gRPC Report

Overview

gRPC (Google Remote Procedure Call) is a high-performance, open-source universal RPC framework initially developed by Google.

It is high performance with a language unspecific approach making it ideal for communication between programs written in different languages. It is written in ProtoBuff and developers can easily store the services and messages in .proto files.

It operates over **HTTP/2**, which supports features like multiplexing, flow control, and full-duplex streaming, enabling four types of communication: unary (single request/response), server streaming, client streaming, and bi-directional streaming.

How gRPC Works

We write the .proto file. Once the .proto file is made we call the protoc function as defined by the language. This protoc function generate 2 file in the directory for the language specified while calling the function. These files have the extension .pb.go . After that we can create the program files that call on to the functions specified in these two files. After that

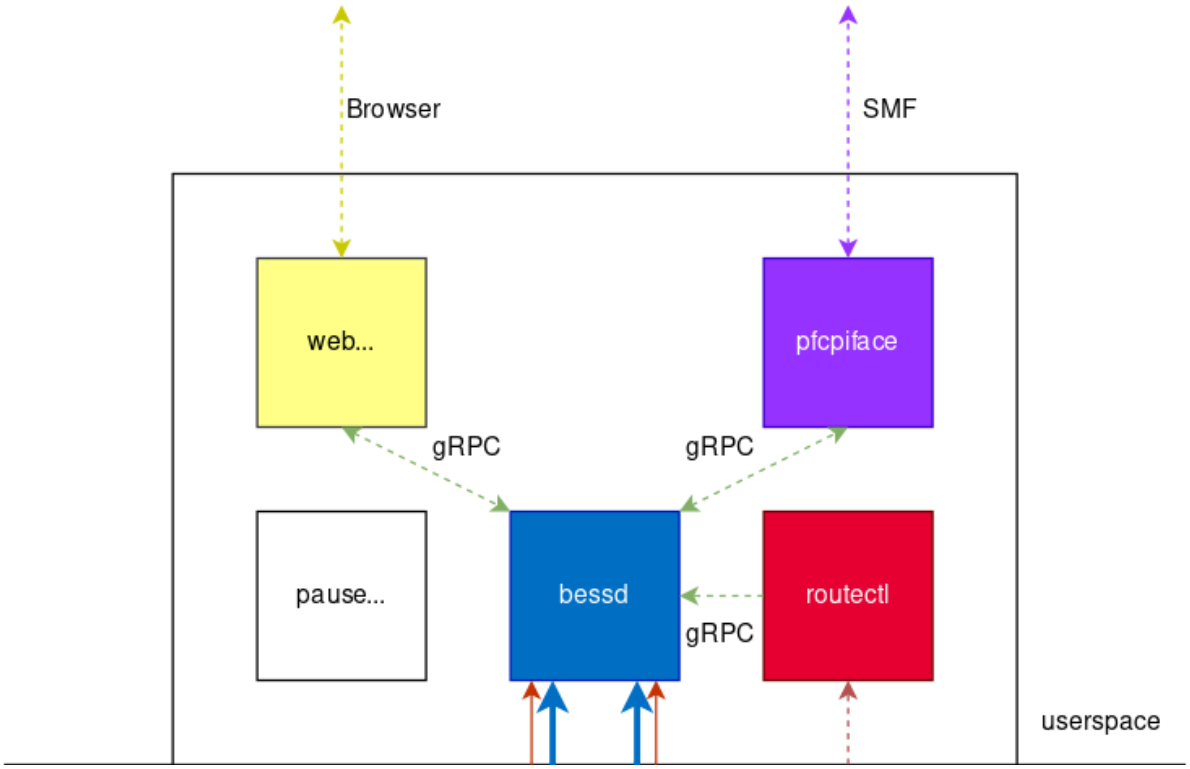
1. **Client calls a method** on the local stub.
2. The stub serializes the request and sends it to the server.
3. The **server deserializes**, processes it, and sends back a serialized response.
4. The stub on the client side gets the response and deserializes it.

Types of Call

Unary	Client send a single request, and the server sends a single response. It is used for standard API calls
Server Streaming	Client send a single request, and the server sends stream of response. It is used for video playback
Client Streaming	Client send multiple request, and the server sends a single response. It is used for file upload
Bi-Directional Streaming	Client send a stream of request, and the server sends a stream of response. It is used for chat apps

Uses in UPF for 5g

pfcp agent ↔ BESS	pfcfiface usually written in Go send the packet rules to BESS written in Python . This communications takes place using gRPC
routectl ↔ BESS	routectl written in Go send the routing instructions to BESS using gRPC
Web ↔ BESS	Web written in Rest displays the pipeline from BESS using gRPC
dBuf ↔ BESS	dBuf written in Go stores and forwards the packets from BESS and stores rules for pfcp agent using gRPC



img via : [omec-project/upf: 4G/5G Mobile Core User Plane]

Advantages

- High performance (uses HTTP/2)
- Bi-directional streaming
- Language and platform neutral
- Strongly typed (via Protobuf)
- Code generation for server and client stubs

Use Cases

- Microservices communication
- Real-time services (chat, video streaming)
- Backend APIs for mobile apps
- 5G architecture (UPF control, telemetry, monitoring)