UE AD FIL A1
UE Services FISE A3 LOGIN
API gRPC

2025-2026

Hélène Coullon

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Table of Contents

1. Les principes de RPC

2. Un peu d'histoire

3. gRPC

Les principes de RPC

RPC

Remote Procedure Call

1. Définition de l'API avec un "Interface Definition Language" (IDL)
2. Appel à distance par un client d'une méthode/procédure fournie par un serveur
   - la *distance* est cachée à l'utilisateur
   - utilisation des méthodes du serveur comme une bibliothèque
   - fonctionnement basé sur un design pattern de Proxy

Proxy et “Stub”

- Les stubs sont les codes générés automatiquement depuis la définition de l’interface avec l’IDL
- Il s’agit de deux proxy : un pour le client et un pour le serveur
  - sérialisation/désérialisation des entrées/sorties
    - aussi appelé marshalling/unmarshalling
  - transformation des données de la mémoire locale pour envoi sur le réseau
  - réception et transformation des données reçues par le réseau
  - gestion de protocoles réseaux utilisés

Diagramme de fonctionnement

Diagramme de fonctionnement

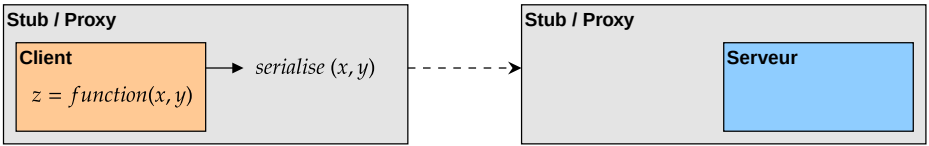


Diagramme de fonctionnement

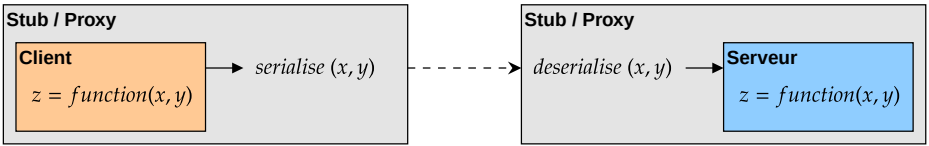


Diagramme de fonctionnement

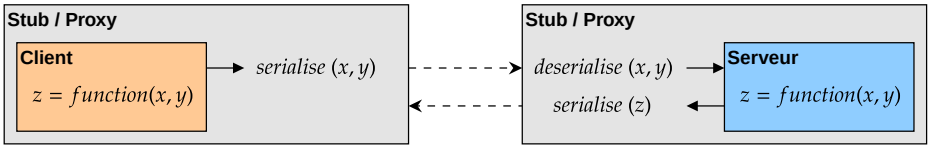
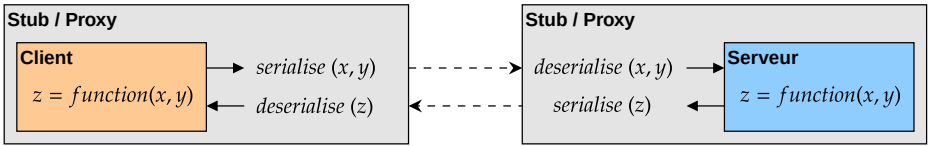


Diagramme de fonctionnement



Interface Definition Language (IDL)

- Langage pour définir une interface, un contrat entre un client et un serveur (entre des services)
- Permet de définir la signature des méthodes/procédures distantes
- Permet de spécifier le type des messages d'entrée et de sortie des méthodes

À quoi sert l'IDL ?

- Documentation
- Les codes des stubs sont générés à partir d'une spécification avec l'IDL

Un peu d'histoire

RPC

Remote Procedure Call : ce type de communication et d'API entre un client et un serveur est apparu avant REST

- Apparition dans les années 1980

CORBA 1991

- Common Object Request Broker Architecture
- Basé sur TCP/IP
- Composants sur des OS différents
- Composants écrits en différents langages
- Composants sur des réseaux différents

SOAP 1998

- Simple Object Access Protocol, Microsoft
- Standard d'échange entre "web services" utilisant le format XML
- Basé sur HTTP (portabilité)
- SOAP est le successeur de XML-RPC
- l'IDL de SOAP est "Web Service Definition Language" (WSDL)
- SOAP est toujours réputé pour sa sécurité et largement utilisé pour les systèmes bancaires !

Un premier pas vers REST avec SOAP !

gRPC

De retour dans les années 90 ?

