RYERSON UNIVERSITY

Faculty of Engineering, Architecture and Science

Department of Electrical and Computer Engineering

Course Number	892
Course Title	Distributed and Cloud Computing
Semester/Year	W2024

Instructor	Dr. Muhammad Jaseemuddin

Lab No.	3

Lab Title	RabbitMQ

Submission Date	March 17th, 2024
Due Date	March 17th, 2024

Student Name	Student ID	Signature*
Abdulrehman Khan	500968727	A.K.

**By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work.*

Introduction

The objective of this lab is to develop a rover control system using gRPC for communication between the client, server, and deminer components. The system allows the client to request map data, commands for rover movements, mine serial numbers, and share mine PINs with the server. The server processes these requests, retrieves necessary information, and communicates with the deminer to disarm mines. This report outlines the implementation steps, main data structures, functions, and includes screenshots of the test run.

Implementation Overview:

The implementation consists of three main components: client, server, and deminer.

Client: Initiates requests to the server for map data, rover commands, and mine information. Upon receiving commands, it explores the map, detects mines, and shares mine PINs with the server.

Server: Listens for client requests, retrieves map data, rover commands, and mine serial numbers. It shares mine PINs with the client upon request and notifies the deminer about the mines to disarm.

Deminer: Listens for tasks from the server, disarms mines, and publishes the mine PINs to the server.

Main Data Structures and Functions:

- **Rover.proto:** Defines the gRPC service and message types used for communication between client and server.

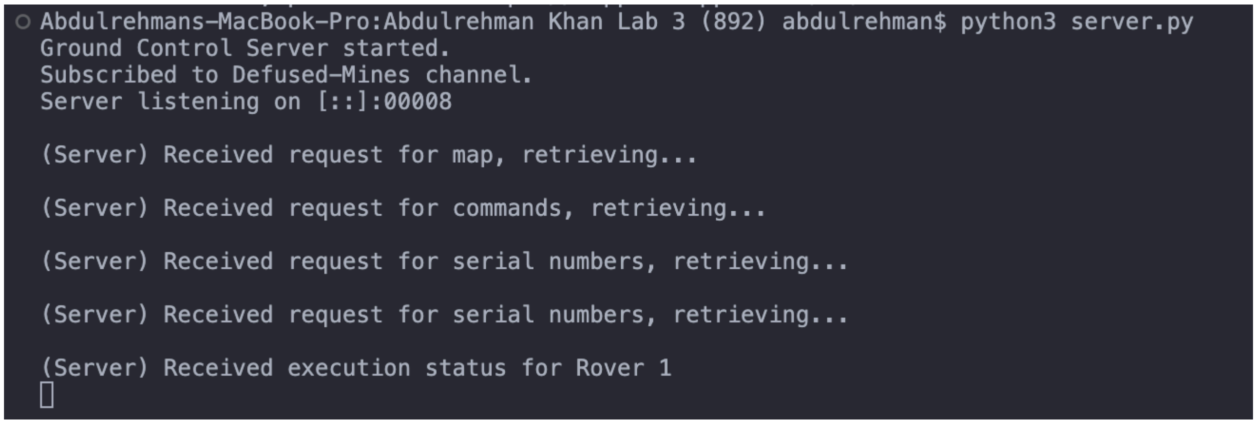
- **Client.py:** Contains functions to handle client-side operations such as requesting map data, commands, and sharing mine PINs with the server.

- **Server.py:** Implements the server-side logic for handling client requests, retrieving map data, commands, and mine information. It also communicates with the deminer to notify about the mines.

- **Deminer.py:** Listens for tasks from the server, disarms mines, and publishes the mine PINs to the server.

