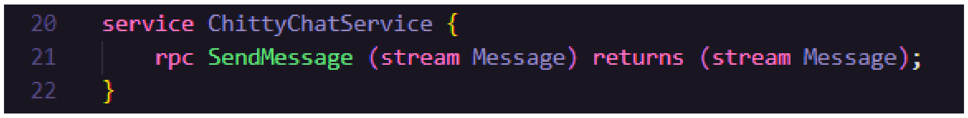
Mandatory Hand-in 3- Chitty Chat

Streaming

Our system is using bidirectional streaming which can be seen in Illustration 1.1. The illustration shows that when a client uses the SendMessage function of the service, 2 streams are made and they operate independently, so clients and servers can read and write in whatever order they like.

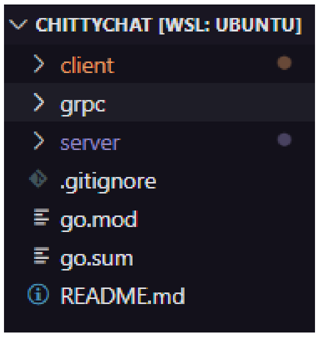


```
service ChittyChatService {
    rpc SendMessage (stream Message) returns (stream Message);
}
```

Illustration 1.1: Illustration of function that the service provides.

System Architecture

Our architecture is based on a client-server one but with adjustments to allow the server to propagate messages to other clients, making it more like a publish-subscribe architecture. In fact, when a client connects, the server creates a record in a map where it associates the key (which is the client's Ip address and port) with the related communication stream so it can retrieve it when it needs to forward messages.

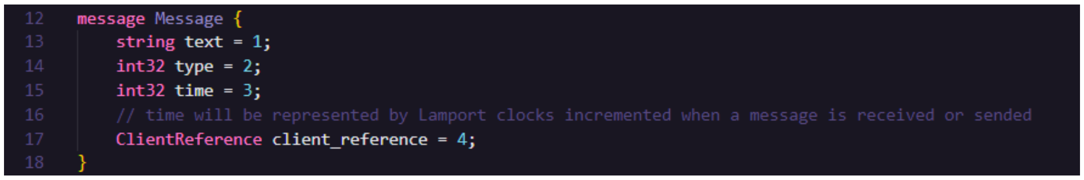


```
CHITTYCHAT [WSL: UBUNTU]
  > client
  > grpc
  > server
    .gitignore
    go.mod
    go.sum
    README.md
```

Illustration 1.2: illustration of the systems structure.

RPC implementation

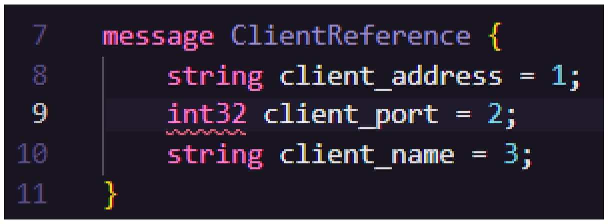
As seen in Illustration 1.1, the service contains just 1 RPC method that opens a stream from the client to the server and vice versa. This method uses a message with a struct seen in Illustration 1.3. This struct contains a string which is the message the client wants to send, an int32 that represents whether a client is connecting, disconnecting or publishing a message. Another int32 is used for the time.



```
message Message {
    string text = 1;
    int32 type = 2;
    int32 time = 3;
    // time will be represented by Lamport clocks incremented when a message is received or sended
    ClientReference client_reference = 4;
}
```

Illustration 1.3: Illustration of the Message struct

The ClientReference is a struct that contains all the necessary information about the connecting clients, see Illustration 1.4. The ClientReference has a string containing the IP-address of the client. An int32 is used to contain the port where the client is established, and a string is used for saving the clients name for better user friendliness.



```
message ClientReference {
    string client_address = 1;
    int32 client_port = 2;
    string client_name = 3;
}
```

Illustration 1.4: illustration of ClientReference.

