Verteilte Systeme im Sommersemester 2022

Patrick Felschen, Matr. Nr. 932056
Julian Voß, Matr. Nr. 934505

Osnabrück, 25.04.2022

Aufgabenblatt 4

Aufgabe 1 – Implementierung

Senden von Client an Server

```
/* Platzhalter erstellen ... */

ClientContext context;
SubscriberAddress subscriber;
ReturnCode reply;
// TODO: Receiver Adresse setzen ...
subscriber.set_ip_address(get_receiver_ip());
subscriber.set_port(PUBSUB_RECEIVER_PORT);
// TODO: RPC abschicken ...
stub_->subscribe(&context, subscriber, &reply);
// TODO: Status / Reply behandeln ...
Status status;
this->handle_status("subscribe()", status, reply);
```

Zuerst werden Platzhalter für ClientContext (enthält optionale Metadaten), SubscriberAddress (enthält hier die IP-Adresse und den Port des Nutzers) und den ReturnCode angelegt. Die Platzhalter werden danach mit Inhalt gefüllt.

Über einen PubSubService können Funktionen für die Übertragung an den Server aufgerufen werden. Dieser verarbeitet die Daten und setzt den ReturnCode auf einen entsprechenden Wert. Dieser wird in der handle_status Methode verarbeitet und auf der Konsole ausgegeben. Die set_topic und unsubscribe Funktionen funktionieren ähnlich.

Verarbeiten der Daten im Server

Der Server empfängt die Daten vom Client und wandelt die empfangene SubscriberAddress in einen String um. Dieser String repräsentiert einen Nutzer. Die Nutzer werden in einem Set verwaltet. Ist der Client bereits registriert, erhält der ReturnCode eine Fehlermeldung, ansonsten wird OK zurückgegeben und der Nutzer im Set eingefügt. Auch hier funktionieren die set_topic und unsubscribe Funktionen ähnlich.

```
Status subscribe(ServerContext *context, const SubscriberAddress *request,
                 ReturnCode *reply) override
{
  std::string receiver = stringify(*request);
  // TODO: Client registrieren und Info ausgeben
  if(subscribers.find(receiver) == subscribers.end()) {
    reply->set_value(ReturnCode::OK);
    subscribers.insert(receiver);
    std::cout << "Client [" << receiver << "] registriert" << std::endl;
  }
  else {
    reply->set_value(ReturnCode::CLIENT_ALREADY_REGISTERED);
    std::cout << "Client [" << receiver << "] registrieren fehlgeschlagen"
    << std::endl;
  }

  return Status::OK;
}
```

