gRPC

The next step for interservice communication

What is RPC?

RPC is short for Remote Procedure Call. It is when a computer program causes a procedure to execute in a difference space.

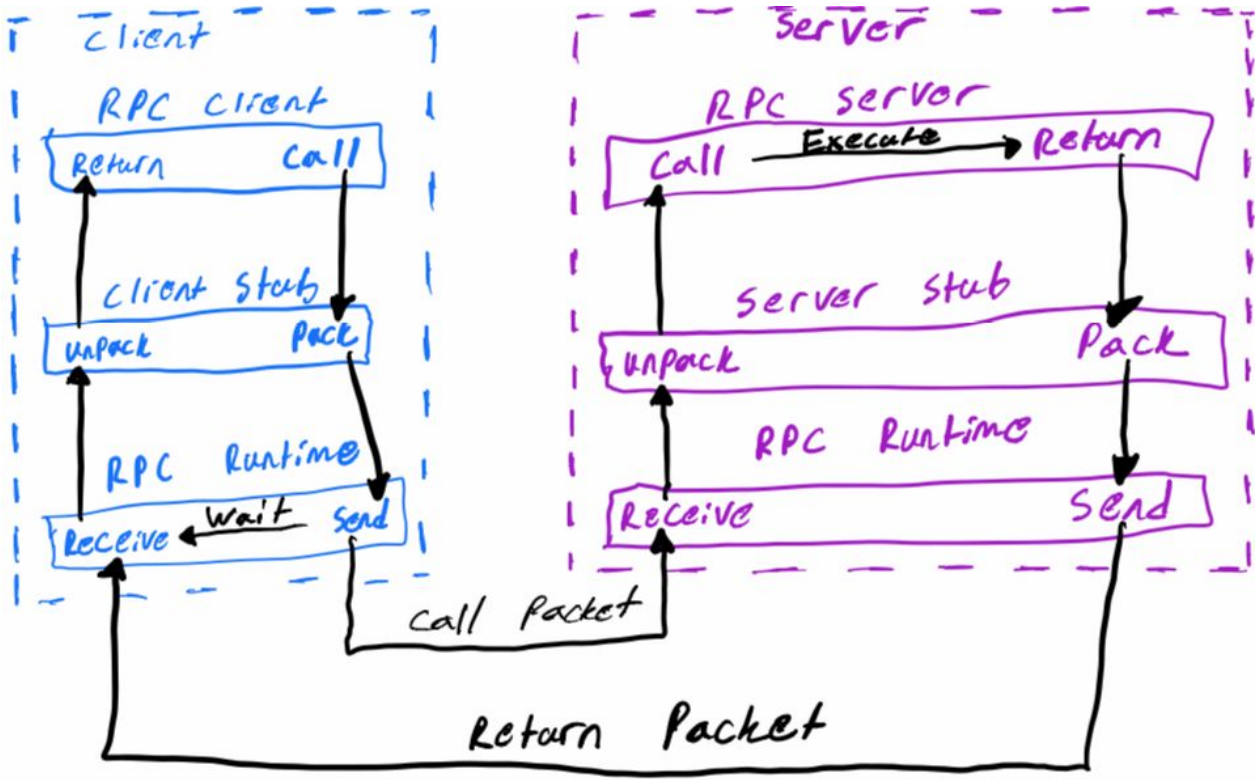
It is commonly used as a form of client-server interaction and it is the basis for gRPC.

Source: https://en.wikipedia.org/wiki/Remote_procedure_call

RPC Types

- Unary
  - Simplest type where the client sends a single request and receives a single response
- Server Streaming
  - Similar to the Unary type except rather than receiving a single response the client will receive a stream of responses
- Client Streaming
  - The client will send a stream of requests and receive a single response
- Bidirectional Streaming
  - Initiated by the client, requests and responses are sent back and forth between the client and server

RPC Communication Diagram



What is a Protocol Buffer?

Protocol Buffers are a language agnostic, platform neutral, extensible mechanism for serializing structured data.

Think of them as a contract that is utilized by both the client and server for communication.

What Languages Do Protocol Buffers Support?

As of proto3 the following languages are officially supported:

- Java
- Dart
- Objective-C
- C++
- C#
- Python
- Go
- Ruby (First party compiler coming soon!)

Source: https://developers.google.com/protocol-buffers

What about Javascript?

