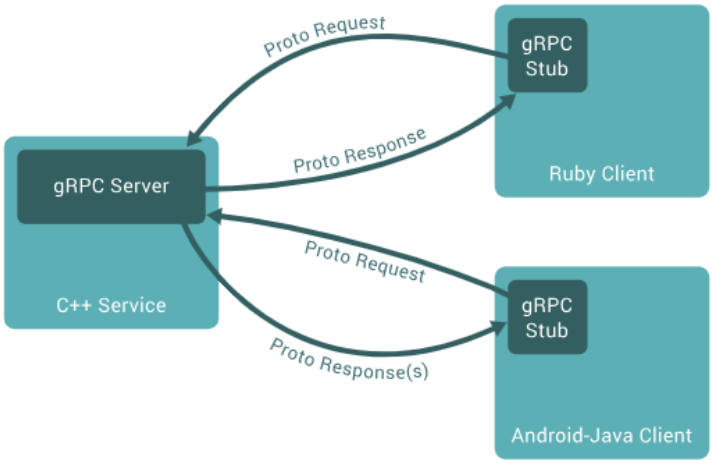
Coste Corina-Teodora,

Grupa 2.1, Anul 4, IS

Middleware - obiecte distribuite gRPC comunicare simpla

Middleware este o tehnologie care ușurează comunicarea dintre client și server. Acesta mai este denumit și "*software-ul lipici*" deoarece permite conectarea diferitelor componente software dintr-o aplicație.

gRPC-ul (Remote Procedure Calls) este o platforma dezvoltată de către Google în 2015. Cu ajutorul lui aplicațiile client-server pot comunica transparent, ușurând creearea sistemelor interconectate.



În imaginea de mai sus avem un exemplu de o astfel de conexiune folosind acest framework.

Acest framework este compus din 2 componente:

- Protocol buffers
- gRPC Protocol

Aplicațiile pot fi scrise în toate limbajele suportate de către gRPC, putând de exemplu, să îți creezi un server în C++ și să ai clienți în Ruby, Python etc., în felul acesta putându-se crea funcționalități cu mai multă ușurință.

Pe lângă cele prezentate mai sus, acest framework dispune de câteva feature-uri care se recomandă să fie utlilizate în orice aplicație, precum:

- Asigură o comunicare strânsă între client și server
- API-ul poate să fie modificat astfel încât să nu afecteze clientul
- Asigură comunicarea simpla și bazată pe streamuri cu suport bidirecțional de streaming
- Generează cod pentru mai multe limbaje de progamare
- Se integrează ușor cu tehnologiile cloud

Acestea sunt doar câteva din ele, cu toate acestea exista și dezavantaze, cum ar fi:

- Inexistența suportului pentru protocolul Payload compression
- Incapabilitatea de a folosi alte IDL (Interface Description Language) în afară de protocol buffers

Comunicarea simplă despre care am menționat mai sus se referă la o comunicare sincronă făcută de către client cu blocare, acest lucru referindu-se la faptul că atunci când serverul primește apelul de la client cu block, serverul nu poate trece la alt apel până când nu îi răspunde primului client. În termeni simplificați asta înseamnă că un client face un apel către server și asteaptă răspunsul.

Pentru o mai bună înțelegere a conceptului de mai sus, avem următorul exemplu:

Presupunând că avem definite interfața și structura mesajului

```
service HelloService {
  rpc SayHello (HelloRequest) returns (HelloResponse);
}

message HelloRequest {
  string greeting = 1;
}

message HelloResponse {
  string reply = 1;
}
```

gRPC ne permite să alegem service method-ul dorit, în cazul nostru comunicarea simplă:

- Clientul trimite un request către server si primește un singur răspuns înapoi

```
rpc SayHello(HelloRequest) returns (HelloResponse){
}
```

Explicația detaliată a ce se întâmplă mai sus este următoarea:

- În momentul în care clientul apelează această metoda, serverul este înștiințat că RPC-ul a fost invocat cu datele clientului pentru acest apel, numele metodei și deadline-ul.
- După care, serverul poate să trimită înapoi datele lui inițiale imediat sau poate să aștepte requestul clientului.
- Odată ce primește informațiile, serverul face ceea ce este necesar pentru creearea răspunsului și îl returnează alături de status.
- Dacă statusul este pozitiv, atunci clientul primește răspunsul și aplelul se încheie din partea clientului.

Pentru a vedea utilitatea practică a acestui framework, iată câteva companii care îl folosesc și de ce:

Netflix: *"Așteptăm să vedem multe îmbunătățiri spre dezvoltarea productivității și abiliatea de a dezvolta limbaje non-JVM ca rezultat al adoptării gRPC."*

Square: *"Am descis să ne mutăm spre gRPC pentru suportul liber pe mai multe platforme, pentru performanta demonstrată a protocolului și pentru abilitatea de a-l personaliza și adapta network-ului nostru."*