Calculation of Lamport Timestamps

Every node has a Lamport timestamp. That means both the clients and the server have a Lamport timestamp. Every time an event has occurred on the client, its local timestamp will increment. Before the client sends a message to the server the timestamp will increment, and the incremented timestamp will be sent with the message. The server receives the message and compares the message's timestamp with its local timestamp. The higher timestamp gets saved and incremented. The server then sends the message to the other clients. For each client before sending the message the timestamp will be incremented and sent with message. On the client side when the message is received the message’s timestamp will be compared to the local timestamp and the higher timestamp will be saved and incremented.

Lamport Timestamp Timeline Diagram

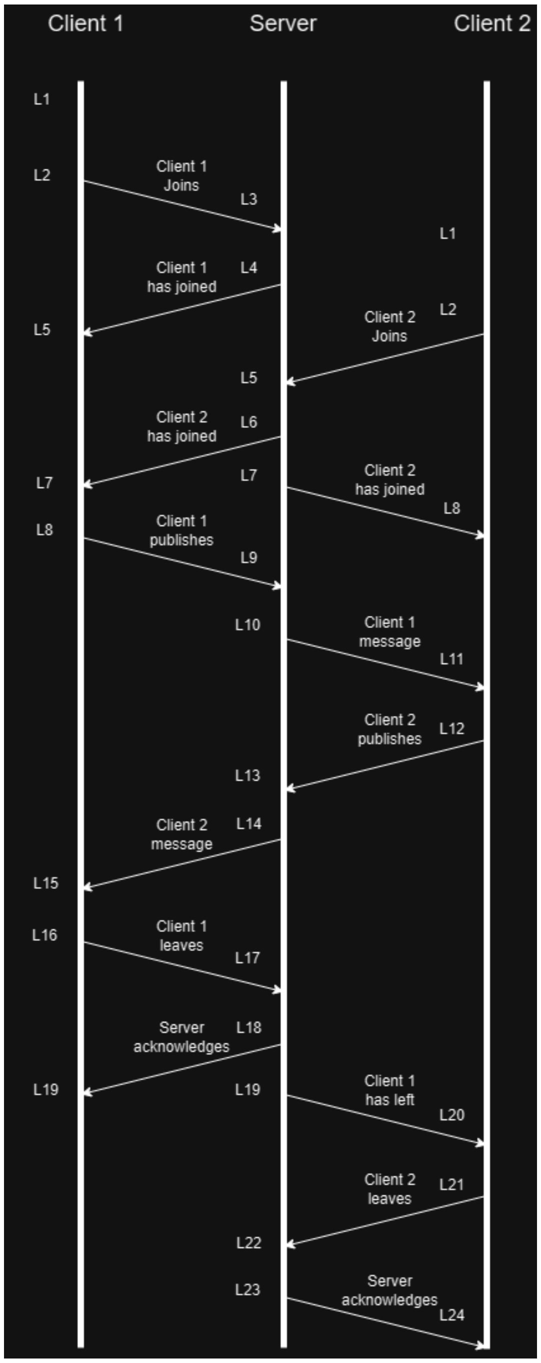


Illustration 1.5: gRPC call diagram

Documentation of the interactions visualized in Illustration 1.5 can be seen in the system logs at appendix 1.

