is an application layer protocol that sits on top of the other layers and is used to convey data between networked devices.In a typical HTTP flow, a client machine sends a request to a server, which then sends back a response message.

- Stateless protocol: It is incapable of processing transactions in memory.
- Flexible - The HTTP protocol allows the sending of any kind of data.
- One request will be accepted per connection as it is a connectionless protocol.

1.3Limitation of the protocols :

- There is no data privacy in HTTP as anyone can access the data.
- Speed of data transfer is slow in TCP protocol when compared to UDP

1.4Working flowchart:

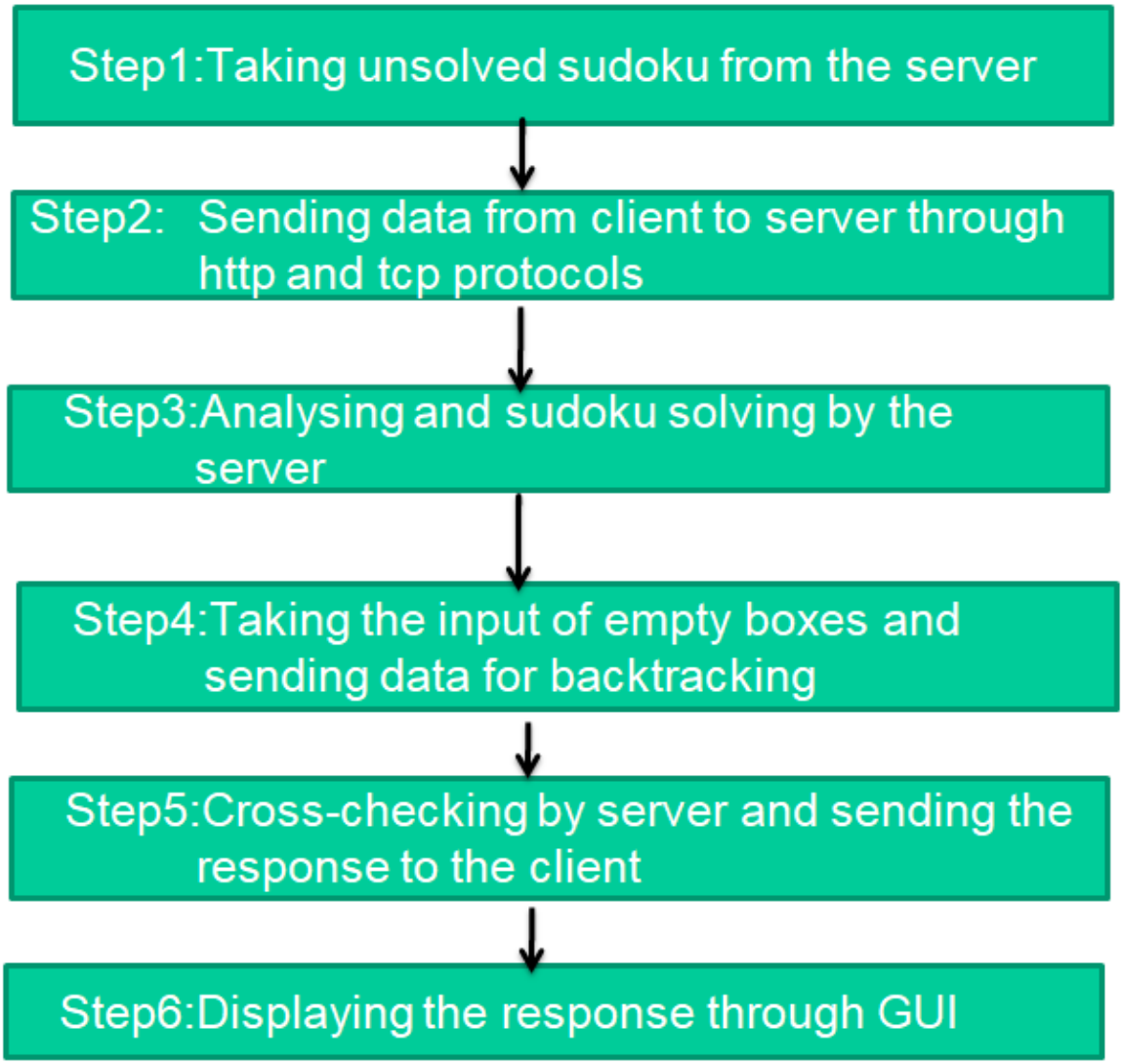

Fig 1.4 working flowchart

CHAPTER 2

RELATED WORK

2.1. Client-Server Model:

2.1.1.Client:

A Client is an individual or organization that uses the Service. In the IT context, a client is a computer/device, also called a host, that actually uses services or accepts information. Client devices include laptops, workstations, IoT devices, and similar network-enabled devices.

