KubeCon

CloudNativeCon

China 2018

gRPC-Go: Architecture and Features

About me

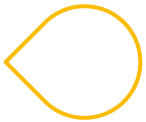
Yuxuan Li（李雨璇)

Software Engineer, Google

@lyuxuan on Github

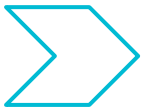
What is gRPC?

A modern open source high performance RPC framework



Multi-Language

+ Java, Go, C/C++, C#, Node.js, PHP, Ruby, Python, Objective-C



Pluggable

+ auth, tracing, resolver, load balancing, IDL, health checking

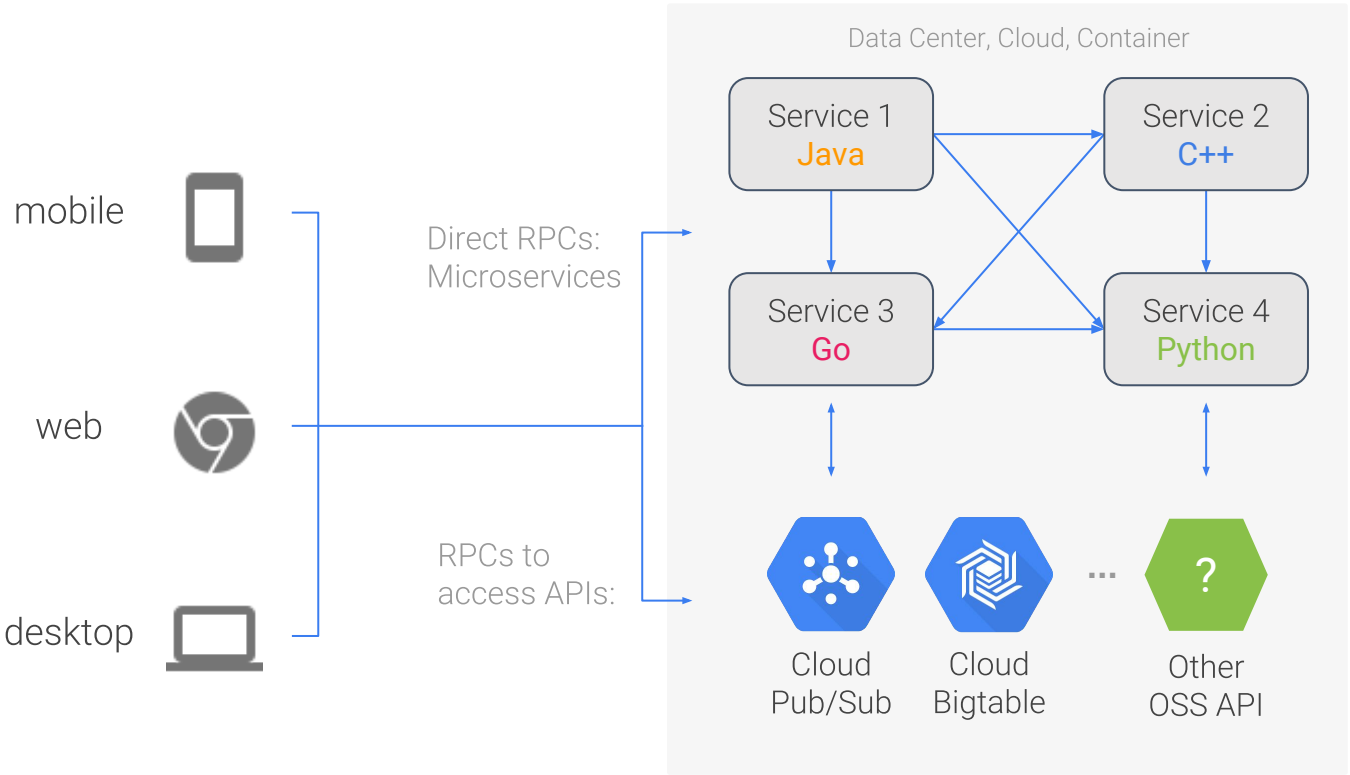
Multi-Platform

+ Linux, Windows, Mac OS X, iOS, Android

Feature-rich

+ binary logging, channelz, tracing, retry, service config

gRPC Use Cases



gRPC Adopters

Microservices: in data centres

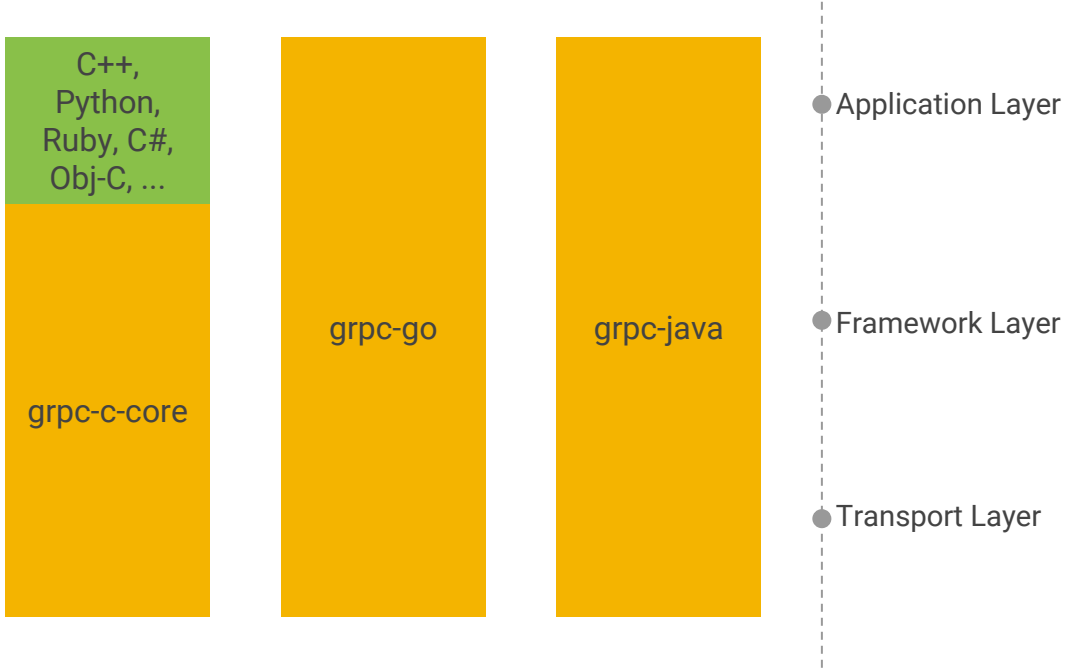
Client Server communication/Internal APIs