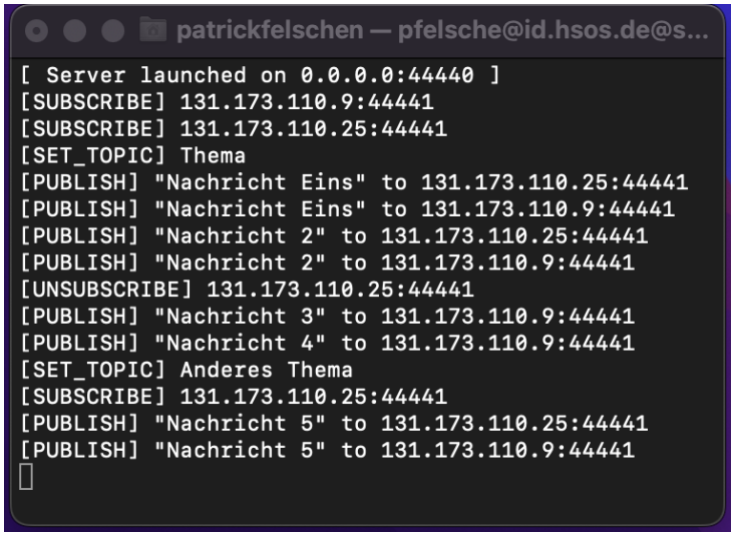
Veröffentlichen einer Nachricht

Nachdem der Server über die publish Funktion eine Nachricht vom Client erhalten hat, wird diese an alle Clients weitergeleitet, die sich im Set befinden. Für jeden Client (hier auch subscriber genannt) wird ein Channel angelegt, wodurch ein PubSubDelivService erstellt werden kann. Dadurch wird die Nachricht an den Receiver weitergeleitet. Im Server wird die Nachricht um ein Präfix erweitert, welches das Topic enthält. Der Receiver gibt die Nachricht inclusive Timestamp auf der Konsole aus.

```
Status publish(ServerContext *context, const Message *request,
ReturnCode *reply) override
{
  std::cout << request->message() << std::endl;
  // TODO: Nachricht an alle Subscriber verteilen
    for (std::string subscriber : subscribers) {
      std::shared_ptr<Channel> channel = grpc::CreateChannel(subscriber,
      grpc::InsecureChannelCredentials());
      stub_ = PubSubDelivService::NewStub(channel);

      EmptyMessage empty_message;
      ClientContext client_context;
      Message topic_message;

      topic_message.set_message("<" + server_topic + ">"
      + request->message());
      Status status = stub_->deliver(&client_context, topic_message,
      &empty_message);

      this->handle_status(subscriber, status);
    }
  reply->set_value(ReturnCode::OK);
  return Status::OK;
}
```

Aufgabe 2 – Test

Server: 131.173.110.9:44440



Zunächst „subscriben“ Client A und Client B dem Server. Danach setzt Client A ein neues Thema „Thema“.

Nachdem dieses gesetzt wurde, veröffentlicht Client B die Nachricht „Nachricht Eins“ und Client A die Nachricht „Nachricht 2“.

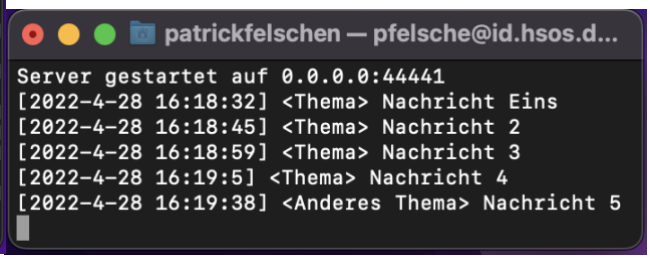
Der Client B „unsubscribt“ dem Server, woraufhin Client A „Nachricht 3“ sendet und Client B „Nachricht 4“.

Zuletzt wird das Thema auf „Anderes Thema“ gesetzt, Client B „subscribt“ dem Server erneut und Client A sendet die Nachricht „Nachricht 5“.

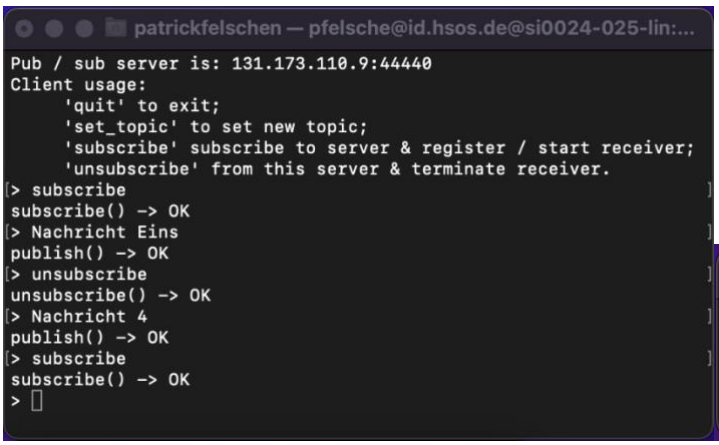
Client A: 131.173.110.9

Receiver A

Der Receiver A erhält alle gesendeten Nachrichten, da dieser durchgehend dem Server subscribt hat.

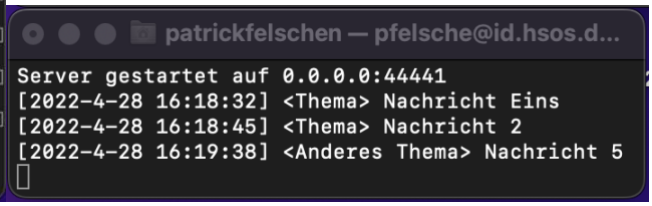


Client B: 131.173.110.25



Receiver B

Der Receiver B erhält Nachricht 3 und 4 nicht, da zu dem Zeitpunkt der Client kein Subscriber war.



Aufgabe 3 – Fragen

1.) Receiver und Client sind getrennte Prozesse, damit der Client gleichzeitig Nachrichten senden und empfangen kann.

2.) Es handelt sich um ein synchrones System (Server-Streaming: 1 Request, n Responses). Client erhalten „ohne zu warten“ direkt eine Antwort vom Server.

3.) Client IP-Adressen könnten wechseln