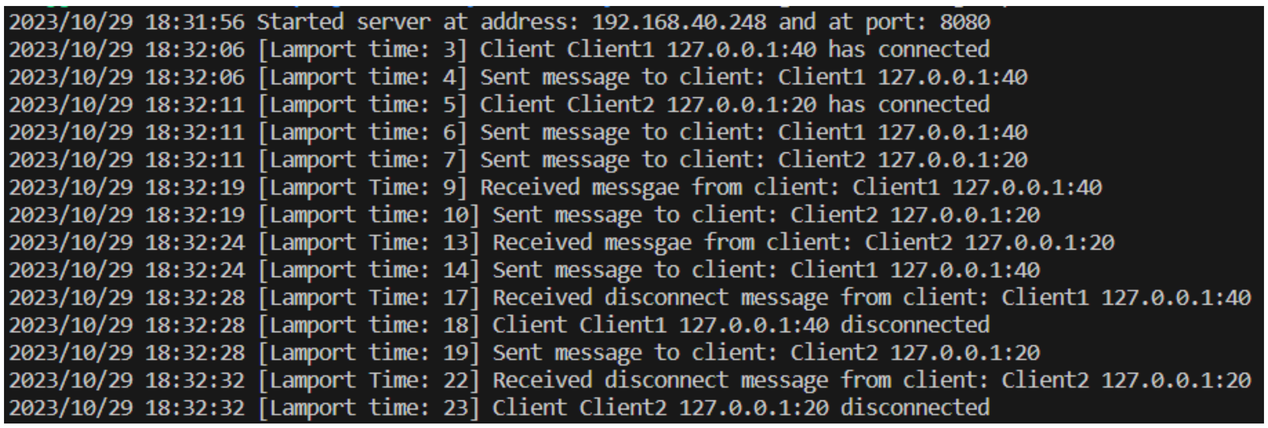
GitHub Repository Link

https://github.com/rafj58/ChittyChat

Appendix

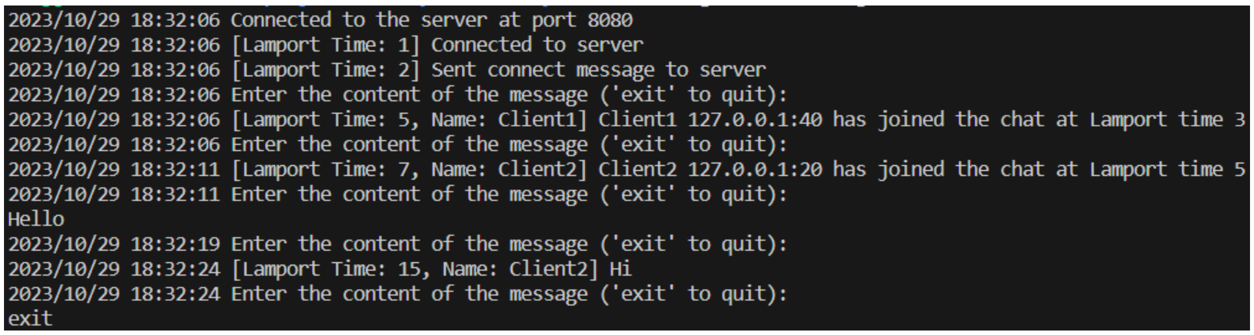
1. System logs for gRPC call diagram

Server:



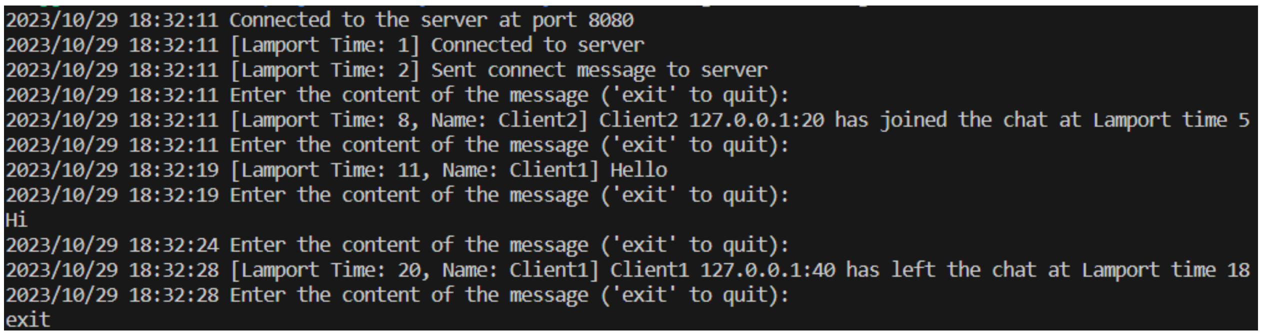
```
2023/10/29 18:31:56 Started server at address: 192.168.40.248 and at port: 8080
2023/10/29 18:32:06 [Lamport time: 3] Client Client1 127.0.0.1:40 has connected
2023/10/29 18:32:06 [Lamport time: 4] Sent message to client: Client1 127.0.0.1:40
2023/10/29 18:32:11 [Lamport time: 5] Client Client2 127.0.0.1:20 has connected
2023/10/29 18:32:11 [Lamport time: 6] Sent message to client: Client1 127.0.0.1:40
2023/10/29 18:32:11 [Lamport time: 7] Sent message to client: Client2 127.0.0.1:20
2023/10/29 18:32:19 [Lamport Time: 9] Received messgae from client: Client1 127.0.0.1:40
2023/10/29 18:32:19 [Lamport time: 10] Sent message to client: Client2 127.0.0.1:20
2023/10/29 18:32:24 [Lamport Time: 13] Received messgae from client: Client2 127.0.0.1:20
2023/10/29 18:32:24 [Lamport time: 14] Sent message to client: Client1 127.0.0.1:40
2023/10/29 18:32:28 [Lamport Time: 17] Received disconnect message from client: Client1 127.0.0.1:40
2023/10/29 18:32:28 [Lamport time: 18] Client Client1 127.0.0.1:40 disconnected
2023/10/29 18:32:28 [Lamport time: 19] Sent message to client: Client2 127.0.0.1:20
2023/10/29 18:32:32 [Lamport Time: 22] Received disconnect message from client: Client2 127.0.0.1:20
2023/10/29 18:32:32 [Lamport time: 23] Client Client2 127.0.0.1:20 disconnected
```

Client 1:



```
2023/10/29 18:32:06 Connected to the server at port 8080
2023/10/29 18:32:06 [Lamport Time: 1] Connected to server
2023/10/29 18:32:06 [Lamport Time: 2] Sent connect message to server
2023/10/29 18:32:06 Enter the content of the message ('exit' to quit):
2023/10/29 18:32:06 [Lamport Time: 5, Name: Client1] Client1 127.0.0.1:40 has joined the chat at Lamport time 3
2023/10/29 18:32:06 Enter the content of the message ('exit' to quit):
2023/10/29 18:32:11 [Lamport Time: 7, Name: Client2] Client2 127.0.0.1:20 has joined the chat at Lamport time 5
2023/10/29 18:32:11 Enter the content of the message ('exit' to quit):
Hello
2023/10/29 18:32:19 Enter the content of the message ('exit' to quit):
2023/10/29 18:32:24 [Lamport Time: 15, Name: Client2] Hi
2023/10/29 18:32:24 Enter the content of the message ('exit' to quit):
exit
```

Client 2:



```
2023/10/29 18:32:11 Connected to the server at port 8080
2023/10/29 18:32:11 [Lamport Time: 1] Connected to server
2023/10/29 18:32:11 [Lamport Time: 2] Sent connect message to server
2023/10/29 18:32:11 Enter the content of the message ('exit' to quit):
2023/10/29 18:32:11 [Lamport Time: 8, Name: Client2] Client2 127.0.0.1:20 has joined the chat at Lamport time 5
2023/10/29 18:32:11 Enter the content of the message ('exit' to quit):
2023/10/29 18:32:19 [Lamport Time: 11, Name: Client1] Hello
2023/10/29 18:32:19 Enter the content of the message ('exit' to quit):
Hi
2023/10/29 18:32:24 Enter the content of the message ('exit' to quit):
2023/10/29 18:32:28 [Lamport Time: 20, Name: Client1] Client1 127.0.0.1:40 has left the chat at Lamport time 18
2023/10/29 18:32:28 Enter the content of the message ('exit' to quit):
exit
```