Hand-in3

Group HelloWorld: Efe Ertug Erdem, Yu Pei, Yu Sun

Discuss, whether you are going to use server-side streaming, client-side streaming, or bidirectional streaming?

- Use server-side streaming to receive messages from the server. because the client need to keep listening to the server, and the server need to broadcast all the the messages to all the clients.
- Use simple RPC to send message from client to server. So only when a client need to send a message, this function will be called. This will save some resources.

Describe your system architecture - do you have a server-client architecture, peer-to-peer, or something else?

Use server-client architecture. The client send messages to the server, and the server broadcast the messages to all the clients.

Describe what RPC methods are implemented, of what type, and what messages types are used for communication

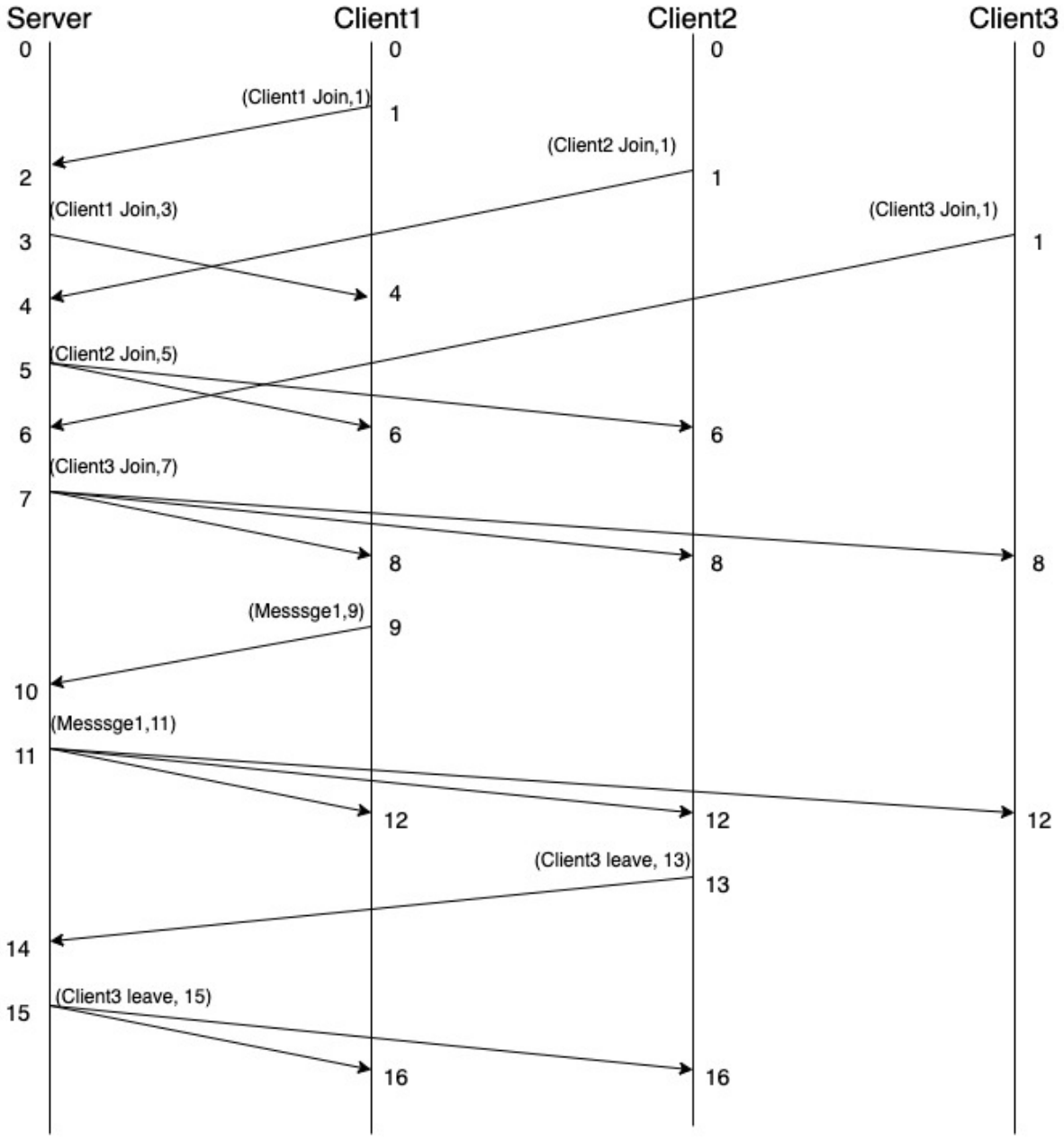
- message ChatMessage: The message type used for communication between server and client. It contains the following fields:
  - senderId: the id of the sender
  - timestamp: the timestamp of the message
  - messageText: the content of the message
- message ServerResponse: The message type used for giving response to the client. It contains the following fields:
  - timestamp: the timestamp of the message
  - result: true if the operation is successful, false otherwise
- message ClientRequest: The message type used for sending request(join or leave) to the server. It contains the following fields:
  - clientId: the id of the sender
  - timestamp: the timestamp of the message
- rpc ClientSend(ChatMessage) returns (ServerResponse): Simple RPC
  - ClientSend is used for publishing messages to the server. It returns a ServerResponse to indicate whether the operation is successful.