Streaming telemetry from network devices

Mobile Apps

Implementations

- Language Idiomatic APIs in 9 languages
- Natively implemented in 3 different languages (C, Java and Go)
- Other than Java and Go, all other language implementations are wrapped around the gRPC C core implementation.

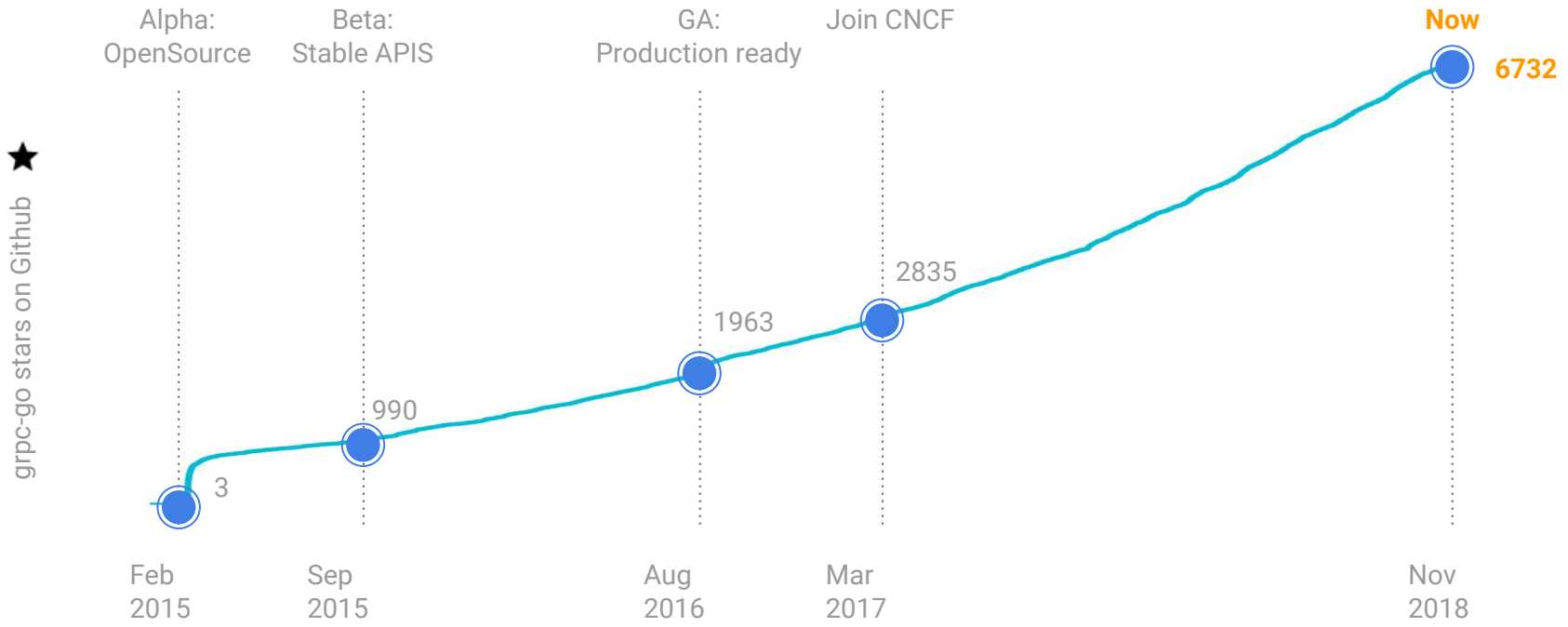


Implementations

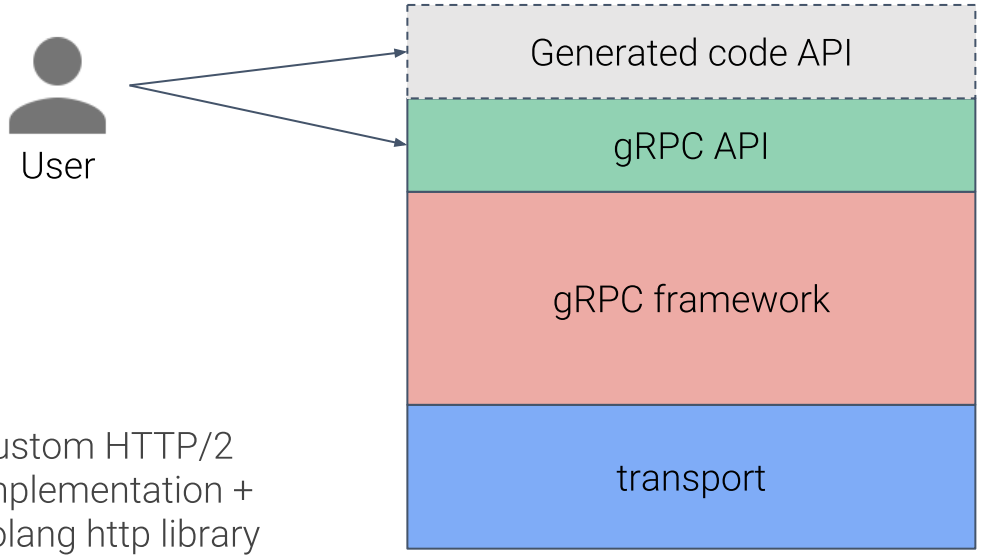
- Language Idiomatic APIs in 9 languages
- Natively implemented in 3 different languages (C, Java and Go)
- Other than Java and Go, all other language implementations are wrapped around the gRPC C core implementation.