Test Run Screenshots:

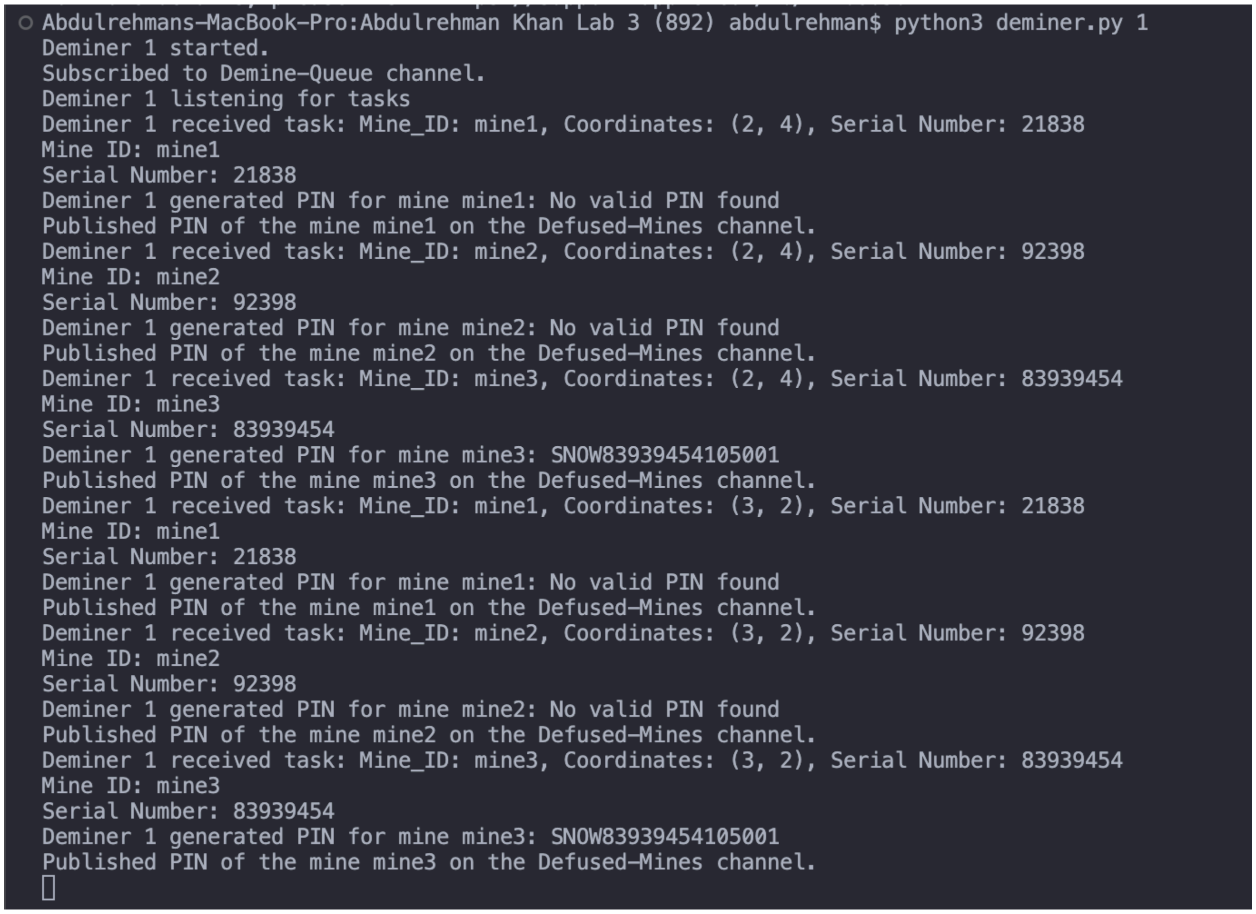
Server.py:



```
○ Abdulrehmans-MacBook-Pro:Abdulrehman Khan Lab 3 (892) abdulrehman$ python3 server.py
  Ground Control Server started.
  Subscribed to Defused-Mines channel.
  Server listening on [::]:00008

  (Server) Received request for map, retrieving...

  (Server) Received request for commands, retrieving...

  (Server) Received request for serial numbers, retrieving...

  (Server) Received request for serial numbers, retrieving...

  (Server) Received execution status for Rover 1
```

1. The server is started, indicating that the Ground Control Server is up and running.
2. It subscribes to the Defused-Mines channel to receive notifications about disarmed mines.
3. The server listens on a specified address and port for incoming connections.
4. Various log messages show that the server received requests for map data, commands, and serial numbers from the client. It processes these requests accordingly.
5. Finally, it receives the execution status for Rover 1, indicating that the rover has completed its task.

