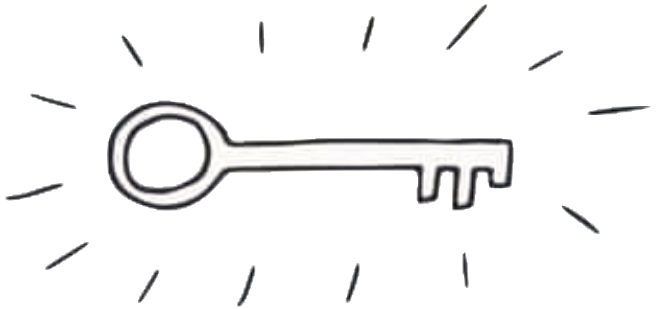
Introduction to Protobuf/GRPC

LOUI Frédéric
WP6T1 – RARE technical leader

STF#21 Virtual meeting

October 26th 2020

www.geant.org



message

Raise technology awareness !

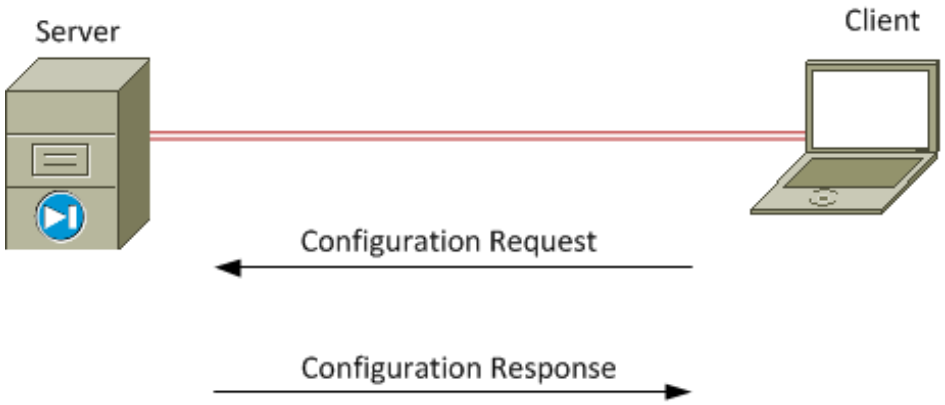
Spark new innovative ideas …

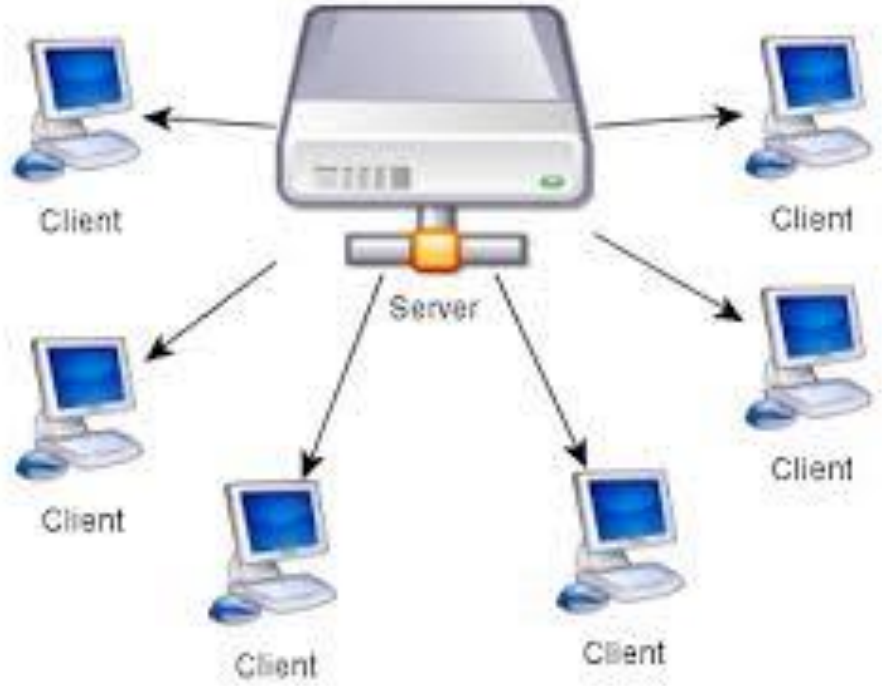
… From you !