2.1.2.Server:

In the realm of IT, a server is a distant computer that offers access to data and services.Servers are typically physical devices like rack servers, but with the rise of cloud computing, virtual servers are entering the equation. Servers manage operations including email, hosting for applications, Internet connectivity, and printing.

2.1.3.Client-Server Architecture:

Client-server architecture refers to systems that host, provide, and manage most of the resources and services requested by clients. In this model, all requirements and services are delivered over the network, also known as the network computing model or client-server network.

Client-server architecture, also known as the client-server model, is a networking application that divides tasks and workloads between clients and servers that reside on the same system or are connected by a computer network.

A client/server architecture typically consists of multiple users' workstations, PCs, or other devices connected to a central server through an Internet connection or other network. A client sends a request for data, a server accepts the request, processes it, and sends the data packet back to the user who needs it.

Briefly summarized,

- First the client sends a request through a network capable device.
- Second, the network server accepts and processes the user's request.
- The response is then returned to the client by the server.

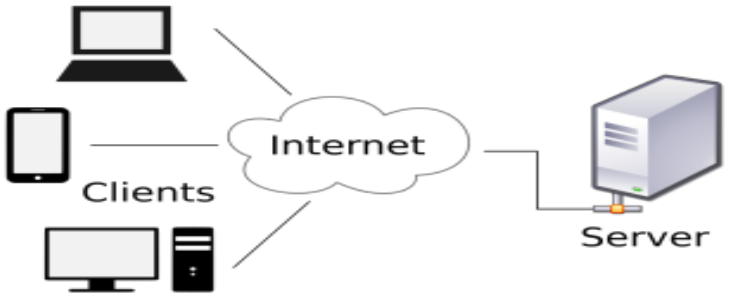

Fig.2.1 Client-Server Model

2.2 Client-Server System Structure:

The primary function of the operating system and communication software is to support distributed applications by offering a fundamental framework. The distribution of task-level applications across the client and server forms the basis of client-server system architecture. The general scenario of the mode is shown in the figure below. Software for client-server communication forms the basis of trade.. An example of this kind of software is TCP/IP.

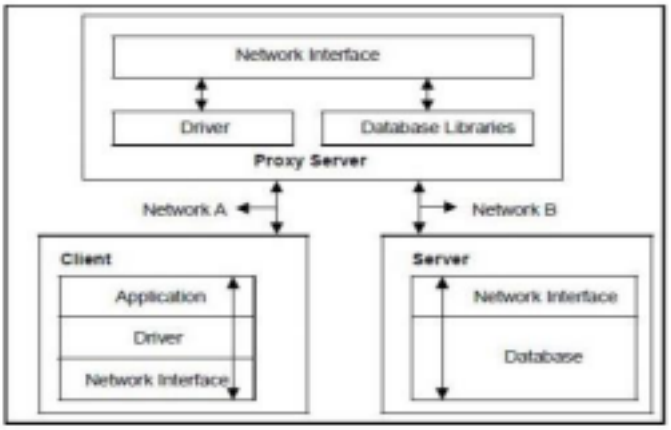

Fig.2.2 Client-Server System Structure

2.3.Sockets and Socket Programming:

2.3.1.Sockets:

Sockets in computer networks are used to transfer information between two processes on the same or different systems in the network.A socket is a communication channel between numerous processes that combines an IP address and a software port number. Sockets help you find the address of the application to send data to based on the IP address and port number.

2.3.2.Socket programming:

In order for two nodes on a network to communicate with one another, socket programming must be used.One socket (host) listens on a specific port on an IP address, and another socket accesses another socket to form a connection.The server creates a listener socket while interacting with the client. They are the true foundation of web browsing.There is a server and a client, to put it simply. Socket programming starts with importing the socket library and creating a simple socket.

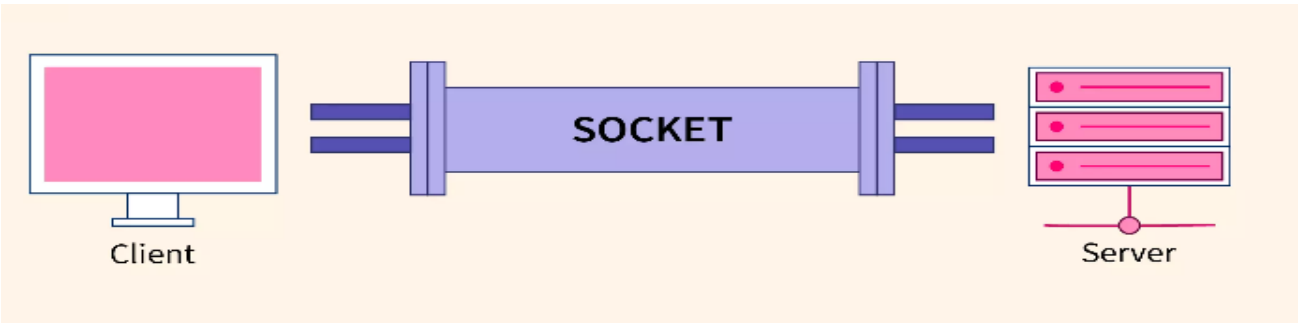

Fig.2.3 Socket Programming

2.4Connection oriented service:

Basically, Connection-Oriented Service is a method that is often used to communicate and transfer data at the session layer. The recipient receives the data streams or packets in the same order as they were sent by the sender.It is basically a technique for transferring data between two computers or devices connected to a different network that was created and modeled after the telephone system. When a network uses this service, it sends or transfers data or messages in the appropriate sequence and format from the sender or source to the receiver or destination.

