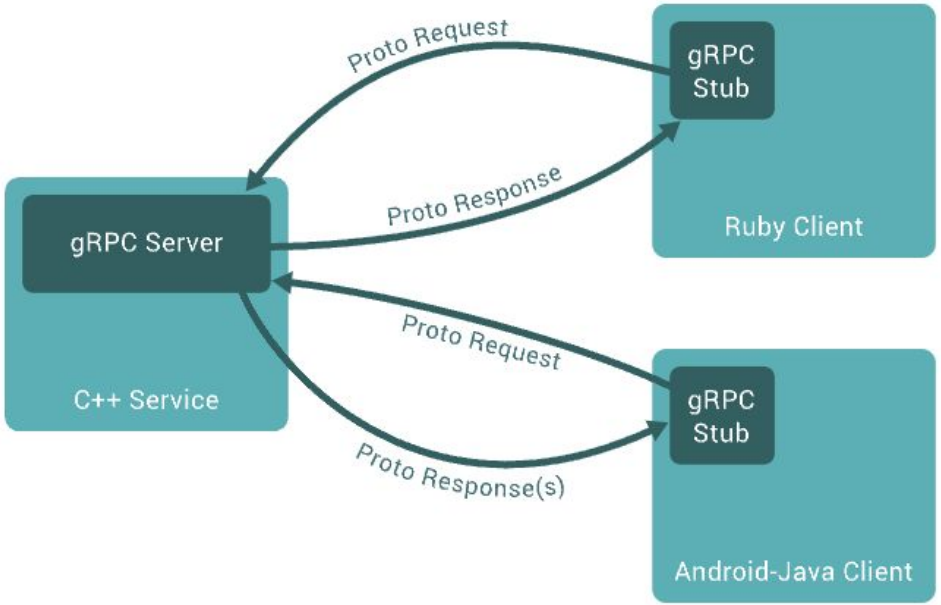
gRPC

remote procedure call framework

How it works


Proto Request
gRPC Stub
Proto Response
Ruby Client
gRPC Server
C++ Service
Proto Request
gRPC Stub
Proto Response(s)
Android-Java Client

Cross-language tool

- C# / .NET
- C++
- Dart
- Go
- Java
- Kotlin
- Node
- Objective-C
- PHP
- Python
- Ruby

Protocol buffer

.proto files

```
syntax = "proto3";

service Greeter{

 rpc SayHello(Request) returns (Response) {}

}

message Request{
 string name = 1;
}

message Response{
 string message = 1;
}
```

Example service configuration

```
def serve():
    port = '50051'
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
    helloworld_pb2_grpc.add_GreeterServicer_to_server(Greeter(), server)
    server.add_insecure_port('[::]:' + port)
    server.start()
    print("Server started, listening on " + port)
    server.wait_for_termination()
```

Example service class

```
class Greeter(helloworld_pb2_grpc.GreeterServicer):

    Yash Tibrewal
    def SayHello(self, request, context):
        return helloworld_pb2.HelloReply(message='Hello, %s!' % request.name)
```

Client call example

```
def run():
    print("Will try to greet world ...")
    with grpc.insecure_channel('localhost:50051') as channel:
        stub = helloworld_pb2_grpc.GreeterStub(channel)
        response = stub.SayHello(helloworld_pb2.HelloRequest(name='you'))
        print("Greeter client received: " + response.message)
```

Stream usage in single rpc

```
service Service {

  rpc StreamCall(stream StreamCallRequest) returns (stream StreamCallResponse);
}
```

Bidirectional streaming

Server handle example

```
def RouteChat(self, request_iterator, context):
  prev_notes = []
  for new_note in request_iterator:
    for prev_note in prev_notes:
      if prev_note.location == new_note.location:
        yield prev_note
    prev_notes.append(new_note)
```

```
def ListFeatures(self, request, context):
  left = min(request.lo.longitude, request.hi.longitude)
  right = max(request.lo.longitude, request.hi.longitude)
  top = max(request.lo.latitude, request.hi.latitude)
  bottom = min(request.lo.latitude, request.hi.latitude)
  for feature in self.db:
    if (feature.location.longitude >= left and
        feature.location.longitude <= right and
        feature.location.latitude >= bottom and
        feature.location.latitude <= top):
      yield feature
```

Client call example

```
for feature in stub.ListFeatures(rectangle):
```

```
for received_route_note in stub.RouteChat(sent_route_note_iterator):
```

Async

```python
import logging
import asyncio
from grpc import aio

import helloworld_pb2
import helloworld_pb2_grpc


class Greeter(helloworld_pb2_grpc.GreeterServicer):

    async def SayHello(self, request, context):
        return helloworld_pb2.HelloReply(message='Hello, %s!' % request.name)


async def serve():
    server = aio.server()
    helloworld_pb2_grpc.add_GreeterServicer_to_server(Greeter(), server)
    listen_addr = '[::]:50051'
    server.add_insecure_port(listen_addr)
    logging.info("Starting server on %s", listen_addr)
    await server.start()
    await server.wait_for_termination()


if __name__ == '__main__':
    logging.basicConfig(level=logging.INFO)
    asyncio.run(serve())
```

Here you can find your tasks and this presentation

git: https://github.com/AdrianKarolewski/gRPC-tutorial