NMaaS in a nutshell

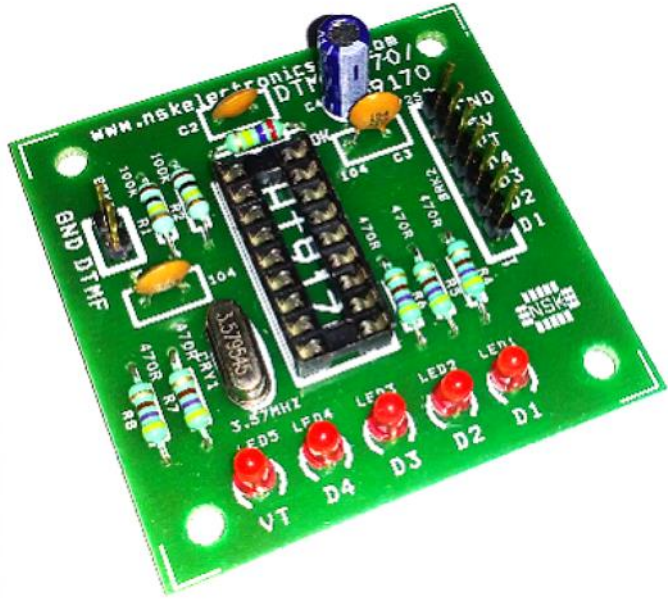
- The problem to solve
  - Client/Server communication
  - A bit of history …
- Protocols
  - Protobuf
  - GRPC
- Let’s write a Protobuf/GRPC client/serveur communication
  - apm.proto
  - Language binding generation
  - APM GRPC Server
  - APM GRPC Client
- Key take away
- Looking ahead

Problem to solve …





A bit of history …



A bit of history …

CSV

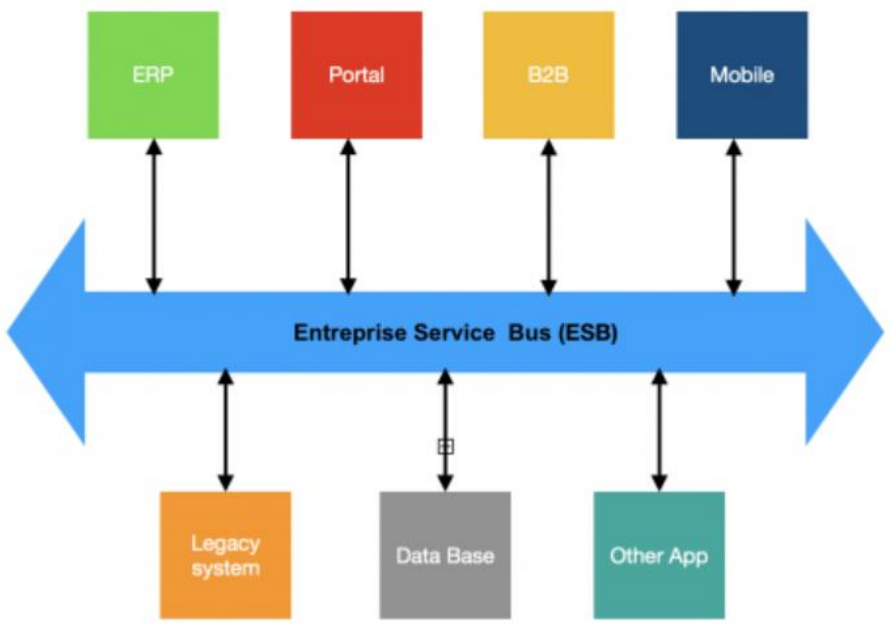
<xml />

SOAP XML

CORBA

Client Server Java RMI

Java Servlets



database

A bit of history …

{JSON}
JavaScript Object Notation

{ REST }

protobuf
Protocol Buffers

gRPC

Protocol buffer

- Binary protocol
- APM.proto

```
syntax = "proto3";

service APM {
    rpc GetAPMFromNRENName(Text) returns (Text) {
    }
}

message Text {
    string str_text = 1;
}
```

