#ASLI ENGINEERING

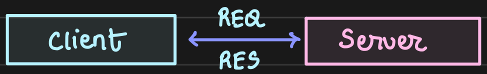
RPC - Remote Procedure Calls

SWIPE

BY

ARPIT BHAYANI

Remote Procedure Calls

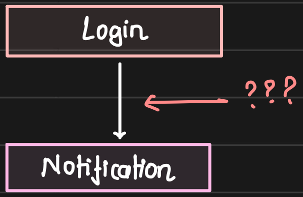


This design is not RPC specific, instead it spands the entire distributed systems paradigm

Client wants something to be done, it sends the request to the server, the server computes and sends the response

So then, what is RPC ?

Notifying a user Say, you are a bank and you have to synchronously send OTP to your user to log them in.



```
def login(email, password):
    ────────
    ────────
    ────────
    ────────
    notification_otp(email)
```

How would we implement this ?

The function notification_otp might make an HTTP call using REST to talk to the notification service and send to OTP.

Core problem

Every language has its own HTTP library like requests and every one who has to talk to notification has to use a a client, make the call, retry, handle failures

ARPIT BHAYANI

what if we abstract out the repetitive and mundane tasks
like communication protocol, object creation, failures, retries,
compression, streaming, etc

and just focus on writing the core business logic in a function body that looks like a local procedure call.

This is how RPC was conceptualized...

```
def login(email, password):
    ────────────
    ────────────
    ────────────
    ────────────

    notification_otp(email)
```

what looks like a regular function call is infact a Remote Procedure Call that happens over the network through stubs.

It looks like a routine function call but infact it talks over the network with the notification service to send OTP over email to the user.

$\boxed{\text{Location Transparency}}$

Core idea is to hide the complexity of a remote call.

* RPC are magnitude slower than local procedure calls

  - marshalling and unmarshalling
  - network packet movement

Stubs :

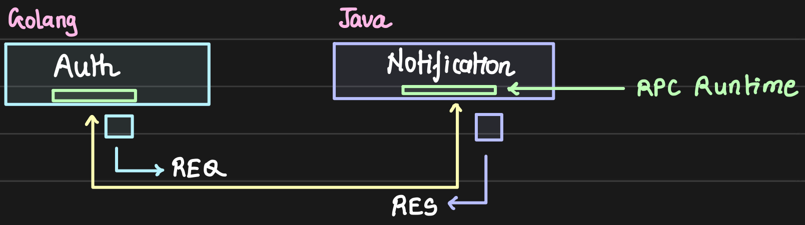
```
def login(email, password):


    notification_otp(email)
```

\`login\` is on the Auth service

\`notification-otp\` is on the Notification Svc

Responsibility of a Stub → someone needs to convert (serialize) the information from Auth service into "some" format that can be sent over the wire; and convert it back to native objects on Notification Service



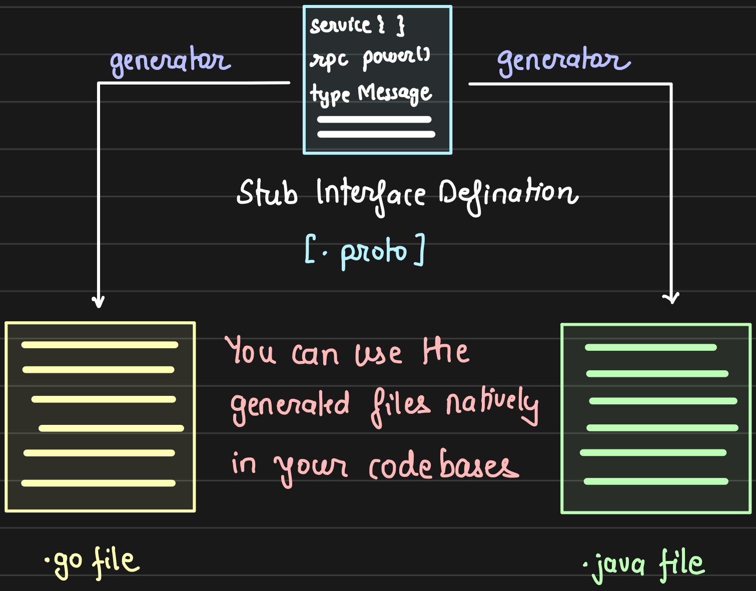
The stub converts its methods, request types, response types into form used by the RPC system.

Send Notification Request	Send Notification Response
Golang Struct in Auth Svc	Java Class in Notification Svc

Stub is the main reason why the remote procedure appear local.

ARPIT BHAYANI

Stub Interface Description language



We first write the Interface Definition.

We then write the server program that implements the interface.

We then write the client program that uses the same interface to talk to the server.

Communication in RPC

RPC can use any layer 3 communication protocol TCP or UDP

The idea of choosing one over other would be

- core features
- performance

* RPC can also happen over HTTP

eg: gRPC uses HTTP/2 as a transport

ARPIT BHAYANI

Why Remote Procedure Calls?

- easy to use and strong API contract
- invoke remote call just like a local call boosts dev productivity
- most RPC frameworks, like gRPC, supports most modern languages and multi-language communication
- most mundane tasks are abstracted
  - failures
  - retries
  - payload conversion to native objects
  - n/w protocol and multiplexing
- performance out of the box
  - compression
  - connection pool
  - multiplexing
  - streaming
  - efficient payload
- security is a plug
- you don't need to write client-libs, they are auto-generated

Concerns while using RPC

- Stubs need to be regenerated when signature changes
- testing RPC is non-trivial
- getting started is a little challenging
- browser support is limited

ARPIT BHAYANI