This talk is about grpc-go

History of gRPC(-Go)



Full stack



Custom HTTP/2
Implementation +
golang http library

RPC operations and types

KubeCon CloudNativeCon China 2018

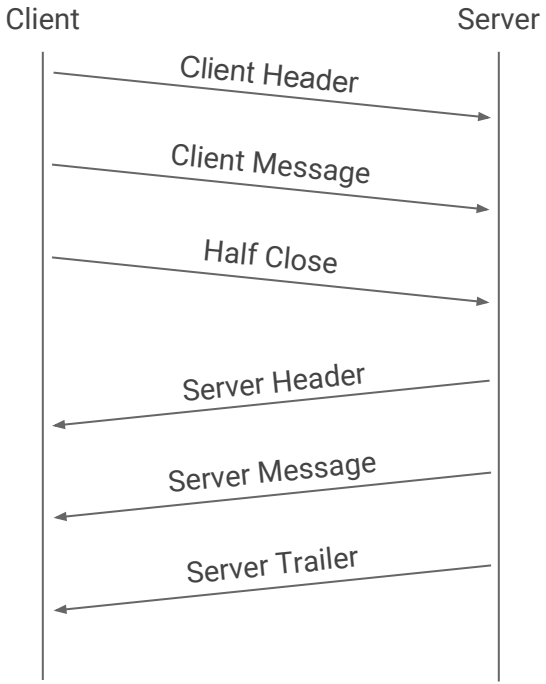
Operations:

Client	Server
Send Header	
Send Message	
Send Half Close	Send Trailer

Four types of RPC:

1. Unary
2. Client Streaming
3. Server Streaming
4. Bidi Streaming

1, 2, 3 are all special cases of 4



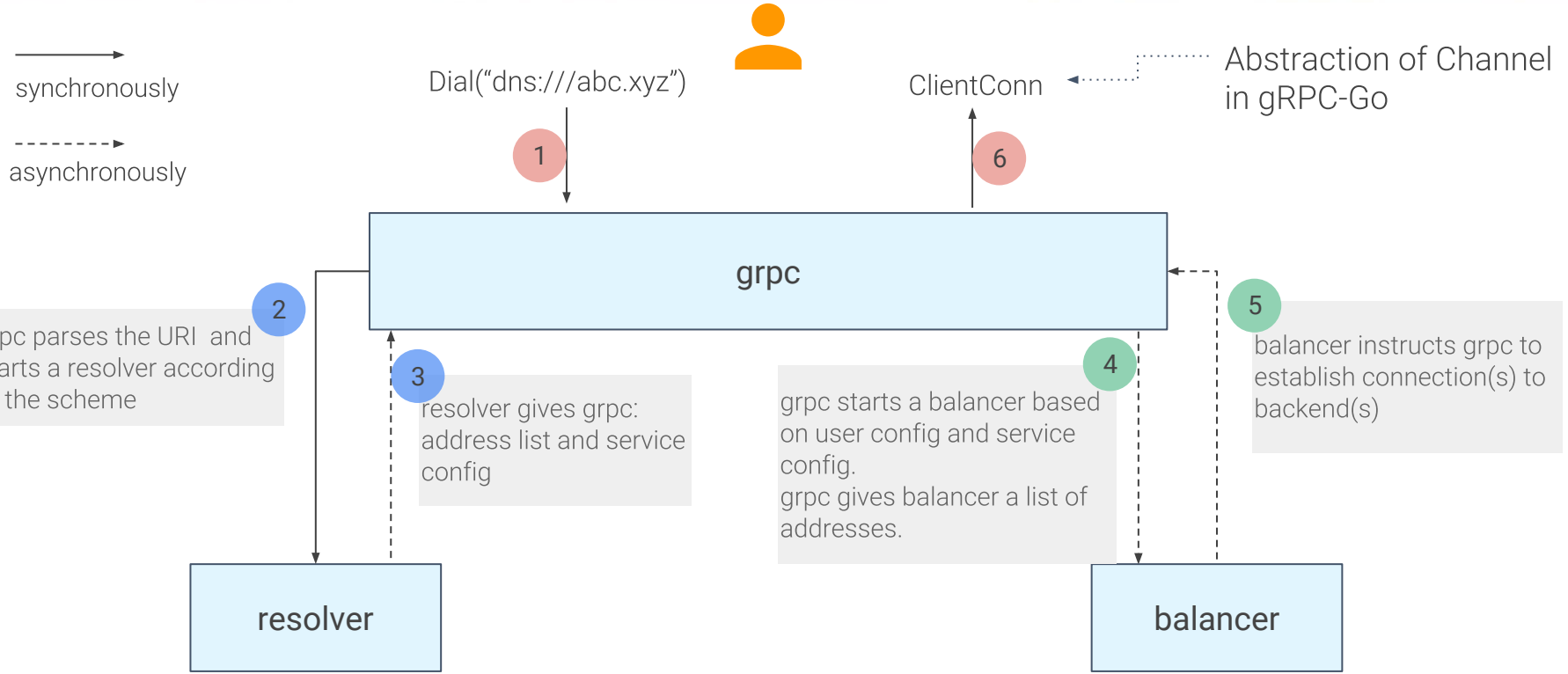
A normal Unary RPC

Life of an RPC: Channel

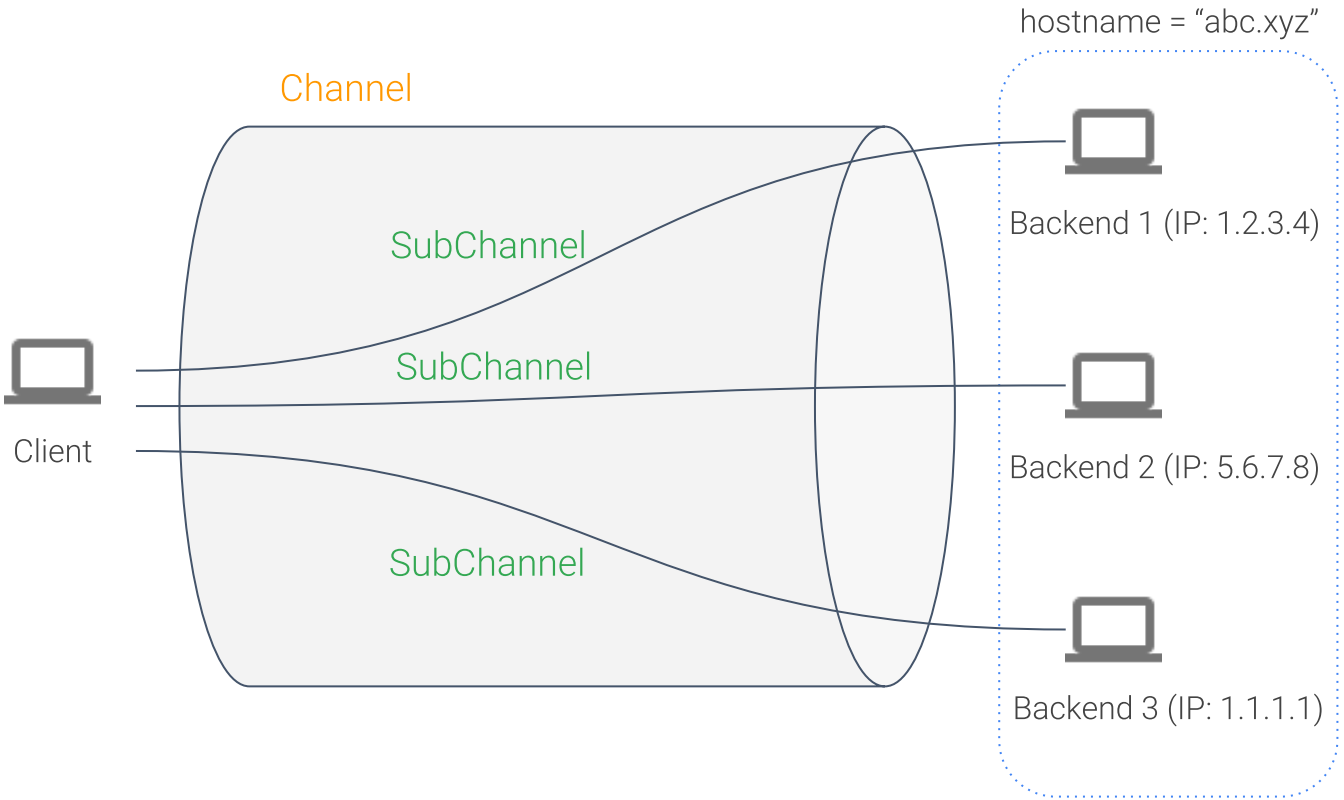
First Step