An illustration of a connection-oriented protocol is TCP. A logical connection between the two processes must be established before data can be transmitted. The link must be maintained during all communication steps before being shut off afterward.During all phases of communication, the link must be maintained, then it must be cut off thereafter.

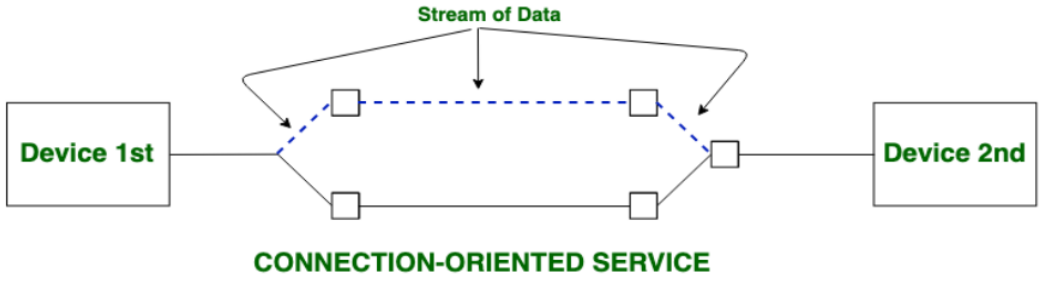

Fig 2.4 connection oriented service

2.5Connectionless Service:

Network-free service simply indicates that even without a session connection to the receiver, a node can transport or send data packets or messages to that node. Unplanned transmission or transfer of a message occurs. The reason this typically works is because error handling protocols allow and approve of the rectification of errors as well as the request for retransmission.

A connectionless protocol is UDP. Because it is comparable to sending a letter that you don't acknowledge receiving, it is known as a datagram protocol. Tftp and broadcasting are two examples of applications that leverage connectionless transport services.

