Normunds Brižs
Zanis Ali Khan (B77407)
Frozan Maqsoodi

Sudoku Solver Application

(Distributed Systems homework 2)

1. Introduction

The goal of this homework is for us advance the application we created in HW1, based on a new communication model, where multiple people can solve the same Sudoku at the same time. In our application (HW2) we have implemented indirection communication using Rabbitmq for implementation and we removed sockets part and used Remote Objects. The application allows multiplayers to solve the Sudoku and have their scores calculated based on their moves.

2. Sudoku

Sudoku is a puzzle that originated in Switzerland. The man credited with the creation of Sudoku is Leonhard Euler, a famous mathematician. He has worked with many different mathematical problems. Leonhard Euler is the man responsible for discovering the ratio of a circle's circumference and diameter known as the Greek letter **π** and **i** for the square root of -1.

Leonhard Euler

Portrait by Jakob Emanuel Handmann (1753)

https://en.wikipedia.org/wiki/Leonhard_Euler

Normunds Brižs
Zanis Ali Khan (B77407)
Frozan Maqsoodi

When developing the basics of Sudoku, Euler made puzzles called ‘Graeco-Roman Squares’ or Latin Squares, which instead of numbers used letters from Greek alphabet and Latin alphabet but these puzzles still had the same concept as Sudoku where the letters in each row and column could appear only once. The board of Latin Squares wasn’t divided into regions as the Sudoku puzzles we know now are.

A α	B γ	C δ	D β
B β	A δ	D γ	C α
C γ	D α	A β	B δ
D δ	C β	B α	A γ

Latin Squares

http://www.sudokudragon.com/sudokuhistory.htm

The original Sudoku’s year of origin is not clear, but the modern Sudoku started to become popular in the late 20th century. Although people weren’t interested in these puzzles, they later still spread to other countries and were changed multiple times over the years. In 1979 a new puzzle called ‘Number Place’ was published in New York. It introduced a new concept. In Roman Squares You must have a unique occurrence of each number in each row and column, but in this puzzle the 9x9 board was divided into 3x3 sections and the new rule was that the same number shouldn’t appear twice in each of these sections which is basically the sudoku we know now. Sudoku got its name when it was brought from America to Japan. At first it was translated as 数字わ独身に限る (Suuji wa dokushin ni kagiru) which roughly means ‘the numbers must occur once only’ but later got simplified into 数独 meaning ‘number only’ or ‘number single’ (source: http://www.sudokudragon.com/sudokuhistory.htm). There have been other elements implemented since then, but that’s when the modern Sudoku became what it is known as now.

Normunds Brižs
Zanis Ali Khan (B77407)
Frozan Maqsoodi

The general rules of Sudoku are fairly simple. It is a logic-based number placement puzzle and has a grid consisting of 9 rows and 9 columns, which is divided into 9 3x3 boxes and you must fill the whole board with numbers from 1 to 9. The numbers must be placed so that they occur only once in each row, column and 3x3 section.

Sudoku board and solution

https://en.wikipedia.org/wiki/Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

3. Indirect Communication

Indirect communication is defined as communication between entities in a distributed system through an intermediary with no direct coupling between senders

and receivers. Indirect communication avoids the direct coupling by inheriting two main properties:

Space uncoupling: In space uncoupling, the sender doesn't know and need not to know the receiver identity or vice versa. This provides the system developers flexibility of dealing with change, such as participants can be replaced, updated, replicated or migrated.

Time uncoupling: In time uncoupling scenario, the sender and receiver do not need to communicate at the same time

4. Rabbitmq

RabbitMQ is an open source message broker software (sometimes called message-oriented middleware) that originally implemented the Advanced Message Queuing Protocol (AMQP) and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol (STOMP), MQTT, and other protocols. As AMQP is a two-way RPC protocol where the client can send requests to the server and the server can send requests to a client, Pika implements or extends IO loops in each of its asynchronous connection adapters. These IO loops are blocking methods which loop and listen for events. Each asynchronous adapters follows the same standard for invoking the IO loop. The IO loop is created when the connection adapter is created. As an example, In our application we used the following steps by importing the pika library first the code is shown in the figure 1:

a. We start by creating our connection object, then starting our event loop.
b. When we are connected, we create a channel.
c. When the channel is created, we declare a queue.

