gRPC in Elixir

Robert Ellen
@robertellen

Elixir Australia - November 2020

Abstract

- gRPC is ‘A high-performance, open source universal RPC framework’
- inter-service communication
- ‘last mile’ communications to devices and browsers
- any language: HTTP2, Protobuf
- option in the landscape occupied by e.g. ReST, GraphQL, busses, SNMP
- gRPC library for Elixir: Gun (client), Cowboy (server)

Agenda

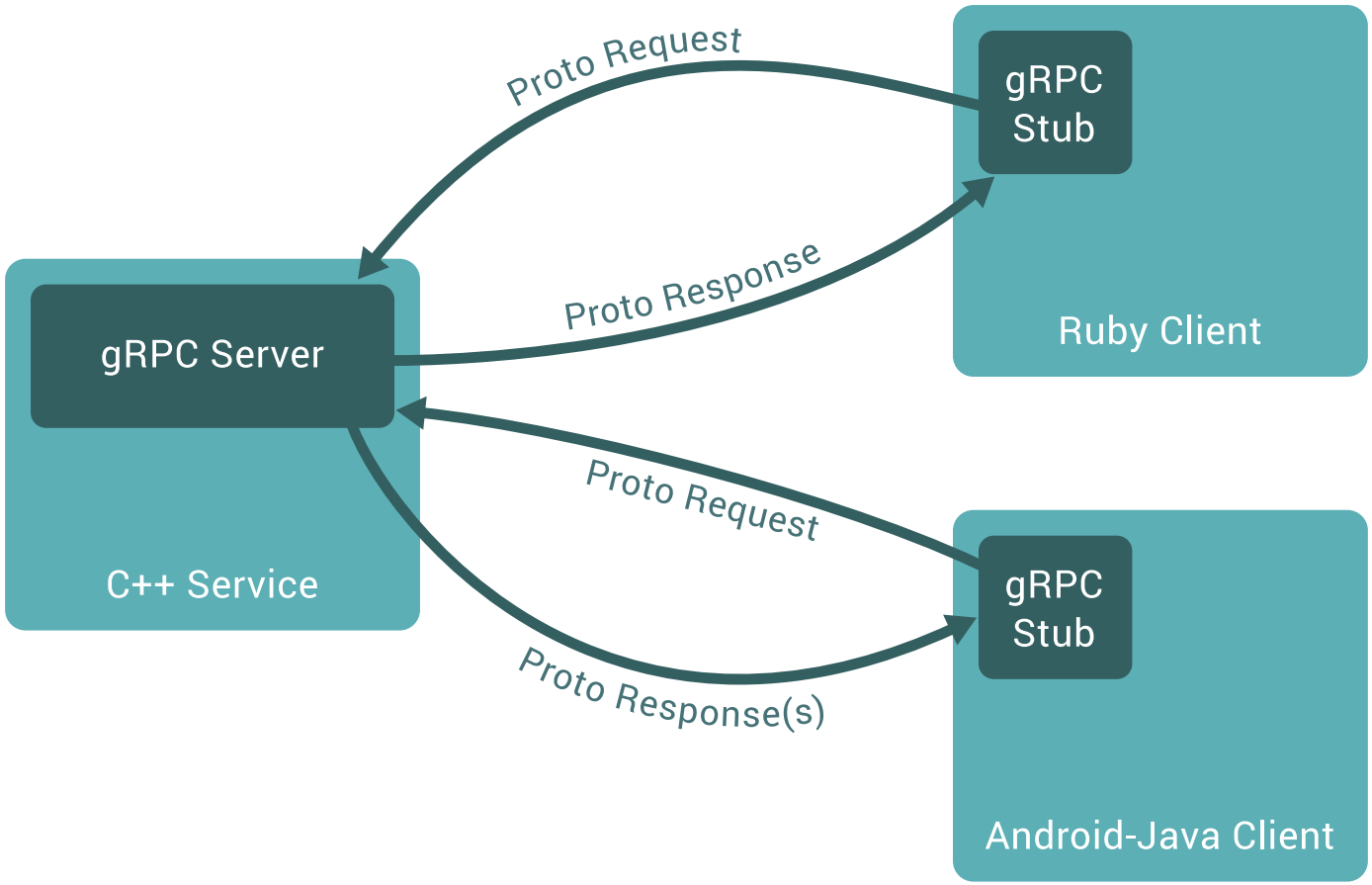
- what is gRPC
- who's using it
- Elixr library (https://github.com/elixir-grpc/grpc) and example application