While not officially supported, there are packages which will handle loading proto files.

One such example is google’s NodeJS library (@grpc/proto-loader).

What Does a Protocol Buffer Look Like?

```
syntax = "proto3";

package helloworld;

service Greeter {
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}

message HelloRequest {
  string name = 1;
}

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

What is gRPC?

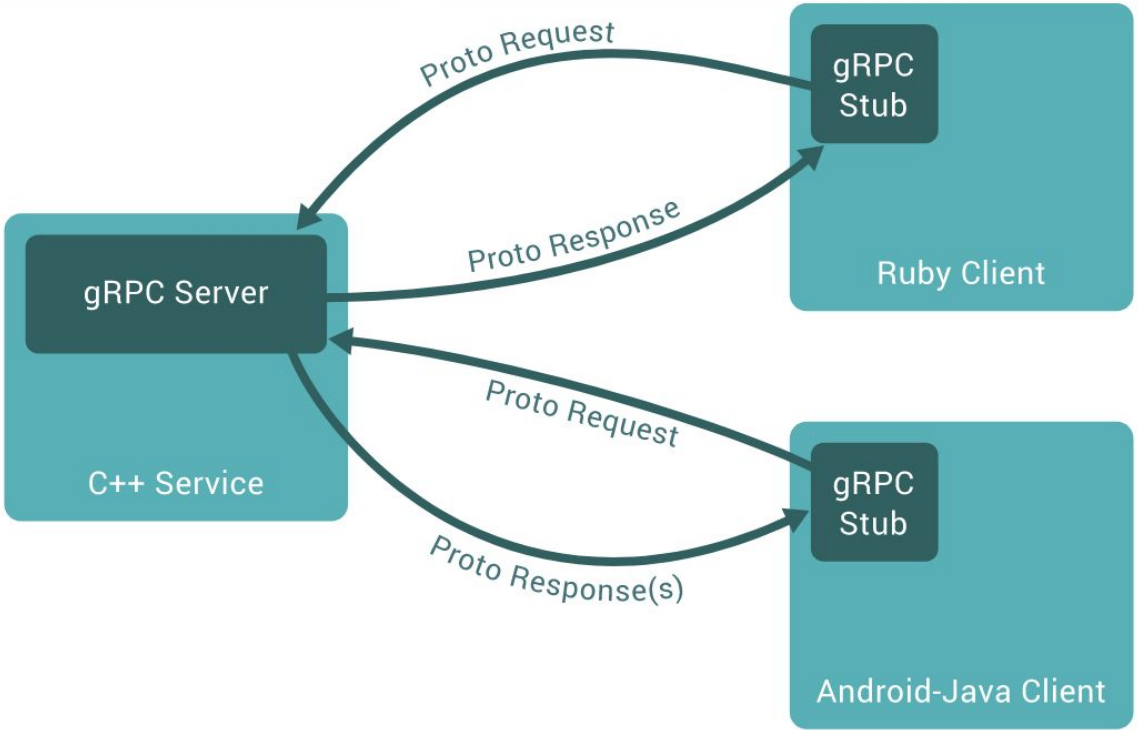
---

gRPC is an open source remote procedure call system initially developed by google.

It makes use of HTTP/2 for transportation and Protocol buffers as the interface description language.

Source: https://en.wikipedia.org/wiki/GRPC

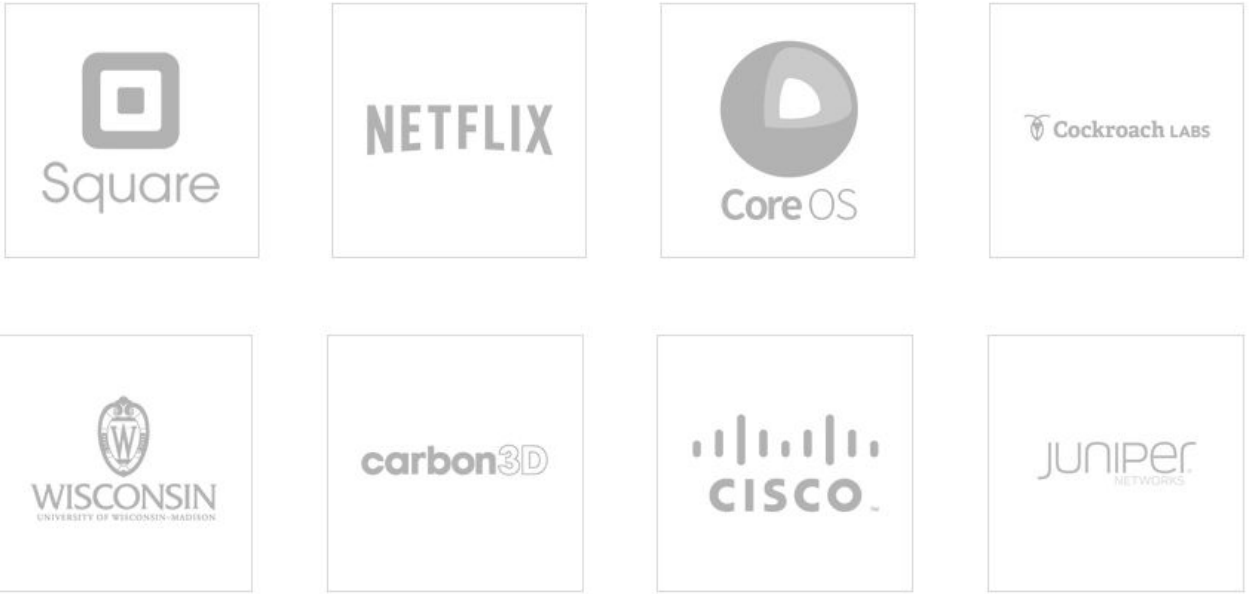
gRPC Overview



gRPC VS REST

gRPC	REST
Uses HTTP/2 (Fast)	Uses HTTP 1.1
Strongly typed	Sensitive to latency