Create a Channel to transmit the RPC

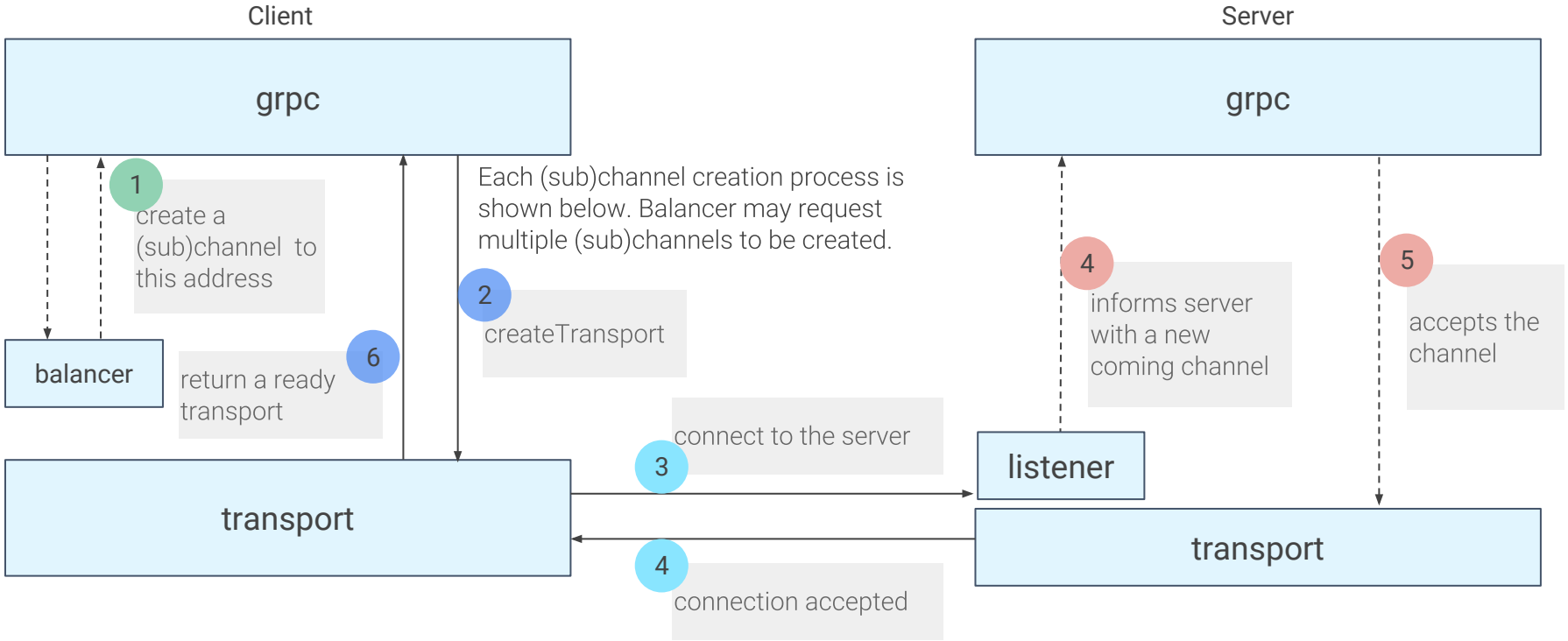
Life of an RPC: Channel



Channel and SubChannel



Life of an RPC: Channel

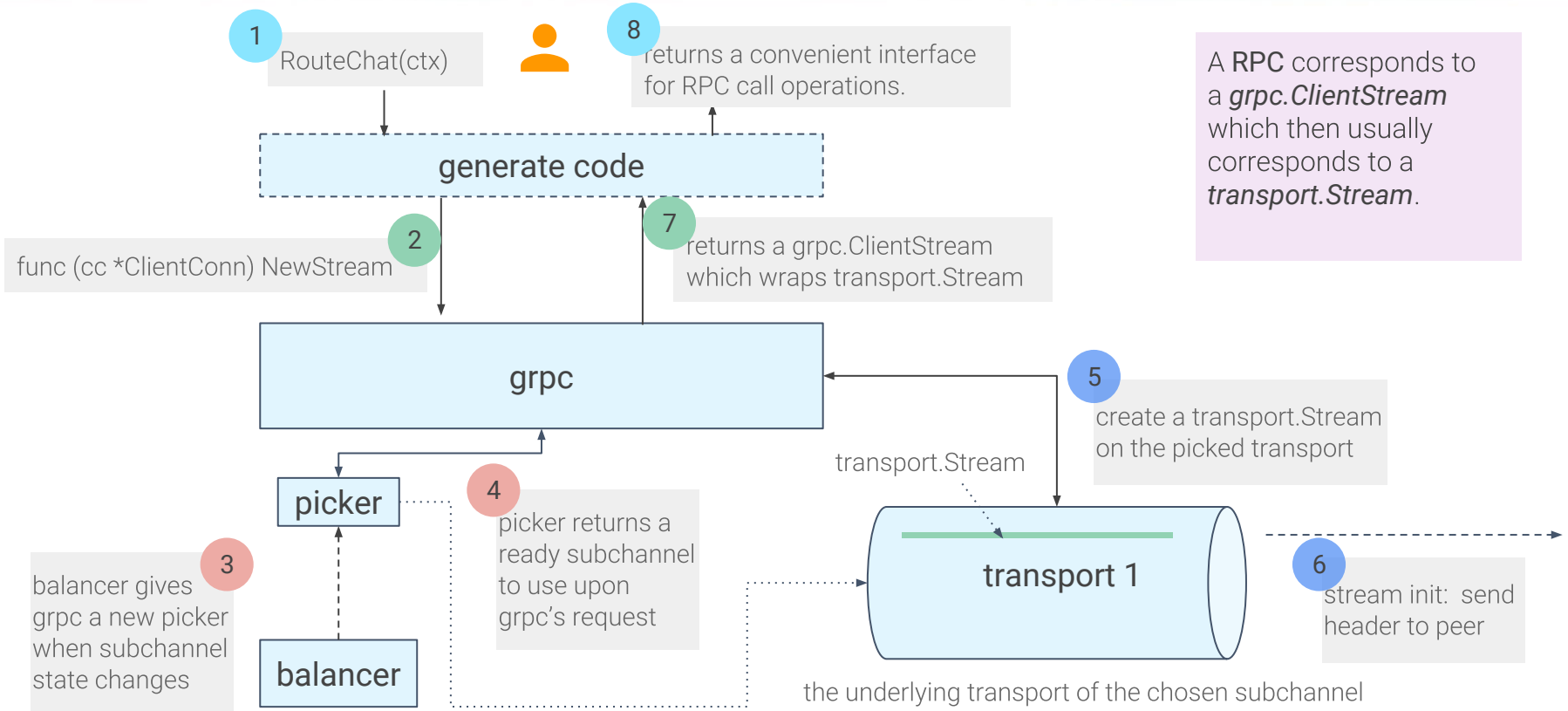


Life of an RPC: Call

Second Step