http://www.tutorialspoint.com/data_communication_computer_network/client_server_model.htm

d. When the queue is declared successfully, we declare channel.basic_consume.

```
class SudokuServer():

        def create_queue(self):
        """
        One queue for all client requests
        """
        self.connection = pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
        self.channel = self.connection.channel()
        self.channel.queue_declare(queue='sudoku')

        self.channel.basic_qos(prefetch_count=1)
        self.channel.basic_consume(self.on_request, queue='sudoku')
```

Figure 1: Pika example

5. Python

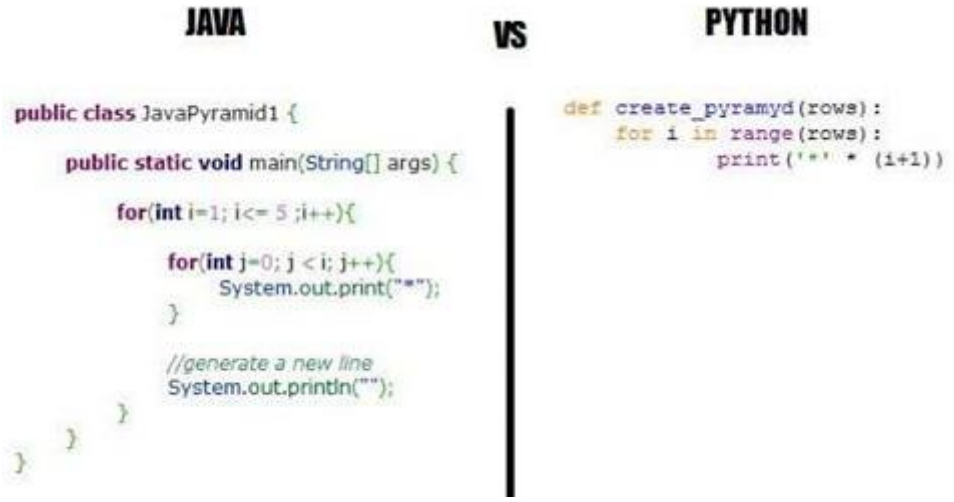
Python is a well-known and popular programming language. It was designed by Guido van Rossum and published in 1991. Python emphasizes on readability. It uses whitespace indentations instead of keywords or curly brackets for code blocks. The code is also more understandable and requires fewer lines of code than some other programming languages. Python supports different paradigms like functional programming, object-oriented programming and procedural styles. It can run on different systems because Python interpreters are available for many different operating systems. Some other paradigms, including logic programming are also available for Python.

The implementation of Python began in 1989 and the 2.0 version of Python was released in 2000, which featured Unicode support. After this version the development process started to become more community-based. After a long testing time Python 3.0 was released, which was a major release for Python. It was initially described as Python 3000 or py3k. Since Python 3.0 was backward-

incompatible, many of its features had to be backported to Python 2.6.x and 2.7.x version series.

The principles of Python are summarized by the document called the 'Zen of Python'. Some of the principles are:

- Beautiful is better than ugly.
- Simple is better than complex.
- Explicit is better than implicit.
- Flat is better than nested.
- Readability counts.



Example of code for a program in Java and code for the same program in Python

https://www.quora.com/If-I-want-to-be-a-software-engineer-at-Google-what-are-the-main-programming-languages-I-have-to-master

- Complex is better than complicated.

6. Tkinter

Tkinter is the most commonly used GUI programming toolkit for Python. It is free software, which was released under a Python license. It is also said to be Python's *de facto* standard GUI. And it is the GUI programming toolkit that is used in our homework. There are alternatives to Tkinter for Python, such as Pyglet, wxPython and PyQt.

Here are some definitions for terms in Tkinter:

- Window – a rectangular area on the user's screen.
- Top Level Window – a window that has the standard frame, can be moved and resized and exists independently on the user's screen.
- Widget – areas like buttons, text fields, labels or frames that make an application in a GUI.