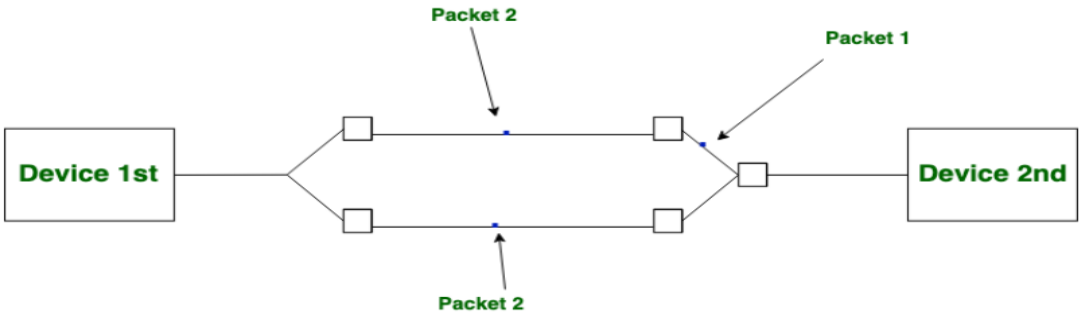

Fig 2.5 connectionless service

CHAPTER 3

SOURCE CODE

3.1.Client Code:

```
import socket

import pygame

from pygame.locals import *

import time

class sudoku:

    solution=[[0 for i in range(9)] for j in range(9)]

    sudoku=[[0 for i in range(9)] for j in range(9)]

    selected=None

    done=0

    start=None

    play_time=None

    final_time=None

def request(self):

        c=socket.socket()

        c.settimeout(10)

        try:
```

```
c.connect(('localhost',9999))
        except:
            print("Check your connection with server")
        print("Connected with server")
        c.send(bytes(str(self.sudoku),'utf-8'))
        print(c.recv(1024).decode())
        print("The solved sudoku is")
        try:
            self.solution=(eval(c.recv(1024).decode()))
        except:
            print("The solution does not exist")
        print(self.solution)
        for i in range(9):
            for j in range(9):
                if(s.solution[i][j]==0):
                    s.done=3
                else:
                    continue
    def gui(self):
```

```
    def format_time(secs):
sec = secs%60
        minute = secs//60
        mat = "Time  " + str(minute) + ":" + str(sec)
        return mat
    def intro():
        logo = pygame.image.load('logo.jpg')
        logo = pygame.transform.scale(logo, (450,200))
        developer_message=font1.render("Developed by",1,(0,0,0))
        name=font1.render("Narra Chandana",1,(0,0,0))
        reg=font1.render("124003065",1,(0,0,0))
        window.blit(developer_message,(170,420))
        window.blit(name,(160,450))
        window.blit(reg,(170,480))
        window.blit(logo,(0,0))
    colour=(250,250,250)
    pygame.init()
    width,height=450,450
    window=pygame.display.set_mode((width,height+70))
```

```
pygame.display.set_caption("Fun sudoku")
window.fill(colour)
font1=pygame.font.SysFont('Comic Sans MS',20)
font2=pygame.font.SysFont('Comic Sans MS',35)
correctnumber = pygame.font.SysFont("comicsans", 40)
wrongnumber = pygame.font.SysFont("comicsans", 40)
intro()
pygame.display.update()
time.sleep(5)
def draw_window():
    window.fill((250,250,250))
    for i in range(10):
        if(i%3==0):
            pygame.draw.line(window,(10,10,10),((width/9)*i,0),((width/9)*i,height),5)
            pygame.draw.line(window,(10,10,10),(0,(height/9)*i),(width,(height/9)*i),5)
        else:
            pygame.draw.line(window,(10,10,10),((width/9)*i,0),((width/9)*i,height),2)
            pygame.draw.line(window,(10,10,10),(0,(height/9)*i),(width,(height/9)*i),2)
    if(s.done==0):
```

```
    text1 = font1.render("Enter Sudoku you want to play", 1, (0,0,0))
    text2=font1.render("Click on Filled once you filled",1,(0,0,0))
    window.blit(text1,(0, 450))
    window.blit(text2,(0,482))
    done=font2.render("Filled",1,(0,0,0))
    window.blit(done,(355,455))
    pygame.draw.rect(window,(0,0,0),(350,455,100,50),2)
    for i in range(9):
        for j in range(9):
            if(s.sudoku[i][j]==0 or s.sudoku[i][j]==None):
                continue
            else:
                number=correct number.render(str(s.sudoku[i][j]), 1, (0,0,0))
                window.blit(number,((j*50)+15,(i*50)-5))
elif(s.done==1):
    done=font1.render("Solution",1,(0,0,0))
    window.blit(done,(360,465))
