Bevezetés a gRPC használatába

Élet a SOAP és a REST után

Márk Sági-Kazár

2023-03-22 @ HWSW meetup

`whoami`

Márk Sági-Kazár

Engineering Technical Lead @ Cisco

@sagikazarmark

https://sagikazarmark.hu

Agenda

- About RPC
- RPC vs REST
- Introduction to gRPC
- Implementing a gRPC service
- Advanced features
- When to use gRPC?
- Alternatives

Code samples

https://github.com/sagikazarmark/grpc-intro-workshop

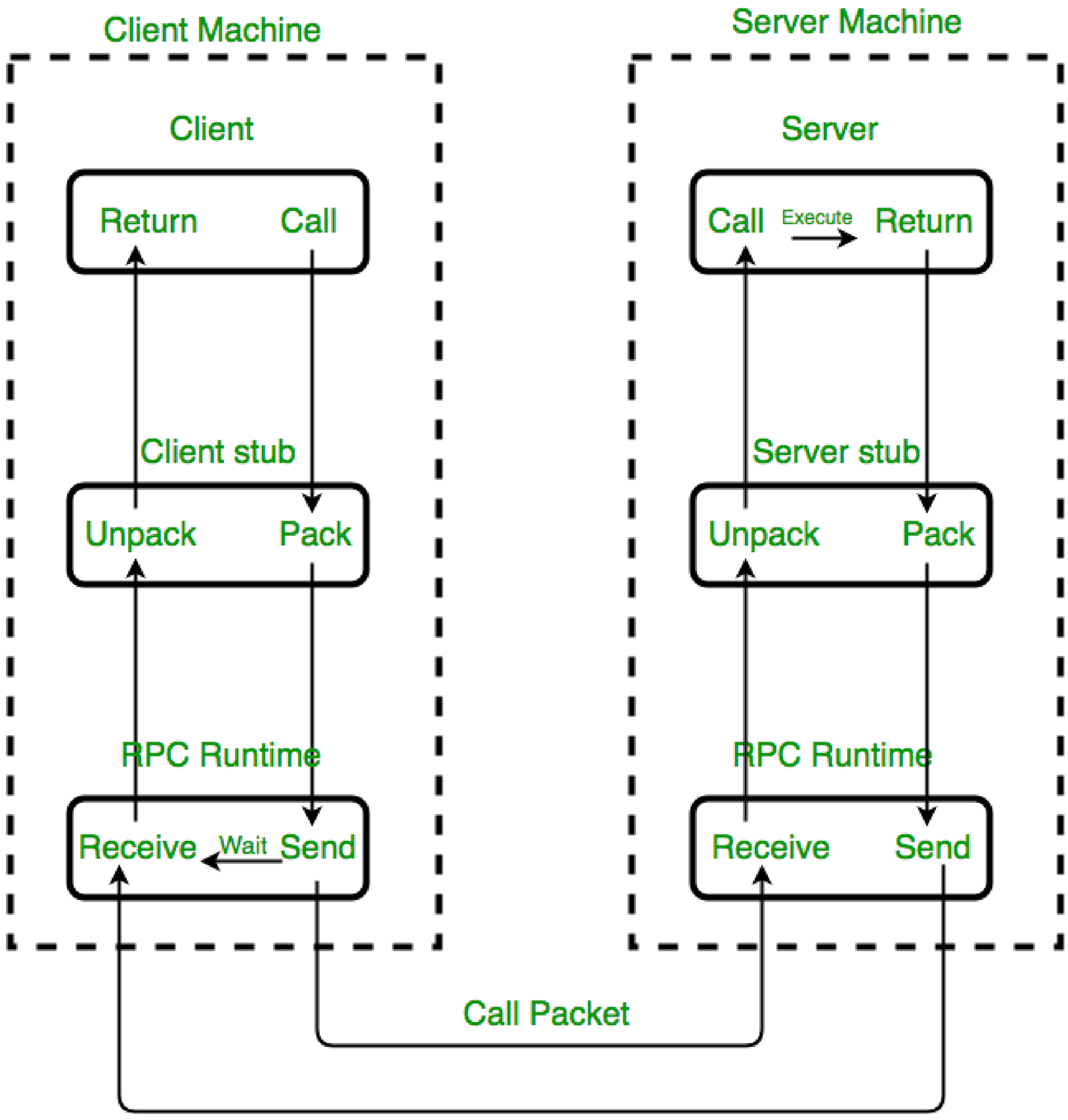
About RPC

Definition 🤔

> […] a computer program causes a procedure (subroutine) to execute **in a different address space** (commonly on *another computer on a shared network*), which is coded as if it were a **normal (local) procedure call**, **without** the programmer explicitly coding the **details for the remote interaction**
>
> *Wikipedia*

Definition

- **in a different address space** → someone else's computer
- **normal (local) procedure call** → interface
- **no details for the remote interaction** → implementation



Implementation of RPC mechanism

Interface Definition Language

- a language for describing communication
- programming language agnostic
- serves as a contract for RPC APIs