7. Proposed work

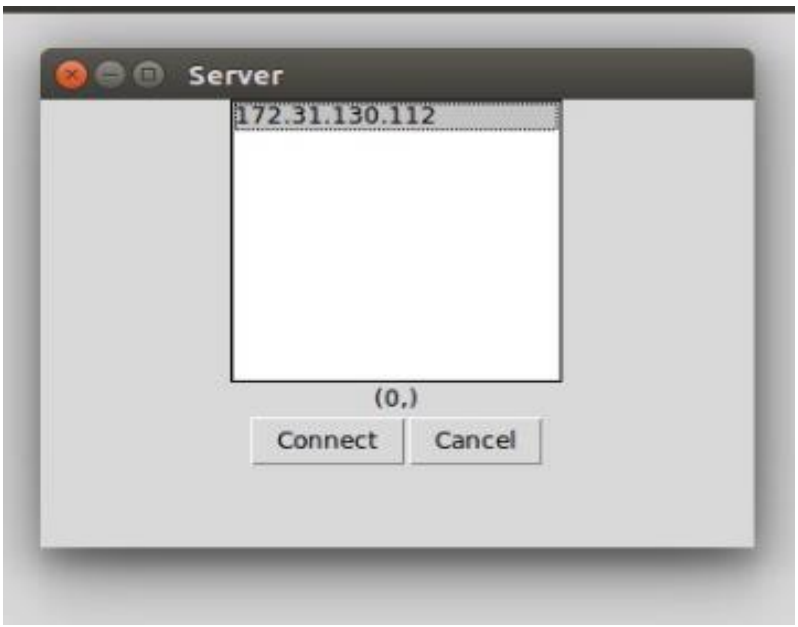
The homework task is to program an application that can allow Sudoku solving for at least 2 players over a network. The main objectives of the homework are to implement any indirect communication technique, create a network-based multiplayer game and to make the solving concurrent. The application should also keep the player's score. A player gains a point for each correct guess if the same guess hasn't been made yet. Players lose a point for an incorrect guess.

The server for the homework should create and handle game sessions, handle player connections, record player activities and count points for each player.

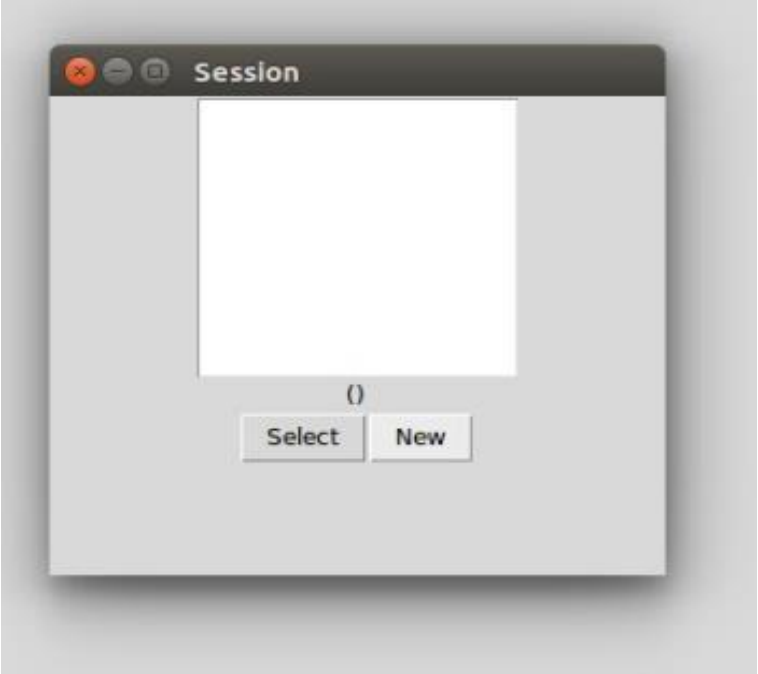
The client of the application should handle user input: request a nickname, server address and whether the user would like to join a session or start a new one. It should also render the game and score for each player and handle connection to the server.