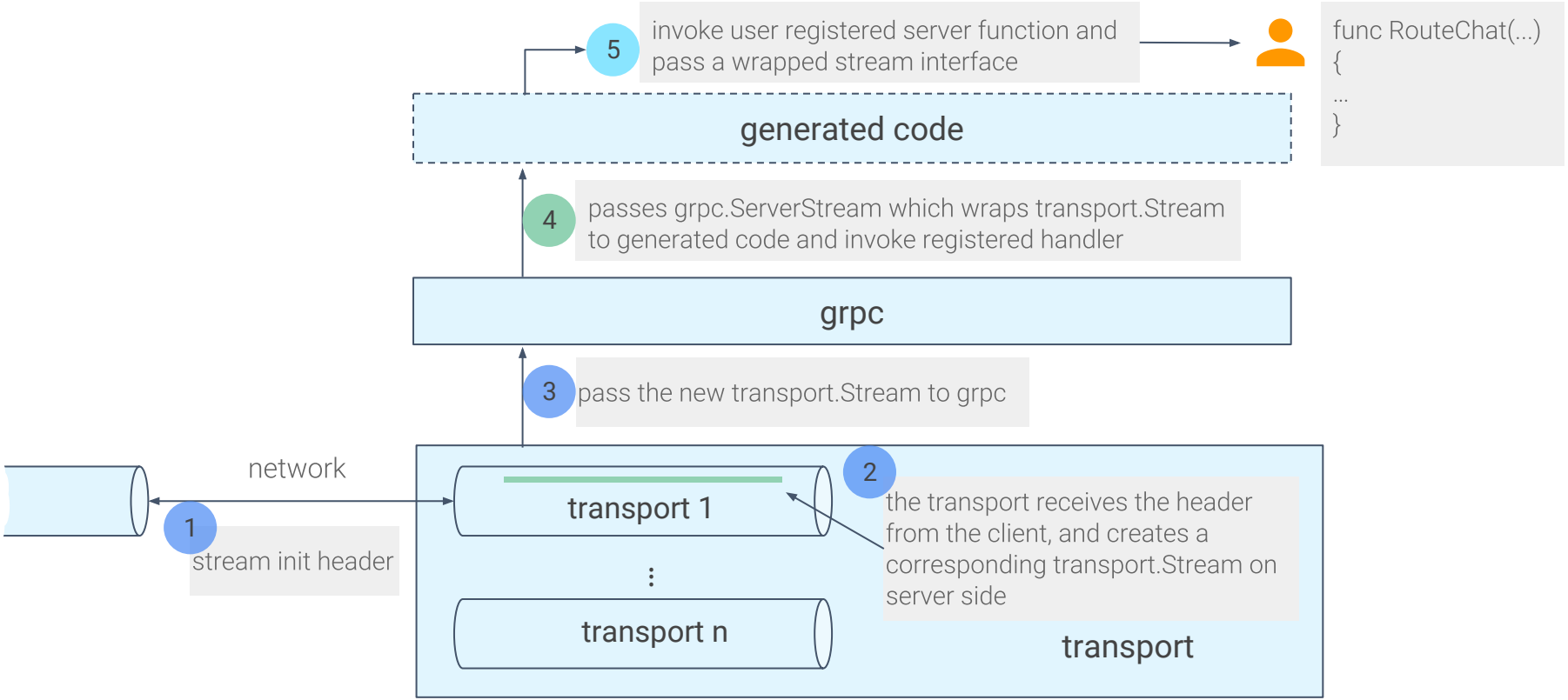
Create a RPC call on the Channel

Life of an RPC: Call (client side)

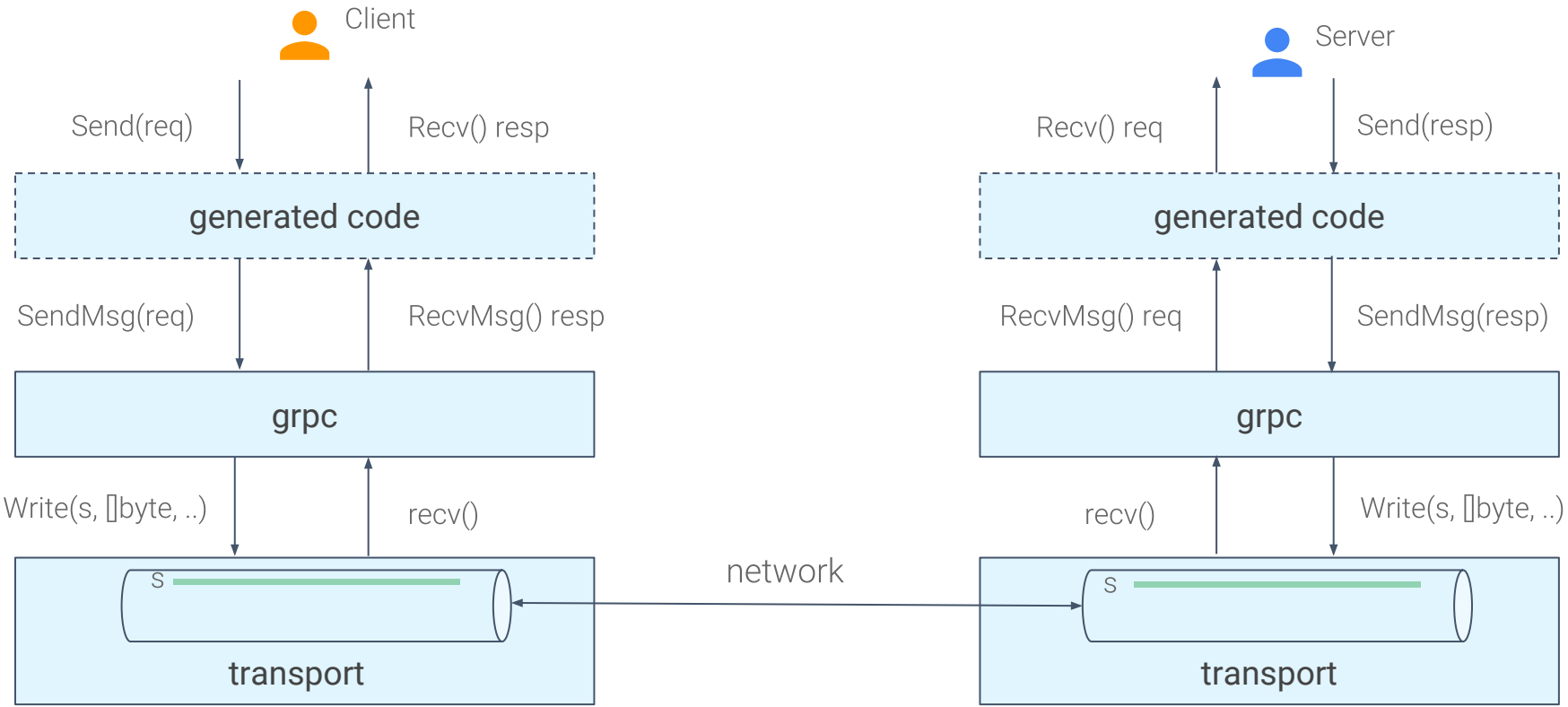


A RPC corresponds to a *grpc.ClientStream* which then usually corresponds to a *transport.Stream*.