RPC vs REST

	RPC	**REST**
Operates on	procedures ("function calls")	resources
Underlying protocol	unspecified	primarily HTTP
Interactions	protocol encapsulated	HTTP is part of the contract

- RPC is more suitable for APIs describing *actions* (vs *resources*)
- RPC has better type-safety guarantees
- RPC is more strict
- REST may be more performant when operating on large amount of data

RPC example in HTTP

```
POST /sayHello HTTP/1.1
Host: api.example.com
Content-Type: application/json

{
  "userId": 1
}
```

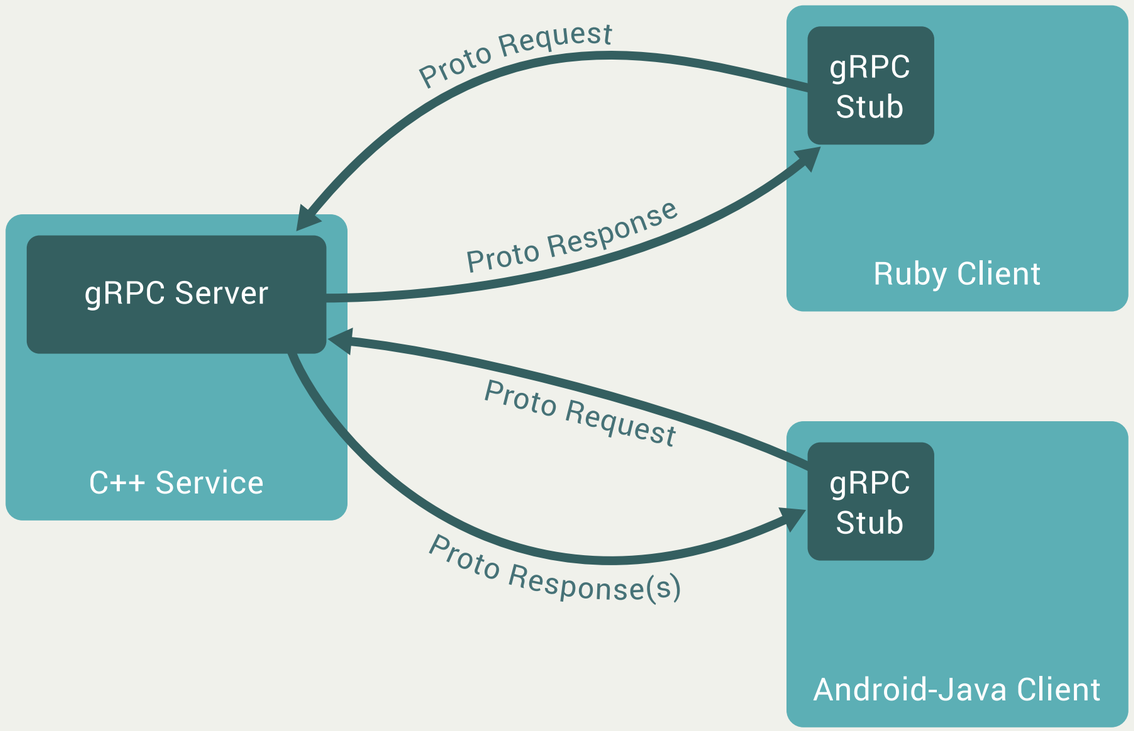
REST example

```
GET /users/1/greeting HTTP/1.1
Host: api.example.com
```

Introduction to gRPC

About gRPC

- High performance
- Language agnostic (supported languages)
- Uses http/2 transport
- Bi-directional streaming
- Pluggable auth, tracing, load balancing and health checking


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

Source

IDL: Protocol Buffers

- Interface Definition Language
- Message format (binary serialization)
- Code generation framework

Example 1: Protocol Buffers

Implementing a gRPC service

Example 2: Service definition and server implementation

Example 3: Using the client stub

RPC lifecycle

- *Unary RPC*
- Server streaming RPC
- Client streaming RPC
- Bidirectional streaming RPC

Example 4: Server streaming RPC

Advanced features

- Authentication
- Metadata
- Error handling

Example 5: Authentication and error handling

Interceptors

- Act as middlewares during the request lifecycle
- Server vs Client
- Unary vs Stream

Example 6: Server unary interceptor

When to use gRPC?

> With great power comes great responsibility

- API operates on *actions* instead of *resources*
- When efficient communication is a goal
- In polyglot environments
- No need for supporting a wide range of clients
- Internal APIs

whisper microservices

gRPC weaknesses

- Limited web/browser support
- Not human-readable format (more difficult to debug)
- Steeper learning curve

Alternatives

Twirp

Developed at Twitch as a lightweight alternative to gRPC.

https://github.com/twitchtv/twirp

Read the announcement blog post.

Connect

A gRPC-compatible framework with an emphasis on browser- and web-compatibility (from the creators of Buf).

https://connect.build/

The End

Any questions?

@sagikazarmark

https://sagikazarmark.hu

Further reading

- https://grpc.io/
- https://protobuf.dev/
- https://github.com/grpc-ecosystem