    s.play_time = round(time.time() - s.start)
    pygame.draw.rect(window,(0,0,0),(350,455,100,50),2)
```

```
time_message=font2.render(format_time(s.play_time), 1, (0,0,0))
# window.blit(time_message,(10, 460))
total=True
for i in range(9):
    for j in range(9):
        if(s.sudoku[i][j]==s.solution[i][j]):
            continue
        else:
            total=False
    if total==False:
        break
if(total==True):
    if(s.final_time==None):
        s.final_time=time_message
    window.fill((250,250,250))
    nosol=font2.render("Congrats!!",1,(0,0,0))
    window.blit(nosol,(150,200))
    window.blit(s.final_time,(150,245))
    s.selected=None
```

```
    else:


        for i in range(9):
            for j in range(9):
                if(s.sudoku[i][j]==0 or s.sudoku[i][j]==None):
                    continue
                elif(s.sudoku[i][j]==s.solution[i][j]):
                    number=correctnumber.render(str(s.sudoku[i][j]), 1, (0,0,0))
                    window.blit(number,((j*50)+15,(i*50)-5))
                else:
                    number=wrongnumber.render(str(s.sudoku[i][j]), 1, (250,0,0))
                    window.blit(number,((j*50)+15,(i*50)-5))
elif(s.done==2):
    for i in range(9):
        for j in range(9):
            number=correctnumber.render(str(s.solution[i][j]), 1, (0,0,0))
            window.blit(number,((j*50)+15,(i*50)-5))
    final_message=font2.render("Final solution...", 1, (0,0,0))
    window.blit(final_message,(10, 460))
```

```
    elif(s.done==3):
        window.fill((250,250,250))
        nosol=font2.render("No solution exists!!",1,(250,0,0))
        window.blit(nosol,(80,200))
        s.selected=None
    if(s.selected!=None):
        pygame.draw.rect(window,(250,0,0),(s.selected[0]*50,s.selected[1]*50,50,50),3)
def click_input():
    draw_window()
    run=True
    while run:
        draw_window()
        for event in pygame.event.get():
            key=None
            if event.type == pygame.QUIT:
                run = False
            if(event.type == pygame.KEYDOWN):
                print("Key entered detected")
                if event.key == pygame.K_1:
```

```
    key = 1
if event.key == pygame.K_2:
    key = 2
if event.key == pygame.K_3:
    key = 3
if event.key == pygame.K_4:
    key = 4
if event.key == pygame.K_5:
    key = 5
if event.key == pygame.K_6:
    key = 6
if event.key == pygame.K_7:
    key = 7
if event.key == pygame.K_8:
    key = 8
if event.key == pygame.K_9:
    key = 9
if event.key == pygame.K_DELETE:
    key = None
```

```
                s.sudoku[s.selected[1]][s.selected[0]]=key
                print(key,"appended")
            if event.type == pygame.MOUSEBUTTONDOWN:
                pos = pygame.mouse.get_pos()
                if((pos[0])>350 and (pos[0]<450) and (pos[1]>455 and pos[1]<505)):
                    print("Done working")
                    s.done=s.done+1
                    if(s.done==1):
                        s.request()
                        s.start=time.time()
                elif(int(pos[0]/50)<9 and int(pos[1]/50)<9):
                    print("boxes working")
                    print("you clicked on ",int(pos[0]/50),int(pos[1]/50))
                    s.selected=(int(pos[0]/50),int(pos[1]/50))
            pygame.display.update()
    click_input()
    pygame.quit()
s=sudoku()
# s.request()
```

```
s.gui()
```