Life of an RPC: Call (server side)



Life of an RPC: Data Flow



Features

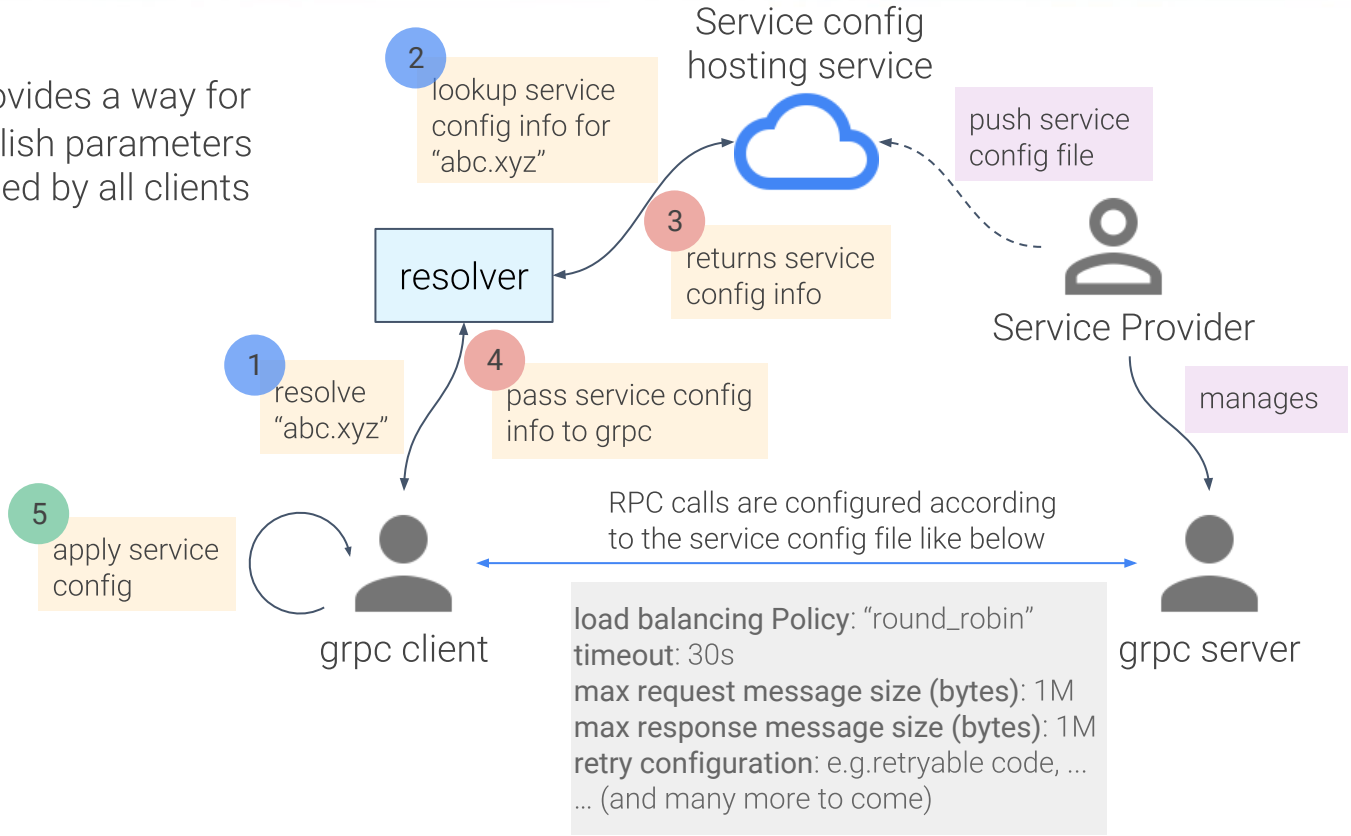
- Pluggable components
  - Resolver, balancer, IDL, compressor, codec, transport (to be), etc.

- Rich features
  - **Binary Logging**
  - **Channelz**
  - Health Checking
  - RPC retry
  - Tracing (e.g. OpenCensus)
  - **Service Config**

Please find more info (design doc, discussions) about gRPC features in the https://github.com/grpc/proposal repository.

Service Config

Service Config provides a way for service owners to publish parameters to be automatically used by all clients of their service.