- Framework OpenSource proposé par Google
- Basé sur HTTP/2.0 pour allier flexibilité et efficacité
- Compatible avec de nombreux langages modernes
  - https://grpc.io/docs/languages/
- Utilisé par des acteurs majeurs
  - Google (bien sûr)
  - NetFlix
  - CoreOS
  - CISCO
  - etc.
- Motivations :
  - API haute Performances
  - Gros payloads et stream processing

IDL de gRPC - protocol buffers

- Spécification des signatures de procédures par service
- Structure des messages d'entrée et de sortie

```
syntax = "proto3";
service Movie {
    rpc GetMovieByID(MovieID) returns (MovieData) {}
}
message MovieID {
    string id = 1;
}
message MovieData {
    string title = 1;
    float rating = 2;
    string director = 3;
    string id = 4;
}

```

Les types de données

- Type scalaires usuels : `float`, `double`, `int` de différentes tailles, string, etc.
- La cardinalité des champs : `singular` (can be optional), `repeated`, `map`
- Les types imbriqués

Les types de données

- Type scalaires usuels : `float`, `double`, `int` de différentes tailles, string, etc.
- La cardinalité des champs : `singular` (can be optional), `repeated`, `map`
- Les types imbriqués

Des exemples

```
message Movies{
    repeated MovieData movies = 1;
}

message MovieData {
    string title = 1;
    float rating = 2;
    string director = 3;
    string id = 4;
}

```

Des exemples

```
message MovieWithActor {
    MovieData movie = 1;
    repeated Actor actors = 2;
}

message Actor{
    string firstname = 1;
    string lastname = 2;
    int32 birthyear = 3;
    repeated string films = 4;
}

```

Des exemples

In proto3

```
map<string, MovieData> movies = 1;

```

Equivalent to

```
message MapMovies {
  string id = 1;
  MovieData movie = 2;
}

repeated MapMovies map_movies = 1;

```

Les types de procédures

Unary RPCs

- Le client envoie une seule requête au serveur
- Le client reçoit une unique réponse
- Appel de fonction normal

```
1 rpc Hello (HelloRequest) returns (HelloReply) {}
2
```

Server streaming RPCs

- Le client envoie une seule requête
- Le client reçoit une séquence de messages (stream) en réponse

```
1 rpc Hello (HelloRequest) returns (stream HelloReply) {}
2
```

Les types de procédures

Client streaming RPCs

- Le clilent envoie une séquence de messages (stream)
- Le client reçoit une unique réponse

```
1 rpc Hello (stream HelloRequest) returns (HelloReply) {}
2
```

Bidirectional streaming RPCs

- Le client envoie un stream
- Le client reçoit un stream

```
1 rpc Hello (stream HelloRequest) returns (stream HelloReply) {}
2
```

Appels synchrones et asynchrones

- Synchrone : l'appel de méthode est bloquant
- Asynchrone : l'appel de méthode est non bloquant, une synchronisation est rendue explicite quand la réponse est nécessaire

Compilation d'un fichier `proto`

Strictement le même fichier `proto` et/ou les même fichiers générés côté client et côté serveur

A chaque modification, il faut recompiler des deux côtés

API internes avec gros transferts de données ou du streaming de données

Stub et Servicer générés

Deux fichiers générés à partir de l'interface `nom.proto`

- `nom_pb2_grpc.py` : contient les stubs client et serveur
- `nom_pb2.py` : gère les types de messages et la sérialisation/désérialisation

Dans `nom_pb2_grpc.py` un Stub et un Servicer

- Stub
  - Il s'agit du Stub client
  - Le code client doit utiliser cette classe pour faire les appels distants
- Servicer
  - Il s'agit du Stub serveur
  - Le code serveur doit surcharger cette classe pour donner l'implémentation des procédures distantes

Comparaison à REST et GraphQL

	Performances	Debug	Flexibilité	Nb fonctions
REST/OpenAPI				
gRPC				
GraphQL				

	gRPC	REST/OpenAPI
Flexibilité	modifier fichier proto re-générer Stubs modifier les codes	modifier requêtes modifier les codes
Dépendances	install gRPC pour tous les langages	installation bibliothèques
Travail	explosion du nb procédures	explosion nb points d'entrée

Exemple d'application mélangeant gRPC et REST

Un exemple d'application fournie par Google qui utilise gRPC pour les communications entre les micro-services internes, avec plusieurs langages à la fois.

https://github.com/GoogleCloudPlatform/microservices-demo

Tutoriel gRPC