What is gRPC

RPC framework

- high performance
  - HTTP/2 transport
  - ProtoBuf for IDL and wire format
- multi-language

Diagram



https://grpc.io/img/landing-2.svg

HTTP/2 features

- binary data frames
- compression
- two-way streaming
- TCP-connection multiplexing

Protocol Buffers

- use as IDL provides a strong contract
- codec generation
- compact binary wire format

Protobuf messages

```
message Person {
  string name = 1;
  int32 id = 2;
  bool has_ponycopter = 3;
}
```

https://grpc.io/docs/what-is-grpc/introduction/

Protobuf IDL

```
1  // The greeter service definition.
2  service Greeter {
3    // Sends a greeting
4    rpc SayHello (HelloRequest) returns (HelloReply) {}
5  }
6
7  // The request message containing the user's name.
8  message HelloRequest {
9    string name = 1;
10 }
11
12 // The response message containing the greetings
13 message HelloReply {
14   string message = 1;
15 }
```

https://grpc.io/docs/what-is-grpc/introduction/

RPC types

```
1  // Unary RPC
2  rpc SayHello(HelloRequest) returns (HelloResponse);
3
4  // Server streaming RPC
5  rpc LotsOfReplies(HelloRequest) returns (stream HelloResponse);
6
7  // Client streaming RPC
8  rpc LotsOfGreetings(stream HelloRequest) returns (HelloResponse);
9
10 // Bidirectional streaming RPC
11 rpc BidiHello(stream HelloRequest) returns (stream HelloResponse);
```

https://grpc.io/docs/what-is-grpc/core-concepts/

gRPC server

- gRPC library generates a codec from `.proto` to native data structures
- gRPC server must implement the services specified
- listen on a particular port for a `channel`
- service name and method name form a path
  - $\rightarrow$ load balancing, routing, etc

gRPC client

- codec also generated for client
- plus stubs of the service methods
- client opens a channel to server
- client calls stub like any other function/method

Who's using it?

- inter-service communication
  - as an alternative to ReST, GraphQL, message busses, etc
- APIs
  - Google
  - Cloud-native, k8s, etc
  - applications, e.g. Cockroach DB
- Model-driven Telemetry (to supersede SNMP)
  - Cisco
  - Juniper

Example: Juniper Telemetry Interface

```
1  service OpenConfigTelemetry {
2    rpc telemetrySubscribe(SubscriptionRequest)
3      returns (stream OpenConfigData) {}
4
5    rpc cancelTelemetrySubscription(
6          CancelSubscriptionRequest)
7      returns (CancelSubscriptionReply) {}
8
9    rpc getTelemetrySubscriptions(
10         GetSubscriptionsRequest)
11     returns (GetSubscriptionsReply) {}
12   ...
13 }
```

https://github.com/juniper/jtimon/blob/master/telemetry/telemetry.proto

Example: Async messaging

```
service DataSink {
  // Each message from the request stream will be a HTTP2 POST request
  // for /DataSink/Send path
  rpc Send(stream Request) return (stream Response);
}

// Request can have different payload types
message Request {
  uint32 version = 1;
  bytes message_id = 2;
  oneof payload {
    bytes plain_message = 9;
    bytes zstd_compressed = 10;
  }
}

message Response {
  uint32 version = 1;
  oneof response {
    AckMessage ack = 9;
    ServiceMessage info = 10;
  }
}

message AckMessage {
  repeated bytes message_ids = 1;
}

message ServiceMessage {
  // ...
}
```

https://pl-rants.net/posts/async-over-grpc/

Elixir gRPC library

- https://github.com/elixir-grpc/grpc
- client uses Gun
  - https://github.com/ninenines/gun
- server uses Cowboy
  - https://github.com/ninenines/cowboy
- separate protobuf library
  - https://github.com/tony612/protobuf-elixir

Demo time!

References

- https://grpc.io/
- https://github.com/elixir-grpc/grpc/
- https://blog.appsignal.com/2020/03/24/how-to-use-grpc-in-elixir.html
- https://pl-rants.net/posts/async-over-grpc/
- https://github.com/fullstorydev/grpcurl

Questions

Licence