Client-Server-Applikation mit gRPC

In dieser Übung wird gRPC als Protokoll zwischen Client und Server studiert. Der Dienst, den der Server in dieser Übung erbringt ist sehr einfach gestrickt. Er überführt einen vorgegebenen Text in Großbuchstaben (uppercase) und Kleinbuchstaben (lowercase). Zum Aufrufen dieser Funktionen auf dem Server seitens des Client mit gRPC ist eine Protocol Buffers Datei mit geeigneten RPC- und Message-Definitionen vorgegeben. Diese Datei ist bereits compiliert worden. Der resultierende Python Code wird in Scripte für Client und Server importiert, die als Vorlage bzw. Ausgangspunkt der Übung dienen und im Gange der Handlung komplettiert werden sollen.

Die finalen Versionen der in der Übung verwendeten Scripte liegen im jeweiligen Arbeitsverzeichnis der Übung und haben im Namen den Zusatz „_final“.

Teil 1: Unary RPCs

1.) Wechseln Sie in das Verzeichnis GRPC/Übung_2/Teil_1:

```
wotan@ubuntu:~$ cd GRPC/Übung_2/Teil_1
wotan@ubuntu:~/GRPC/Übung_2/Teil_1$
```

2.) Lassen Sie sich die vorgegebene Protocol Buffers Datei transcase.proto auf der Shell ausgeben und analysieren sie ihren Inhalt:

```
wotan@ubuntu:~/GRPC/Übung_2/Teil_1$ cat transcase.proto

syntax = "proto3";

service SetCase {
 rpc UpperCase(Input) returns (Output);
  rpc LowerCase(Input) returns (Output);
}

message Input {
        string text_in = 1;
}

message Output {
        string text_out  = 1;
}
```

Woran ist erkennbar, dass es sich um Unary RPCs handelt und was charakterisiert einen Unary RPC?

3.) Öffnen Sie nun die Vorlage server_transcase.py für den Server mit der Funktion cat. Aktuell kann der Server nur Text in Großbuchstaben überführen. Dafür ist die Funktion upper() in der Klasse Trans() zuständig:

```
class Trans(transcase_pb2_grpc.SetCaseServicer):
```

```
def UpperCase(self, request, context):
    result = request.text_in.upper()
    print(result)
    return transcase_pb2.Output(text_out = result)
```

Der Name der Funktion ist identisch mit dem RPC-Namen UpperCase in der Protocoll Buffers Datei transcase.proto. Sie wird automatisch aufgerufen wenn seitens des Client der RPC UpperCase an den Server gesendet wird. Erklären Sie die Aufgabe folgender Python-Anweisungen im Zusammenhang mit gRPC:

request.text_in.upper() _____
return transcase_pb2.Output(text_out = result) _____

4.) Starten Sie den Server in der Shell:

```
wotan@ubuntu:~/GRPC/Übung_2/Teil_1$ python3 server_transcase.py
Server launched. Waiting for RPCs.
```

5.) Öffnen Sie nun eine weitere Shell, von der aus Sie den Client bedienen können und wechseln Sie in das Verzeichnis GRPC/Übung_2/Teil_1. Öffnen Sie mit cat die Vorlage für den Client client_transcase.py. Hier ist aktuell nur eine Funktion upper() definiert, die Text in Großbuchstaben überführen soll, indem sie den RPC UpperCase in Richtung Server auf die Reise schickt:

```
def upper(eingabe):

    with grpc.insecure_channel('localhost:50051') as channel:

        stub1 = transcase_pb2_grpc.SetCaseStub(channel)
        message_input = transcase_pb2.Input(text_in = eingabe)
        response = stub1.UpperCase(message_input)
        print(response.text_out)
        return response.text_out
```

Analysieren Sie die Python-Anweisungen in dieser Funktion und erklären Sie deren Funktion:

grpc.insecure_channel() _____
SetCaseStub(channel) _____
Input(text_in = eingabe) _____
stub1.UpperCase(message_input) _____

6.) Beim Starten des Client über die Kommandozeile ist der Text zu übergeben, der in Uppercase übersetzt werden soll. Starten Sie den Client und übergeben Sie einen Text beliebiger Wahl:

```
wotan@ubuntu:~/GRPC/Übung_2/Teil_1$ python3 client_transcase.py Hello World, how are
                                                                                    you
HELLO WORLD, HOW ARE YOU
```

Beobachten Sie auch die Reaktion des Servers:

```
wotan@ubuntu:~/GRPC/Übung_2/Teil_1$ python3 server_transcase.py
Server launched. Waiting for RPCs.
HELLO WORLD, HOW ARE YOU
```

7.) Stoppen Sie den Server mit Str-C. Öffnen Sie das Server Script server_transcase.py mit dem Texteditor und ergänzen sie die Funktionsdefinition für den RPC LowerCase. Orientieren Sie sich dabei an der programmatischen Struktur der bereits vorhandenen Funktion UpperCase().
Starten Sie danach den Server erneut.

8.) Öffnen Sie das Client Script client_transcase mit dem Texteditor und ergänzen Sie eine Funktion lower(), die den RPC LowerCase in Richtung Server auf den Weg schickt. Orientieren Sie sich an den Programmanweisungen der bereits vorhandenen Funktion upper(). Ergänzen Sie den Funktionsaufruf für lower auch unter if __name__ == “__main__“:

```
if __name__ == '__main__':

    input_list = sys.argv[1:]
    text_in = " ".join(input_list)

    upper(text_in)
    lower(text_in)
```

Starten Sie den Client mit einem beliebigen Text und verifizieren Sie, dass dieser nun auch in lowercase übertragen wird:

```
wotan@ubuntu:~/GRPC/Übung_2/Teil_1$ python3 client_transcase.py Hello World, whats
                                                                                  up
HELLO WORLD, WHATS UP
hello world, whats up
```

Beachten Sie auch hier die Reaktion des Servers:

```
wotan@ubuntu:~/GRPC/Übung_2/Teil_1$ python3 server_transcase_final.py
Server launched. Waiting for RPCs.
HELLO WORLD, WHATS UP
hello world, whats up
```

Teil 2: Bidirectional Streaming RPCs (Optional)

9.) Im Verzeichnis GRPC/Übung_2/Teil_2 wird derselbe Dienst über Bidirectional Streaming RPCs realisiert. Verifizieren Sie das durch Inaugenscheinnahme der Protocol Buffers Datei transcase_stream.proto.

10.) Ergänzen sie die Vorlagen für den Client client_transcase_stream.py und Server server_transcase_stream.py an den Platzhaltern <****> mit den korrekten Python-Anweisungen, damit der Code funktionstüchtig wird.

Wozu wird in dem Client Script die Funktion generate_message() benötigt?
Welche Aufgabe hat in dem Server Script die for Loop in den Funktionen UpperCase() und LowerCase()?

11.) Starten Sie den Server und den Client in unterschiedlichen Shells. Übergeben sie dem Client beim Aufruf einen beliebigen Text:

```
wotan@ubuntu:~/GRPC/Übung_2/Teil_2$ python3 client_transcase_stream.py Hello World,
                                                                        whats up
HELLO
WORLD,
WHATS
UP
hello
world,
whats
up
```

Beachten Sie auch hier die Reaktion der Server-Seite:

```
wotan@ubuntu:~/GRPC/Übung_2/Teil_2$ python3 server_transcase_stream_final.py
Server launched. Waiting for RPCs
HELLO
WORLD,
WHATS
UP
hello
world,
whats
up
```