Group RPC Communication Report

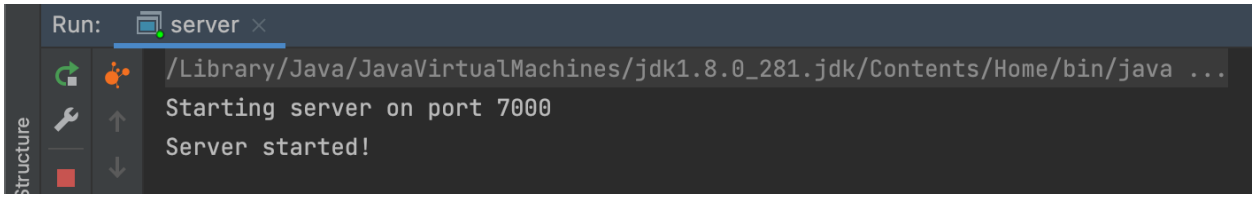
1. Introduction

This project is a server-client “group chat”, which contains one main server and multiple clients communicating through rpc. Clients in the same group can send/receive the message from each other, they won’t send/receive any message to/from the other group. In this project, I choose Java as the programming language and grpc framework, which is fast and flexible with open source, to implement rpc communication.

2. Project

1) Entity.
    a) User. Store name and groupId;
    b) ChatMsg. Store sender(User) and msg(message content);
    c) ChatMsgFromServer. Store timestamp and ChatMsg;
    d) UnreadMsg. Multiple ChatMsgFromServer.
2) Service.
    a) createUser (User) return User.
    Initialize the user and register on the server. If the client doesn’t offer a group if, then assign a timestamp as group id, which won’t be the same with other groups.
    b) sendMsg (ChatMsg) return ChatMsgFromServer.
    Send a message to the server and assign a timestamp. Then broadcast the message to all clients in the group. The server will store messages in the ConccurentHashMap<long, ConcurrentLinkedDeque<ChatMsgFromServer>> senders in the ConcurrentHashMap<long, Set<StreamObserver<ChatMsgFromClient>>> according to group id.
    c) getUnreadMsg (User) return UnreadMsg.
    Get all messages whose timestamp is greater than the user's last login timestamp, which are the unread messages, from the group.
3) Workflow.
    - Start server assigning a port 7000 (or default 8000)

```
java -jar Msg-Server-1.0-SNAPSHOT-jar-with-dependencies.jar -client Alice,Bob,Chad,Doug -port 7000
```

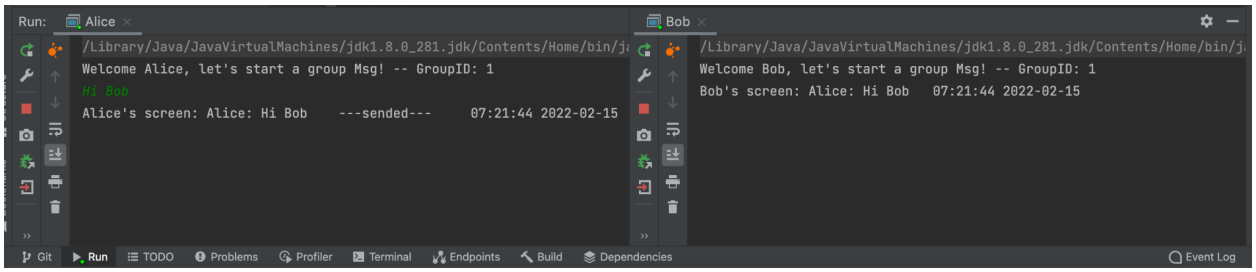


- Start client assigning client name (id), group id (default timestamp),

```
java -jar Msg-Client-1.0-SNAPSHOT-jar-with-dependencies.jar -client_id Alice -group_id 1 -port 7000
```

- Send the message to the group Bob can receive the message from Alice
- Before that anyone who is new to the group will send an initialized message before start chat, which lets the server know that the new members joined in and put them in the broadcast list. And then get all unread messages in the group from the server. If the user is not in any existing group, the server will not send any unread messages.

https://github.com/grpc/grpc-web



4) Deploy.

```
docker build -t rpc .
docker run -d -p 8080:8080 --name rpc rpc
```

Output is in the log.txt and log image

3. Improvement

1) More functional. Users can change their group and be allowed to join multiple groups.
2) More Available. Clients should be able to store some messages in cache. Once message size increases, it helps release pressure for the server.

4. Reference

Quick start | Java | gRPC
github/grpc/grpc-web
gRPC 101 for Java Developers - WEB2DAY 2016