- rpc ClientReceive(ClientRequest) returns (stream ChatMessage): Server-side streaming RPC
    - ClientReceive is used for receiving messages from the server. It returns a stream of ChatMessage. Server-side streaming RPC
- rpc Join(ClientRequest) returns (ServerResponse): Simple RPC
    - Join is used for sending join request to the server.
- rpc Leave(ClientRequest) returns (ServerResponse): Simple RPC
    - Leave is used for sending leave request to the server.

Describe how you have implemented the calculation of the Lamport timestamps

- At the beginning the clock of server and all the client are set to 0.
- When a client join the server/leave the server, send a message, it will increase the clock by 1. Then attach the current client clock to the TimeStamp.
- When a client receive a message, it will compare the clock of the message and its own clock, and set the clock to the max of the two clocks. Then increase the clock by 1.
- When a server broadcast a message to the clients, it will increase the clock by 1. Then attach the current server clock to the ChatMessage.TimeStamp.
- When a server receive a message, it will compare the clock of the message and its own clock, and set the clock to the max of the two clocks. Then increase the clock by 1.

Provide a diagram, that traces a sequence of RPC calls together with the Lamport timestamps, that corresponds to a chosen sequence of interactions: Client X joins, Client X Publishes, ..., Client X leaves. Include documentation (system logs) in your appendix.



- see Appendix

Provide a link to a Git repo with your source code in the report

https://github.itu.dk/HelloWorld/Hand-in3

Include system logs, that document the requirements are met, in the appendix of your report

- see Appendix

Include a readme.md file that describes how to run your program.

- Start server: go run server/server.go

- Start client x and join the chat: go run client/client.go -cid x (replace x with an integer)
- in the client's terminial, type any message to publish
- type "leave" to leave the chat

Appendix

Log:

```
2023/10/30 11:15:38 Started server at port: 5454
2023/10/30 11:15:43 Client1 clock is updated to 1
2023/10/30 11:15:43 Server clock is updated to 2
2023/10/30 11:15:43 Server clock is updated to 3
2023/10/30 11:15:43 Client 1 join chat //a client node can join the system
2023/10/30 11:15:43 Client1 clock is updated to 4
2023/10/30 11:15:43 Participant 1 joined Chitty-Chat at Lamport time 3, current clock is 4
2023/10/30 11:15:52 Client2 clock is updated to 1
2023/10/30 11:15:52 Server clock is updated to 4
2023/10/30 11:15:52 Server clock is updated to 5
2023/10/30 11:15:52 Client1 clock is updated to 6
2023/10/30 11:15:52 Participant 2 joined Chitty-Chat at Lamport time 5, current clock is 6
2023/10/30 11:15:52 Client 2 join chat
2023/10/30 11:15:52 Client2 clock is updated to 6
2023/10/30 11:15:52 Participant 2 joined Chitty-Chat at Lamport time 5, current clock is 6
2023/10/30 11:15:57 Client3 clock is updated to 1
2023/10/30 11:15:57 Server clock is updated to 6
2023/10/30 11:15:57 Server clock is updated to 7
2023/10/30 11:15:57 Client1 clock is updated to 8
2023/10/30 11:15:57 Participant 3 joined Chitty-Chat at Lamport time 7, current clock is 8
2023/10/30 11:15:57 Client2 clock is updated to 8
2023/10/30 11:15:57 Participant 3 joined Chitty-Chat at Lamport time 7, current clock is 8
2023/10/30 11:15:57 Client 3 join chat
2023/10/30 11:15:57 Client3 clock is updated to 8
2023/10/30 11:15:57 Participant 3 joined Chitty-Chat at Lamport time 7, current clock is 8
2023/10/30 11:23:53 Client1 clock is updated to 9
2023/10/30 11:23:53 Server received a message from Client 1   // a client node can send a message
2023/10/30 11:23:53 Server clock is updated to 10
2023/10/30 11:23:53 Server clock is updated to 11
2023/10/30 11:23:53 Client2 clock is updated to 12
2023/10/30 11:23:53 From 1 at clock 9: message1, current clock is 12
2023/10/30 11:23:53 Client3 clock is updated to 12
2023/10/30 11:23:53 From 1 at clock 9: message1, current clock is 12
2023/10/30 11:23:53 Client1 clock is updated to 12
2023/10/30 11:23:53 From 1 at clock 9: message1, current clock is 12
2023/10/30 11:31:02 Client3 clock is updated to 13
2023/10/30 11:31:02 Server clock is updated to 14
2023/10/30 11:31:02 Server clock is updated to 15
2023/10/30 11:31:02 Client2 clock is updated to 16
2023/10/30 11:31:02 Participant 3 left Chitty-Chat at Lamport time 15, current clock is 16
2023/10/30 11:31:02 Client1 clock is updated to 16
2023/10/30 11:31:02 Participant 3 left Chitty-Chat at Lamport time 15, current clock is 16
2023/10/30 11:31:02 Client 3 left chat // a client node can leave the system
```