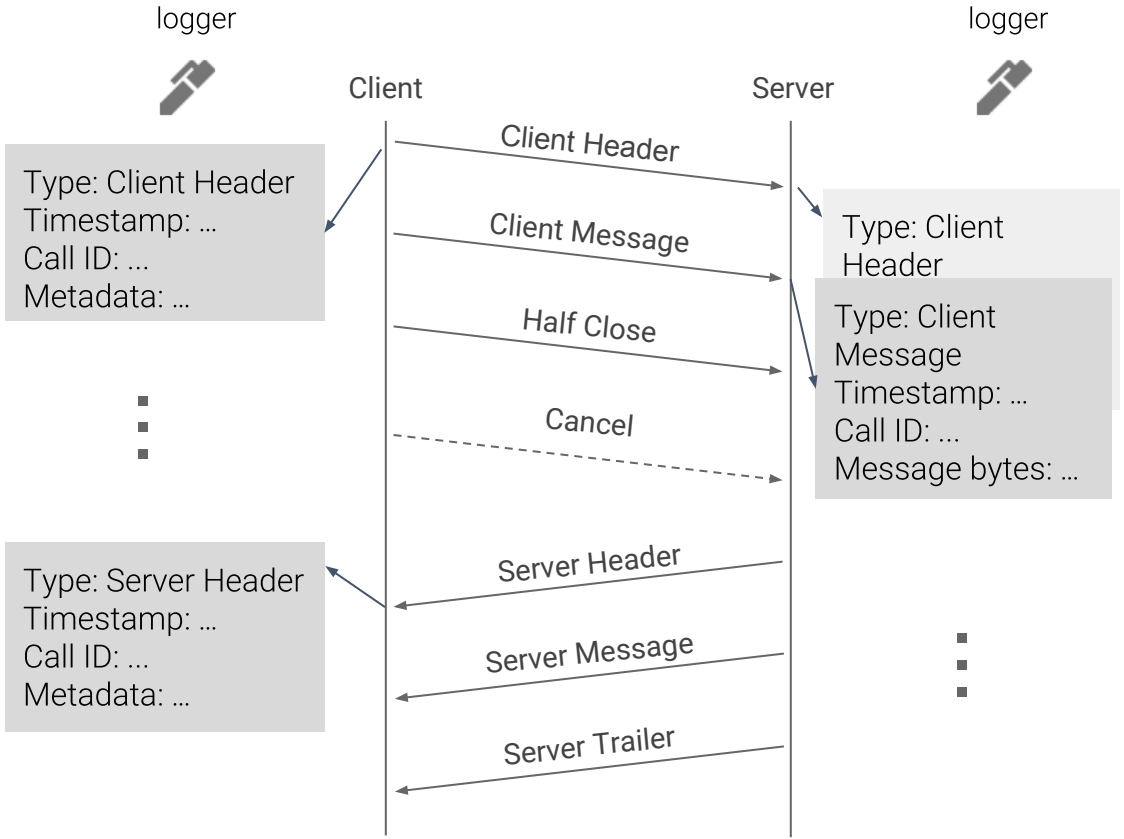


Binary Logging

Binary logging logs RPCs in binary format.

typical use cases:

- Troubleshooting services, finding exceptions
- Load testing
- Replaying RPCs from production

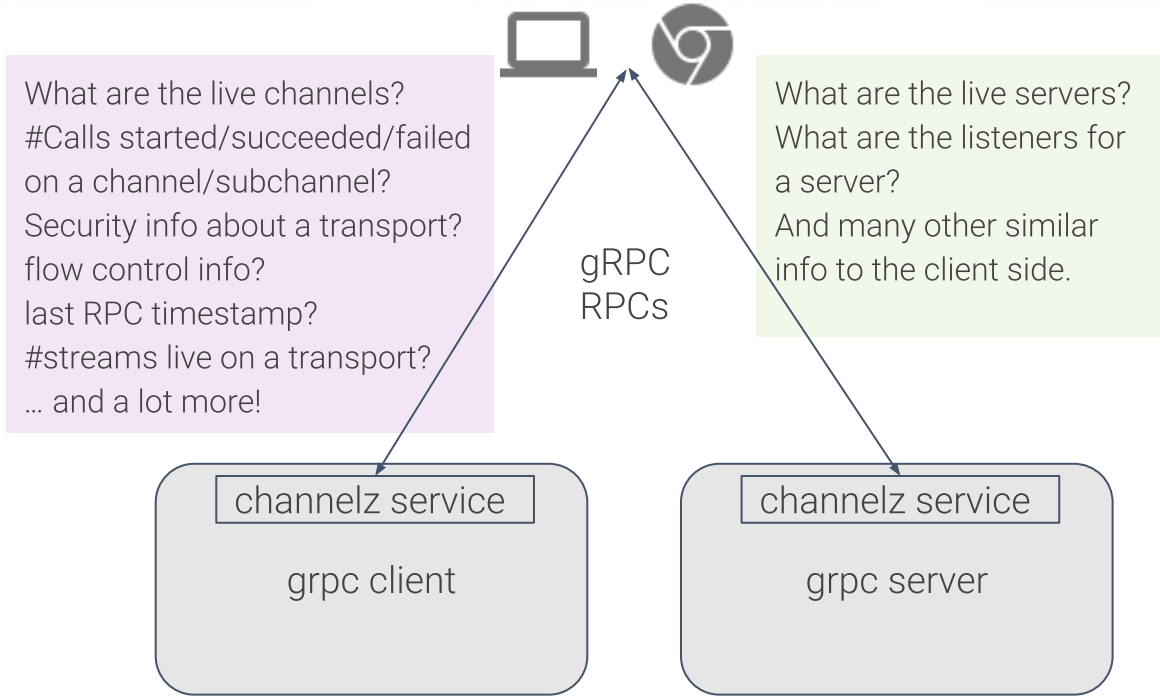


Channelz

Channelz is a tool that provides comprehensive runtime info about connections at different levels in gRPC.

It is designed to help debug live programs, which may be suffering from

- network issue
- performance issue
- configuration issues
- ...



See https://grpc.io/blog/a_short_introduction_to_channelz for a detailed introduction and demo.

Thank you

KubeCon CloudNativeCon China 2018

Q & A

http://http2.golang.org/gophertiles

HTTP/1.1 HTTP/2

Twitter: **@grpcio**
Site: http://grpc.io
Group: grpc-io@googlegroups.com
Repo: github.com/grpc

KubeCon

CloudNativeCon

China 2018