gRPC

A high performance, open source universal RPC framework

Mehdi ELKHEMIRI

Timeline

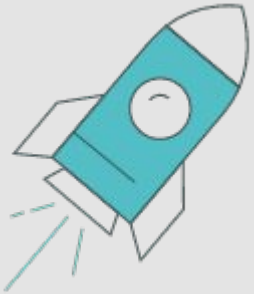
1. What is gRPC
2. PROTOBUF
3. HTTP/2
4. How it works (typical flow)
5. Four kind of service method
6. Demo: Banking Kata

What is gRPC ?

- Remote Procedure Call (RPC) framework
- High performance
- Developed by Google to connect microservices in their datacenters
- Open Source in 2016
- Part of Cloud Native Computing Foundation (CNCF) since 2017

High Performance

gRPC = PROTOCOL BUFFERS + HTTP/2



PROTOCOL BUFFERS (PROTOBUF)

- Data exchange format
- Language & Platform agnostic
- Like JSON, but smaller and faster
- Binary
- Encoding structured data
- Strongly typed
- Message schema defined in proto file

search.proto

```
syntax = 'proto3';

message SearchRequest {
  string query = 1;
  int32 page_number = 2;
  int32 result_per_page = 3;
  bool active = 4;
}
```

https://developers.google.com/protocol-buffers/docs/proto3

PROTOCOL BUFFERS (PROTOBUF)

Official Libraries and Tools

- gRPC Core - C, C++, Ruby, Node.js, Python, PHP, C#, Objective-C
- gRPC Java - The Java gRPC implementation. HTTP/2 based RPC
- gRPC Kotlin - The Kotlin gRPC implementation. Based on gRPC Java
- gRPC Node.js - gRPC for Node.js
- gRPC Go - The Go language implementation of gRPC. HTTP/2 based RPC
- gRPC Swift - The Swift language implementation of gRPC
- gRPC Dart - The Dart language implementation of gRPC
- gRPC C# - The C# language implementation of gRPC
- gRPC Web - gRPC for Web Clients
- gRPC Ecosystem - gRPC Ecosystem that complements gRPC
- gRPC contrib - Known useful contributions around github
- grpc_cli - gRPC CLI tool

https://github.com/grpc-ecosystem/awesome-grpc

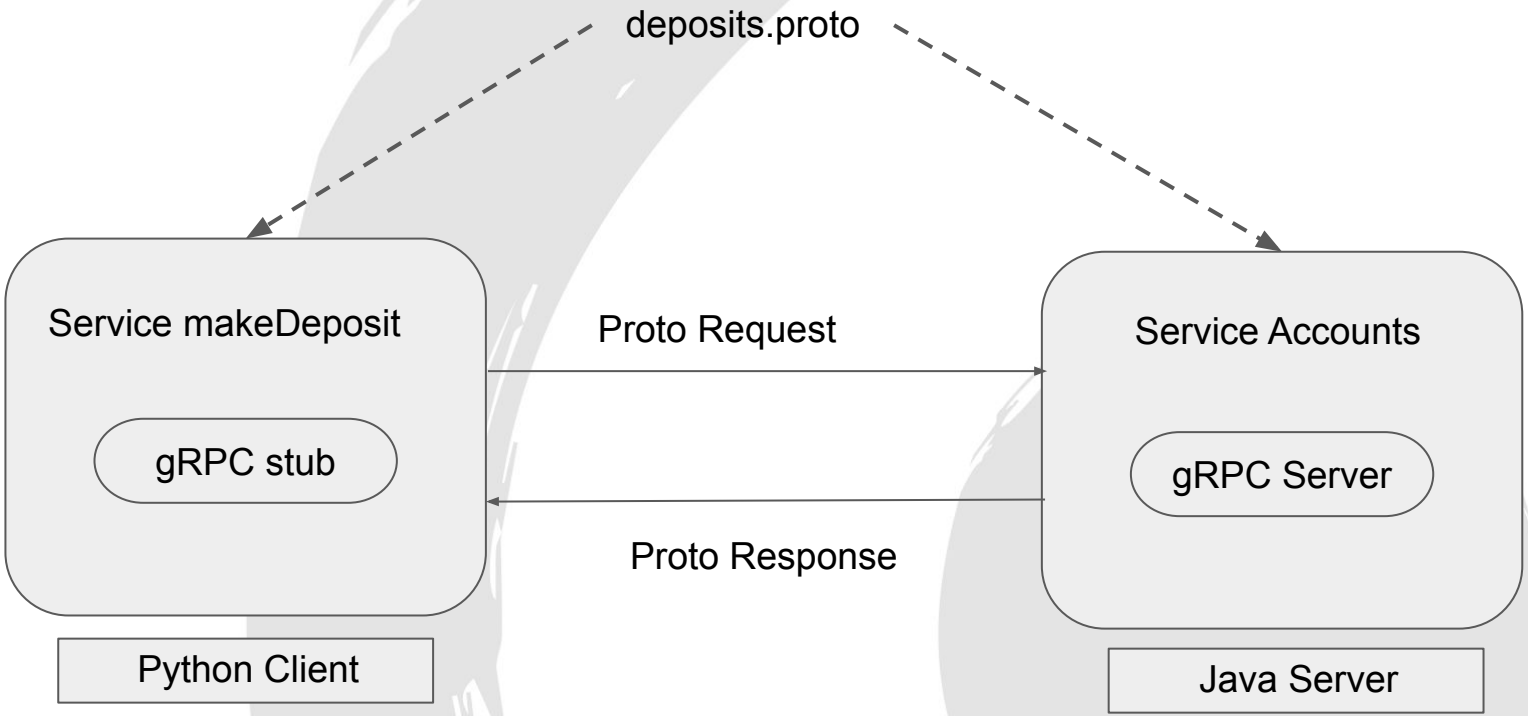
PROTOCOL BUFFERS (PROTOBUF)