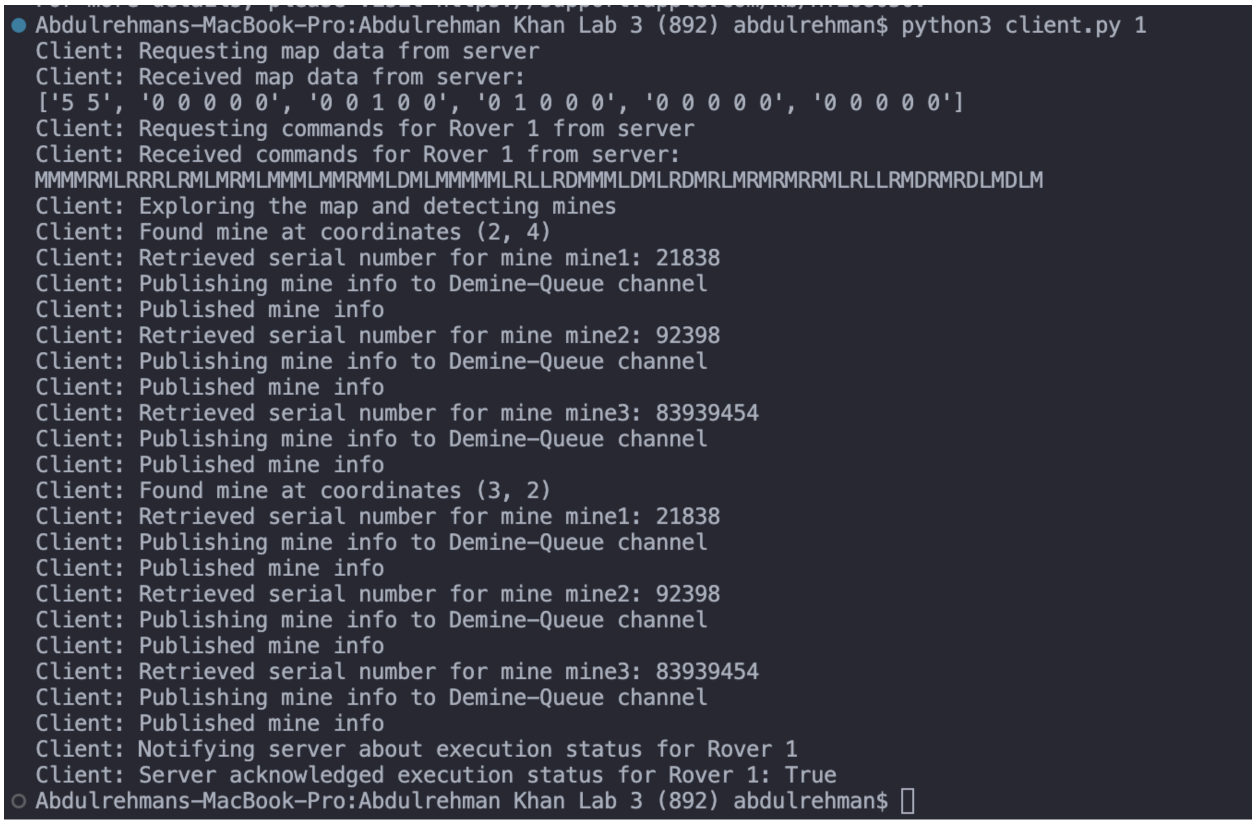
Deminer.py:



```
○ Abdulrehmans-MacBook-Pro:Abdulrehman Khan Lab 3 (892) abdulrehman$ python3 deminer.py 1
  Deminer 1 started.
  Subscribed to Demine-Queue channel.
  Deminer 1 listening for tasks
  Deminer 1 received task: Mine_ID: mine1, Coordinates: (2, 4), Serial Number: 21838
  Mine ID: mine1
  Serial Number: 21838
  Deminer 1 generated PIN for mine mine1: No valid PIN found
  Published PIN of the mine mine1 on the Defused-Mines channel.
  Deminer 1 received task: Mine_ID: mine2, Coordinates: (2, 4), Serial Number: 92398
  Mine ID: mine2
  Serial Number: 92398
  Deminer 1 generated PIN for mine mine2: No valid PIN found
  Published PIN of the mine mine2 on the Defused-Mines channel.
  Deminer 1 received task: Mine_ID: mine3, Coordinates: (2, 4), Serial Number: 83939454
  Mine ID: mine3
  Serial Number: 83939454
  Deminer 1 generated PIN for mine mine3: SNOW83939454105001
  Published PIN of the mine mine3 on the Defused-Mines channel.
  Deminer 1 received task: Mine_ID: mine1, Coordinates: (3, 2), Serial Number: 21838
  Mine ID: mine1
  Serial Number: 21838
  Deminer 1 generated PIN for mine mine1: No valid PIN found
  Published PIN of the mine mine1 on the Defused-Mines channel.
  Deminer 1 received task: Mine_ID: mine2, Coordinates: (3, 2), Serial Number: 92398
  Mine ID: mine2
  Serial Number: 92398
  Deminer 1 generated PIN for mine mine2: No valid PIN found
  Published PIN of the mine mine2 on the Defused-Mines channel.
  Deminer 1 received task: Mine_ID: mine3, Coordinates: (3, 2), Serial Number: 83939454
  Mine ID: mine3
  Serial Number: 83939454
  Deminer 1 generated PIN for mine mine3: SNOW83939454105001
  Published PIN of the mine mine3 on the Defused-Mines channel.
```