3.2. Sudoku Brute:

```
# N is the size of the 2D matrix N*N

N = 9

# Checks whether it will be legal to assign num to the given row, col

def isSafe(grid, row, col, num):

    for x in range(9):

        if grid[row][x] == num:

            return False

    for x in range(9):

        if grid[x][col] == num:

            return False

    startRow = row - row % 3

    startCol = col - col % 3

    for i in range(3):

        for j in range(3):

            if grid[i + startRow][j + startCol] == num:

                return False

    return True
```

```
# Takes a partially filled-in grid and attempts to assign values to all unassigned locations

# Sudoku solution (non-duplication across rows,columns, and boxes) */

def solveSudoku(grid, row, col):

        if (row == N - 1 and col == N):

                return True

        if col == N:

                row += 1

                col = 0

        if grid[row][col] > 0:

                return solveSudoku(grid, row, col + 1)

        for num in range(1, N + 1, 1):

                if isSafe(grid, row, col, num):

                        grid[row][col] = num

                        if solveSudoku(grid, row, col + 1):

                                return True

                grid[row][col] = 0

        return False
```

3.3.Server Code:

```
import sudokubrute as sbf
```

```
import socket
import multiprocessing
import time
class sudoku_solver:
    sudoku=[]
    #This function of the sudoku class takes the sudoku list and displays on the screen
    def display(self):
        print("The solved sudoku is")
        for i in range(9):
            for j in range(9):
                print(self.sudoku[i][j],end=" ")
            print()
    #This functions checks the number whether it is  present in the given row or not
    def not_inrow(self,row,value,occurrence=0):
        occ=0
        for i in range(9):
            if(self.sudoku[row][i]==value):
                occ=occ+1
            else:
```

```
            continue
    if(occ<=occurrence):
        return True
    else:
        return False
#This function checks the number whether the number is present in the given column or not
def not_incoloumn(self,column,value,occurrence=0):
    occ=0
    for i in range(9):
        if(self.sudoku[i][column]==value):
            occ=occ+1
        else:
            continue
    if(occ<=occurrence):
        return True
    else:
        return False
#This function checks whether the given number is present in the 3*3 matrix in an order
def not_inbox(self,row,column,value,occurrence=0):
```

```
    occ=0
    r=int(row/3)
    c=int(column/3)
    for i in range(3):
        for j in range(3):
            if(self.sudoku[(3*r)+i][(3*c)+j]==value):
                occ=occ+1
            else:
                continue
    if(occ<=occurrence):
        return True
    else:
        return False
#This function is designed to fill the all the empty boxes with the basic rules of the sudoku game
def basic_filling(self):
    for i in range(9):
        for j in range(9):
            if(self.sudoku[i][j]==0):
                repeat=0
```

```
            value=0
            for k in range(1,10):
                if((self.not_inrow(i,k)) and self.not_incoloumn(j,k) and self.not_inbox(i,j,k)):
                    repeat=repeat+1
                    value=k
                else:
                    continue
            if(repeat==1):
                self.sudoku[i][j]=value
    #This checks the rows columns and boxes simultaneously so that different possibilities can be eliminated and the boxes can be filled as the game advances
    def mainfilling(self):
        for i in range(9):
            for j in range(9):
                if(self.sudoku[i][j]==0):
                    for k in range(1,10):
                        if(self.not_inbox(i,j,k)):
                            r=int(i/3)
                            c=int(j/3)
                            count1=0
```

```
                    count2=0

                    for p in range(3):

                        if(c+p!=j and (not(self.not_incoloumn(c+p,k)) or (self.sudoku[r][c+p]!=0 and
self.sudoku[r+1][c+p]!=0 and self.sudoku[r+2][c+p]!=0))):count1=count1+1

                        if(r+p!=i and (self.sudoku[r+p][j]!=0 or
not(self.not_inrow(r+p,k)))):count1=count1+1

                        if(r+p!=i and (self.not_inrow(r+p,k) or (self.sudoku[r+p][c]!=0 and
self.sudoku[r+p][c+1]!=0 and self.sudoku[r+p][c+2]!=0))):count2=count2+1

                        if(c+p!=j and (self.sudoku[i][c+p]!=0 or
not(self.not_incoloumn(c+p,k)))):count2=count2+1

                    if(count1==4 and count2==4):self.sudoku[i][j]==k

    #This function checks whether there is any empty space in the sudoku

    def check_zero(self):

        for i in range(9):

            for j in range(9):

                if(self.sudoku[i][j]==0):

                    return True

    #This function takes the input of the row column and the value and it backtracks i.e it gives the
output of whether it is correct or not

    def check_sudoku(self):

        for i in range(9):

            for j in range(9):
```

```
            if(self.sudoku[i][j]==0):
                continue
            else:
                if(self.not_inrow(i,self.sudoku[i][j],1) and self.not_incoloumn(j,self.sudoku[i][j],1) and
self.not_inbox(i,j,self.sudoku[i][j],1)):
                    continue
                else:
                    return False
        return True
sudoku=sudoku_solver()
s=socket.socket() #ipv4,tcp
#Socket program takes the 2 arguments types of ip address and type of connection(TCP/UDP)
#Socket program by default accepts the ipv4 address and TCP connection
s.bind(('localhost',9999))
s.listen(3)
print("Waiting for the connections")
while(1):
    c,address=s.accept()
    print("Connected with the",address)
    sudoku.sudoku=(eval(c.recv(1024).decode()))
```

```
print("Data is received on server side successfully")
print(sudoku.sudoku)
c.send(bytes("Data is received on server side successfully","utf-8"))
if(sudoku.check_sudoku()):
    print("Sudoku is valid")
    for i in range(3):
        sudoku.basic_filling()
        sudoku.mainfilling()
    if(sudoku.check_zero):
        if (sbf.solveSudoku(sudoku.sudoku,0,0)):
            sudoku.display()
            print("The sudoku is solved\n\n")
        else:
            print("No solution exists")
    else:
        sudoku.display()
        print("The sudoku is solved\n\n")
    print("sending the solved sudoku")
else:
```

```
    print("Sudoku is invalid")
    print("sending the same unsolved sudoku")
print(sudoku.sudoku)
c.send(bytes(str(sudoku.sudoku),'utf-8'))
c.close()
```