gRPC Remote Procedure Call

- Use Protobuf as IDL (Interface Definition Language)
- HTTP2
  - http://www.http2demo.io/
- TLS
- Heterogenous bings
- Streaming
  - Client
  - Server
  - Bidirectional streaming

apm_server.py

```
#!/usr/bin/python3

import grpc
from concurrent import futures
import time

import apm_pb2
import apm_pb2_grpc

import apm

class APMServicer(apm_pb2_grpc.APMServicer):

    def GetAPMFromNRENName(self, request, context):
        print('Receiving from GRPC client %s for NREN: %s'
                % (context.peer(),request.str_text))
        response = apm_pb2.Text()
        response.str_text = apm.get_apm_name(request.str_text)
        return response

server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
```

```
apm_pb2_grpc.add_APMServicer_to_server(
        APMServicer(), server)

print('Starting server. Listening on port 50051.')
server.add_insecure_port('[::]:50051')
server.start()

try:
    while True:
        time.sleep(86400)
except KeyboardInterrupt:
    server.stop(0)
```

apm_client.py

```
#!/usr/bin/python3

import grpc
import time

import apm_pb2
import apm_pb2_grpc

channel = grpc.insecure_channel('localhost:50051')

stub = apm_pb2_grpc.APMStub(channel)

NREN='RENATER'
organization = apm_pb2.Text(str_text=NREN)
response = stub.GetAPMFromNRENName(organization)
print("Request from GRPC client ---> [%s]"
        % (NREN))
print("Answer from GRPC server ---> [%s]"
        % (response.str_text))

time.sleep(2)
```

```
NREN='KIFU'
organization = apm_pb2.Text(str_text=NREN)
response = stub.GetAPMFromNRENName(organization)
print("Request from GRPC client ---> [%s]"
        % (NREN))
print("Answer from GRPC server ---> [%s]"
        % (response.str_text))

time.sleep(2)

NREN='DFN'
organization = apm_pb2.Text(str_text=NREN)
response = stub.GetAPMFromNRENName(organization)
print("Request from GRPC client ---> [%s]"
        % (NREN))
print("Answer from GRPC server ---> [%s]"
        % (response.str_text))
```

Useful links

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

https://grpc.io/

P4Runtime

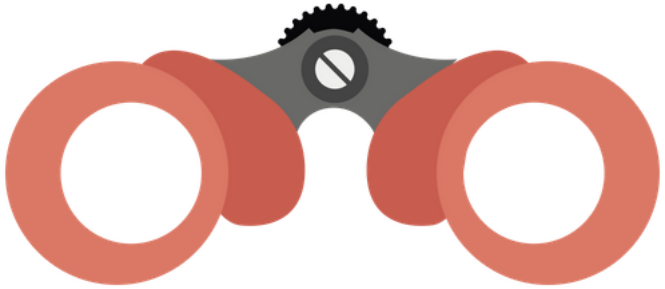
https://p4.org/p4runtime/spec/v1.2.0/P4Runtime-Spec.html

https://github.com/frederic-loui/grpc-apm-example

Key take-away

- Protobuf
  - IDL used to define client/server communication
- gRPC
  - "Enveloppe" used to transport message
  - Fast as relying on binary and HTTP2
  - Secured by TLS
- You can now read & understand
  - Existing Client/Server communication by reading proto files
  - Understand vendor mentioning gRPC in their presentation
- You can create your own client/server communication
  - Write proto file
  - Compile proto file in order to generate your favorite language binding
    - (Not all language have gRPC bindings)
  - Write your own Client/Server

Looking ahead



P4Runtime

BfRuntime

OPENCONFIG

kubernetes

NREN gRPC PROXY ?

Thank you

Any questions?

www.geant.org

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 856726 (GN4-3).