1. The deminer component is started, indicating that it's ready to receive tasks.
2. It subscribes to the Demine-Queue channel to receive tasks from the server.
3. The deminer listens for tasks and receives information about mines to disarm.
4. For each mine task received, it disarms the mine, generates a PIN (if applicable), and publishes the PIN to the Defused-Mines channel.
5. The deminer handles each task sequentially and provides feedback about the mine disarming process.

Client.py:



```
Abdulrehmans-MacBook-Pro:Abdulrehman Khan Lab 3 (892) abdulrehman$ python3 client.py 1
  Client: Requesting map data from server
  Client: Received map data from server:
  ['5 5', '0 0 0 0 0', '0 0 1 0 0', '0 1 0 0 0', '0 0 0 0 0', '0 0 0 0 0']
  Client: Requesting commands for Rover 1 from server
  Client: Received commands for Rover 1 from server:
  MMMMRMLRRRLRMLMRMLMMMLMMRMMLDMLMMMMMLRLLRDMMMLDMLRDMRLMRMRMRRMLRLLRMDRMRDLMDLM
  Client: Exploring the map and detecting mines
  Client: Found mine at coordinates (2, 4)
  Client: Retrieved serial number for mine mine1: 21838
  Client: Publishing mine info to Demine-Queue channel
  Client: Published mine info
  Client: Retrieved serial number for mine mine2: 92398
  Client: Publishing mine info to Demine-Queue channel
  Client: Published mine info
  Client: Retrieved serial number for mine mine3: 83939454
  Client: Publishing mine info to Demine-Queue channel
  Client: Published mine info
  Client: Found mine at coordinates (3, 2)
  Client: Retrieved serial number for mine mine1: 21838
  Client: Publishing mine info to Demine-Queue channel
  Client: Published mine info
  Client: Retrieved serial number for mine mine2: 92398
  Client: Publishing mine info to Demine-Queue channel
  Client: Published mine info
  Client: Retrieved serial number for mine mine3: 83939454
  Client: Publishing mine info to Demine-Queue channel
  Client: Published mine info
  Client: Notifying server about execution status for Rover 1
  Client: Server acknowledged execution status for Rover 1: True
Abdulrehmans-MacBook-Pro:Abdulrehman Khan Lab 3 (892) abdulrehman$ 
```

1. The client requests map data from the server and receives the map information, indicating the size of the map and the locations of obstacles or mines.
2. It requests rover commands for Rover 1 and receives a sequence of commands to explore the map.
3. The client explores the map, detects mines at specific coordinates, and retrieves the serial numbers for each mine detected.
4. For each detected mine, it publishes the mine information (ID, coordinates, serial number) to the Demine-Queue channel.
5. The client notifies the server about the completion of execution for Rover 1 and receives acknowledgment from the server.

Conclusion

The rover control system successfully demonstrates the communication between client, server, and deminer components using gRPC. The system allows clients to request map data, rover commands, and mine information, while the server processes these requests and coordinates with the deminer to disarm mines. The implementation showcases the use of gRPC for efficient and scalable communication in distributed systems.