CHAPTER 4
SNAPSHOTS

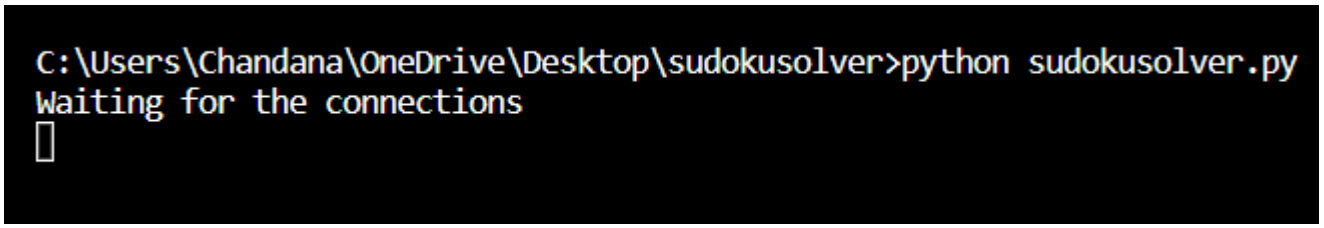

Fig 4.1 starting the server

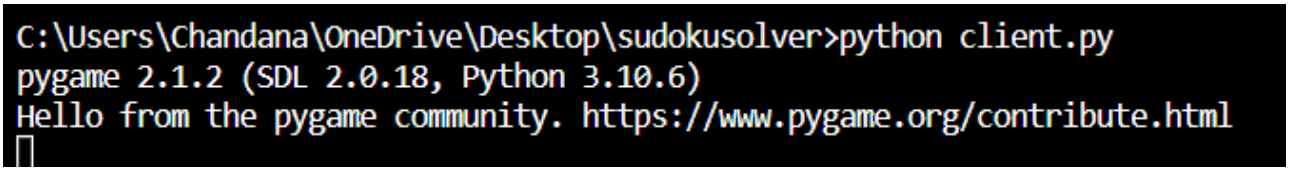

Fig 4.2 starting a client

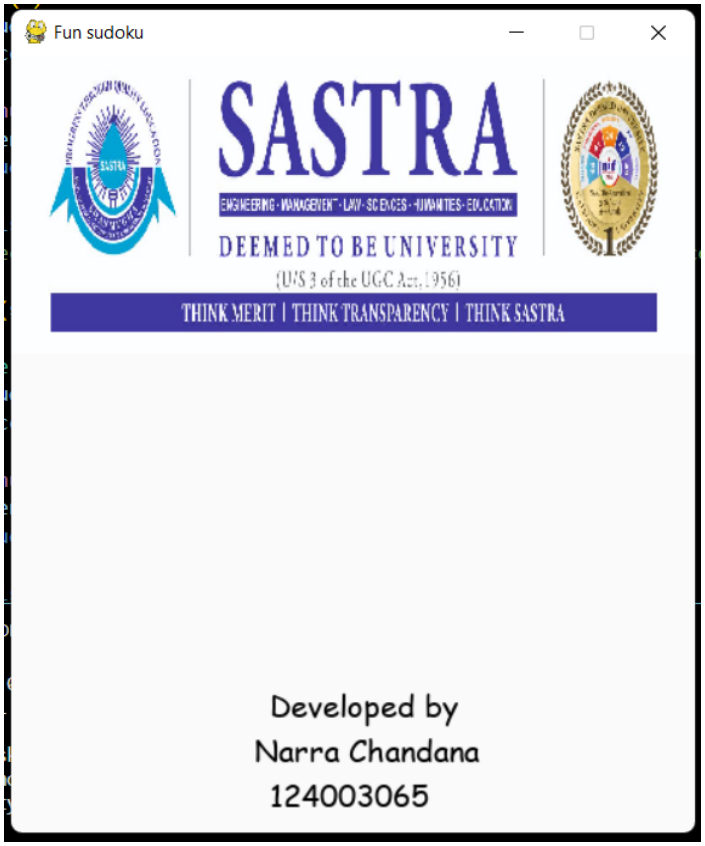

Fig 4.2 player's GUI after connecting to the server

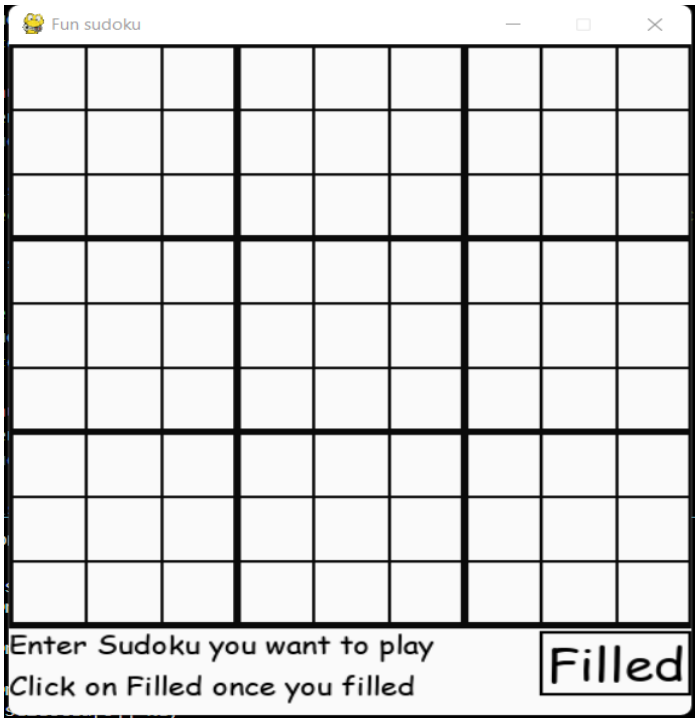

Fig 4.3 sudoku to take input from player

Fig:4.4 Taking input from the player

```
C:\Users\Chandana\OneDrive\Desktop\sudokusolver>python client.py
pygame 2.1.2 (SDL 2.0.18, Python 3.10.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
boxes working
you clicked on  0 0
Key entered detected
5 appended
boxes working
you clicked on  1 2
Key entered detected
8 appended
boxes working
you clicked on  2 2
Key entered detected
2 appended
boxes working
you clicked on  5 0
Key entered detected
1 appended
boxes working
```

