Parthiv A Dholaria , Arnav Agarwal , Arjit Singh Arora

DSCD ASSIGNMENT - 1

Q1 Online Shopping Platform : gRPC

Python Files submitted:

Seller.py

Seller2.py

Buyer.py

Buyer2.py

Market.py

Along with generate.proto

Run the proto file using the following command

python3 -m grpc_tools.protoc -I protos --python_out=. --grpc_python_out=. protos/generate.proto

Note :

To run the files on the Google cloud each seller and buyer needs to manually hard code the value of the external IP and port of the market [as shown]. Also we have taken the IP and port for the notifications server's of client and buyer as input.

```
# with grpc.insecure_channel('34.131.59.13:50051') as
channel: # Server IP
```

Q2) Low Level Group Messaging Application using ZeroMQ

Run the files in the following order:

1) message_server.py: python3 message_server.py(on google cloud)
2) group.py: python3 group.py (on google cloud)
3) user_1.py: python user_1.py
4) user_2.py: python user_2.py

The group.py and the user_1.py(and user_2.py) both need the IP address of the server. Obtain that from the Google Cloud(external IP)

Assumptions:

1) user_1.py(user_2.py) needs to leave all groups before terminating itself, since we would want to deallocate all resources(for this user) on the group_server side.
2) Multiple users can connect to the same group at a point in time. Also a single user can make themselves a part of multiple groups.

Q3) YouTube-like Application with RabbitMQ

Introduction

This is a simplified version of a YouTube application built using RabbitMQ . It consists of three components: YouTuber, User, and YouTubeServer. YouTubers can publish videos, users can subscribe to YouTubers and receive notifications when they upload new videos, and the YouTubeServer manages communication between users and YouTubers.

Components

youtuber.py: Represents the service for YouTubers to publish videos.

user.py: Represents the service for users to subscribe/unsubscribe to YouTubers and receive notifications. Multiple files can be run simultaneously simulating different users.

youtube.py: The server which , manages communication between users and YouTubers, consuming messages from both parties and handling subscriptions, unsubscriptions, and notifications.

Important Notes

- Update the market IP address in User.py and Youtuber.py to the server hosted on Google Cloud.
- Persistence is achieved using durable queues, exchanges, and delivery_mode=2. This ensures that messages remain in the queue even if the user or server is restarted.
- The YouTube server simultaneously consumes messages from users and YouTubers.

Messages are exchanged between components using RabbitMQ message queues, following the Advanced Message Queuing Protocol (AMQP).