8. Screens of the application

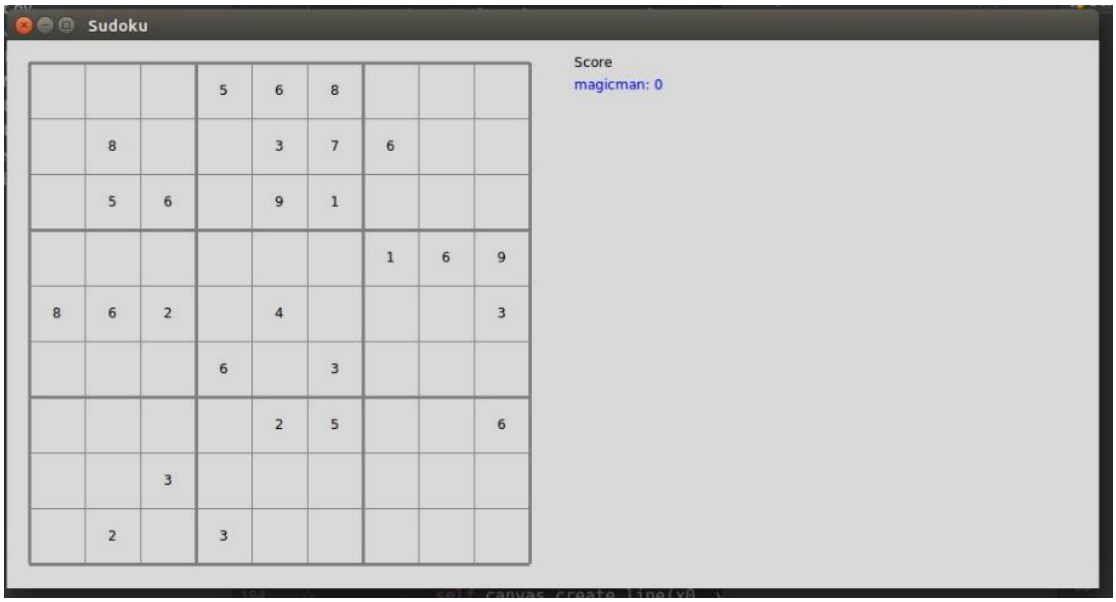
The first screen from the game when running the client shows the nickname input screen. The player's nickname must be between 5 and 8 characters long. After entering a nickname, click on Game Start button to begin a Sudoku game. If you are unsure about the game rules, click on 'How to Play' button, this will open a web page of Sudoku game rules.



The second screen is the Server window, which allows you to select a server. You can select the server address (e.g. 172.31.130.112) and click on ‘Connect’ button to join a game session or click on ‘Cancel’ button to end without joining any session.



The third screen is selecting a game Session, If entering for the first time, the window session will look empty because there is session started yet. You can start a new session by clicking on ‘New’ button else you can select any existing session by clicking on the ‘Select’ session.



After successfully creating a new session or joining any existing session, the final screen of Sudoku will appear with the player name and player score on the right. Any second player can join this existing game session.

9. Running the game

Before running the game make sure you have setup the environment ready, read the following steps:

1. Rabbitmq-server should be installed and configured. For liinux, rabbitmq.config file is uploaded on github (for guest access), copy that file to /etc/rabbitmq folder of linux. Then start the server again.
2. Install Tkinter library, you can use the command (*sudo apt-get install tk-dev python-tk*)
3. Other libraries used in the package are json, pika

After you have the environment set up, you need 2 files to play this game: Server.py, and Client.py. In order to play the game, the first thing you need to do is run the server file, either in a Python IDE or by going to the directory, where the game files are located, open a command prompt or a terminal and run it by entering the name of the server file (Server.py). Then you can run the client file the same way as with the server file (Client.py). A dialogue box will open asking for a player nickname. There you have to input the nicknames. They have to be 5 to 8 characters long. After entering the nickname, the server window will open where you need to select the server address to enter a game session. You can create a new session to start the game or join an existing session to play against another player. More details can be found at the Application screens section

10. Result and conclusion

Although the application was not easy to make and it took us a lot of time to fix what we could, we tested it using Server on linux machine and running one Client on Windows. Setting rabbimq on Windows was quite a challenge. It worked fine. The client input is active throughout the whole game and it counts points as intended. Scores are visible to players, the application keeps open the communication channel with the sudoku server, the players can also leave in the middle of the game.

References

https://en.wikipedia.org/wiki/Sudoku

http://www.sudokudragon.com/sudokuhistory.htm

https://en.wikipedia.org/wiki/Leonhard_Euler

https://en.wikipedia.org/wiki/Python_(programming_language)

Normunds Brižs
Zanis Ali Khan (B77407)
Frozan Maqsoodi

https://www.python.org/doc/essays/blurb/

https://en.wikipedia.org/wiki/Zen_of_Python

https://www.quora.com/If-I-want-to-be-a-software-engineer-at-Google-what-are-the-main-programming-languages-I-have-to-master

https://en.wikipedia.org/wiki/RabbitMQ

https://pika.readthedocs.io/en/0.10.0/intro.html