- gRPC uses PROTOBUF also as Interface Description Language (IDL).
- Define services by specifying methods, parameters and return types.
- Same tools to generate classes for :
  - Client : to make RPC calls.
  - Server: to fulfill RPC requests.

deposits.proto

```
syntax = 'proto3';
service deposits {
  rpc makeDeposit(DepositRequest) returns (DepositResponse);
  rpc makeWithdraw(WithdrawRequest) returns (WithdrawResponse);
}
message DepositRequest {
  string accountId = 1;
  int32 amountInCents = 2;
}
message DepositResponse {
  string responseMessage = 1;
  int32 responseCode = 2;
}
message WithdrawRequest {
  string accountId = 1;
  int32 amountInCents = 2;
}
message WithdrawResponse {
  string responseMessage = 1;
  int32 responseCode = 2;
}
```

PROTOCOL BUFFERS (PROTOBUF)



HTTP/2

- Key differences to HTTP/1.1
  - Secure by default
  - Header compression
  - Binary instead of textual
  - Multiplexing
  - Server push

https://http2.github.io

HTTP/2 Multiplexing

HTTP/1.x has a problem called “head-of-line blocking,” where effectively only one request can be outstanding on a connection at a time.

https://http2.github.io

HTTP/2 Multiplexing

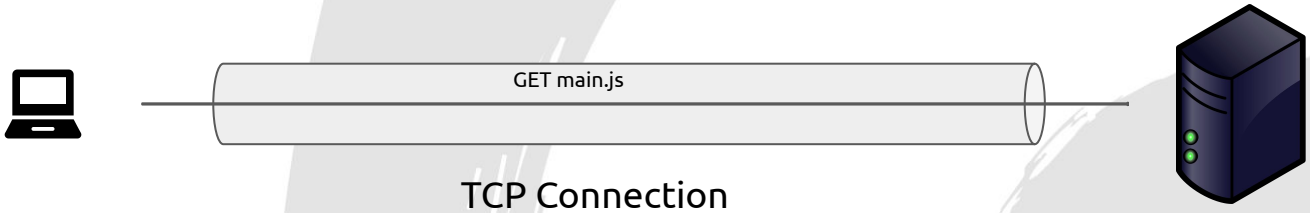
HTTP/1.1

index.html

- main.js
- main.css
- img.jpg

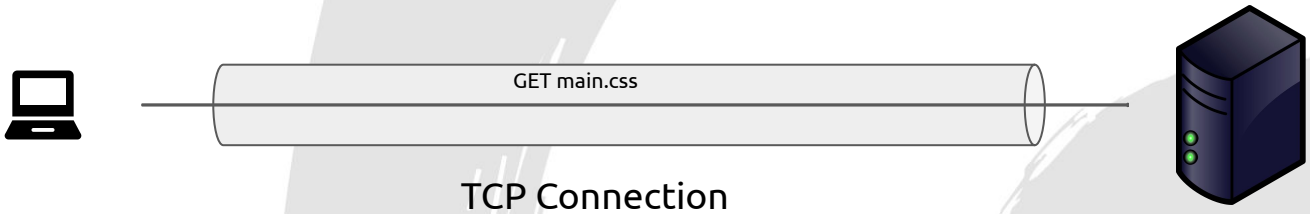
HTTP/2 Multiplexing

HTTP/1.1



HTTP/2 Multiplexing

HTTP/1.1



HTTP/2 Multiplexing

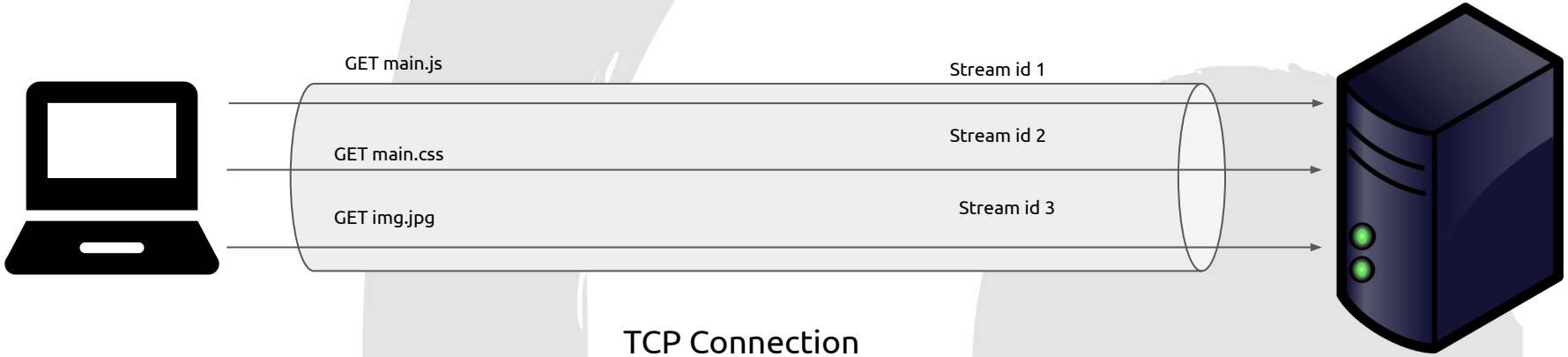
HTTP/2

index.html

- main.js
- main.css
- img.jpg

HTTP/2 Multiplexing