Uses a language and platform neutral contract that defines the interfaces and structured requests for requests and responses	TCP handshake required for each request (Increased bandwidth usage and page load time)
Support for streaming	Only supports request-response communication method
Recent benchmarks have shown gRPC to be roughly 7 times faster at receiving data and 10 times faster at sending data vs REST	

Who is Using gRPC?

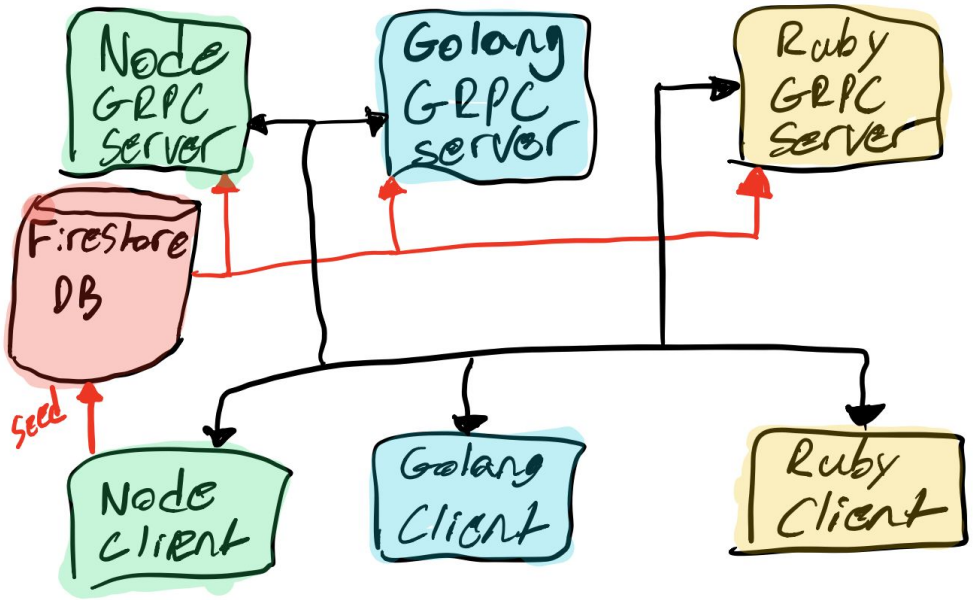


Source: https://grpc.io/

Let’s Use gRPC Today!

Hold on now! While gRPC has quite a few benefits over REST, it is not yet widely adopted and proto3 has yet to officially support Javascript. In the meantime we can take advantage of it’s benefits for mobile clients and interservice communication which I will demo in the next slide.

Demo



See: https://github.com/RyanLafferty/hello-grpc/