Fig:4.5 code working in the backend

Fig:4.6 player playing sudoku

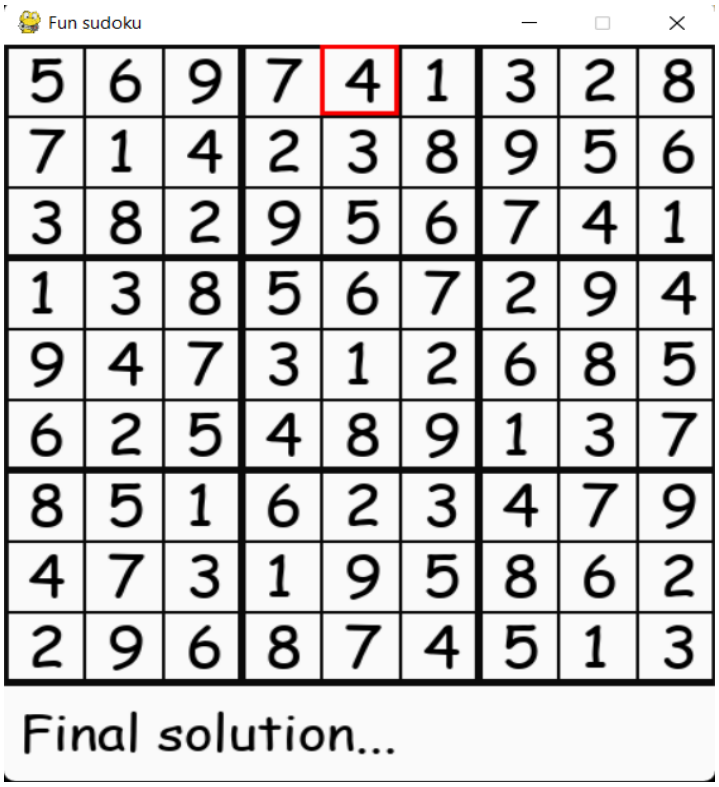

Fig:4.7 solution of sudoku entered by the player

Fig:4.8 GUI when player completes playing sudoku

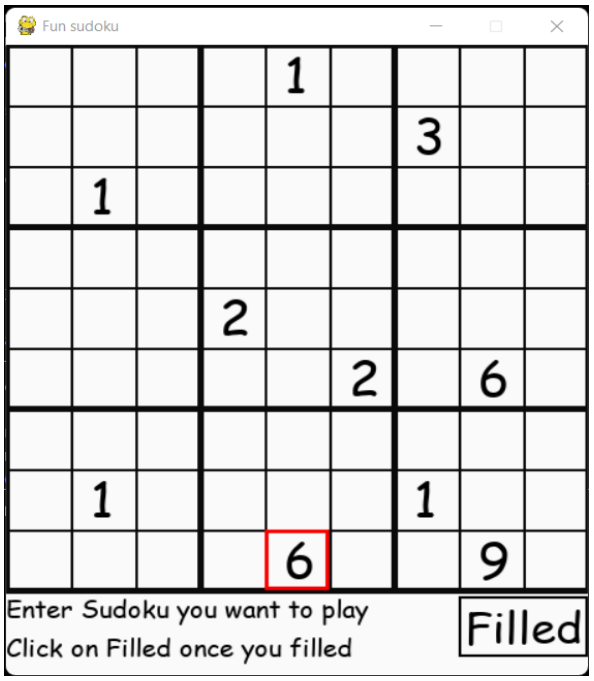

Fig:4.9 user entering wrong sudoku

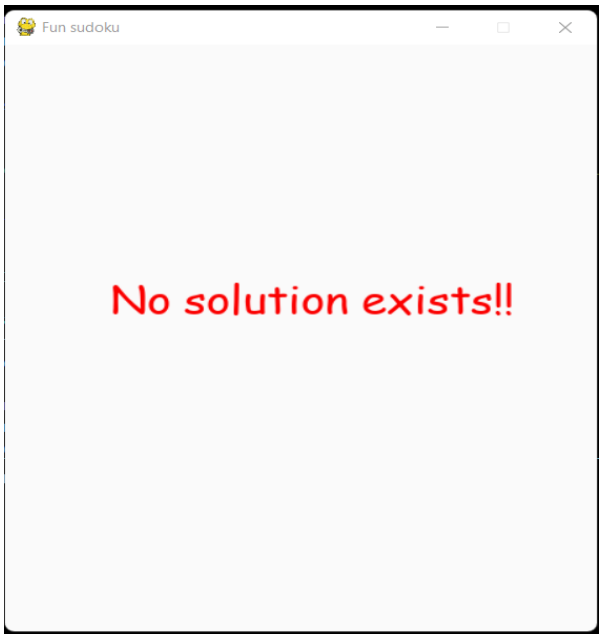

Fig 4.10 GUI showing no solution exists

5. CONCLUSION & FUTURE WORK

5.1.Conclusion:

Sudoku is a most commonly played game all over the world which makes users' brains smart and healthy. Sudoku solvers help users to play the game interactively. In this client server sudoku solver, solving of sudoku using tcp protocol along with the ability of the user to play the game if they wish to is implemented successfully. Main advantage of this project is giving users a good gaming environment to play along with the solution of the game provided.

5.2.Future Work:

In order to optimize the current application, we can help the user while solving sudoku by giving hints. We can also add a score system based on time and accuracy which makes users more enthusiastic to play. We can also create sudoku puzzles of various levels such as hard, medium and easy which will be given to the user based on the selection of the level.

6.REFERENCES

- https://en.wikipedia.org/wiki/Sudoku_solving_algorithms
- https://www.geeksforgeeks.org/client-server-model/
- <http://en.wikipedia.org/wiki/Mathematics_of_sudoku>
- James F. Kurose, and Keith W. Ross. Computer Networking: A Top-down Approach. Pearson Education, Seventh Edition, 2017.