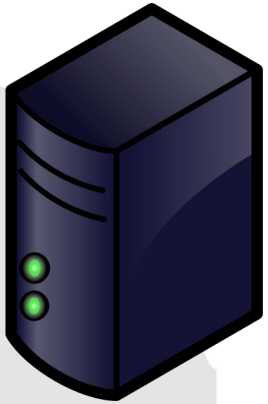
HTTP/2



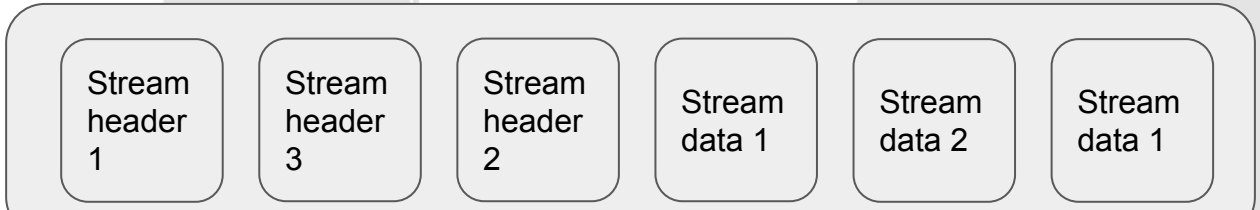
HTTP/2 Multiplexing

HTTP/2

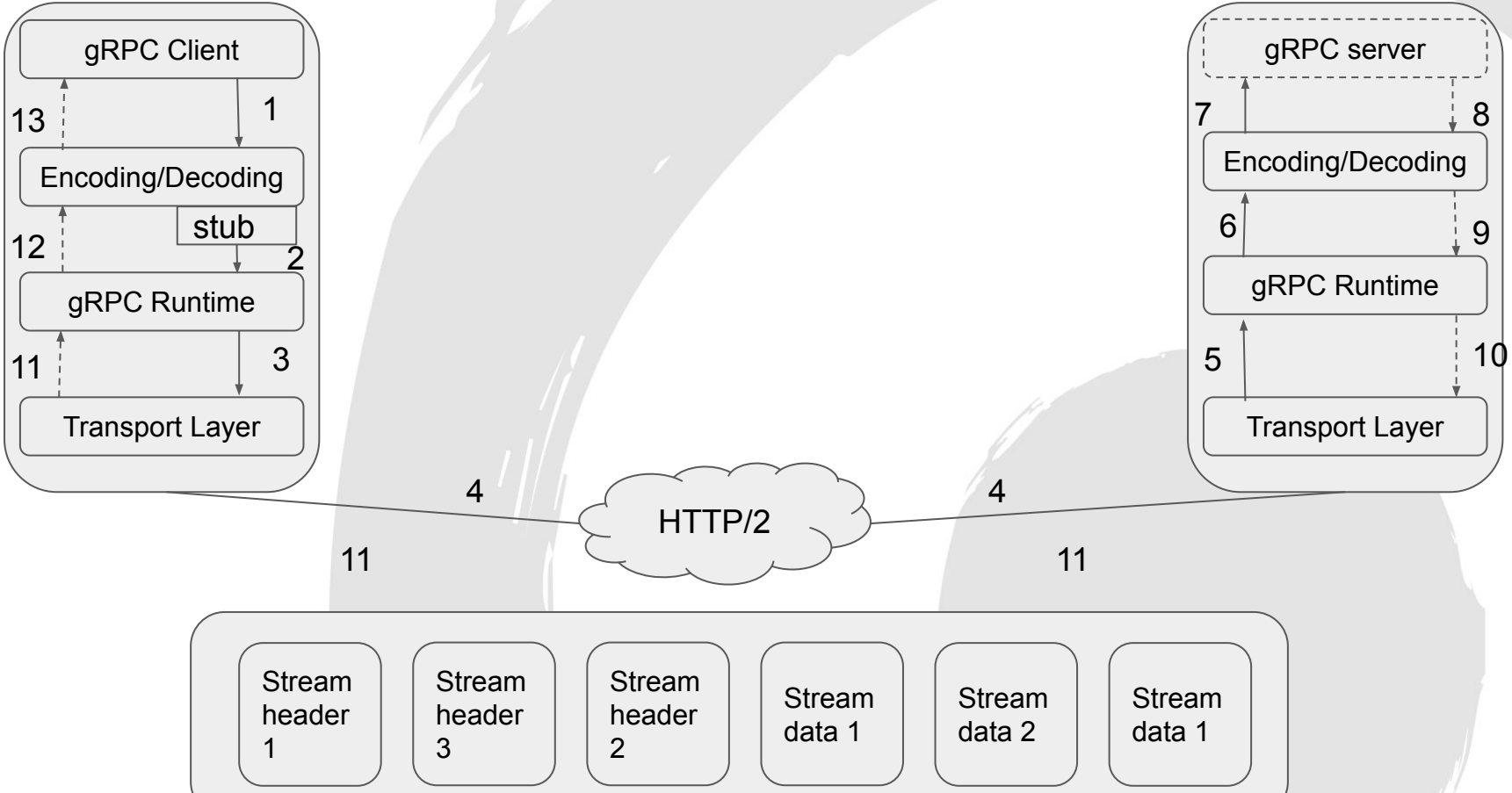
TCP



Only One TCP Connection



Typical Flow



Four kind of service method

- **Unary RPC** : One Request -> One Response
  - rpc SayHello(HelloRequest) returns (HelloResponse);

- **Server Streaming RPC** : One Request -> Stream of Responses
  - rpc SayHello(HelloRequest) returns (stream HelloResponse);

- **Client Streaming RPC** : A sequence of requests -> One Response
  - rpc SayHello(stream HelloRequest) returns (HelloResponse);

- **Bidirectional Streaming RPC** : Both sides send a sequence of messages.
  - rpc SayHello(stream HelloRequest) returns (stream HelloResponse);