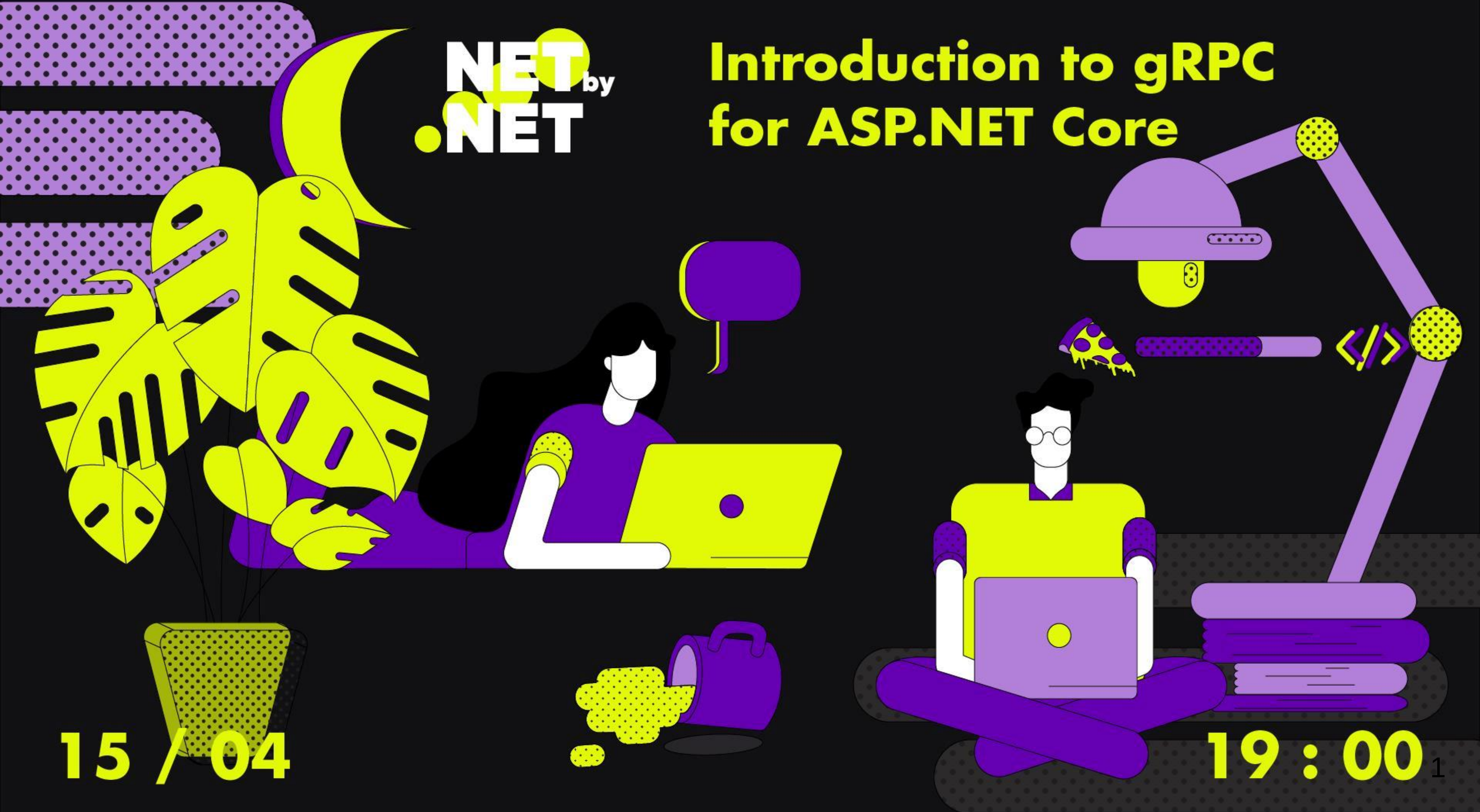
Introduction to gRPC for ASP.NET Core

NET by .NET



15 / 04

19 : 00

AGENDA

19:00	~~Coffee~~* & Chat
19:10	Intro Talk
19:15	Tigran's Talk
20:00	Questions

*Come with your own coffee ☺

Tigran Topchyan, PhD

Solution Architect, Director – TopSoft LLC

tigran@topsoft.am
www.linkedin.com/in/tigran-topchyan
https://t.me/tigertop

- Enterprise developer since 2009
- Certified Microsoft Azure Architect
- Electronics hobbyist
- Lego fan

gRPC

NET by .NET

Introduction to gRPC for ASP.NET Core

Outline

- What is gRPC?
- gRPC vs REST
- gRPC in .NET Core 3
- Demos

HOW DO WE MAKE **COMPUTERS** TALK TO EACH OTHER?

NET by .NET

HOW DO WE MAKE APPS THAT TALK TO EACH OTHER?

HOW DO WE BUILD **WEB APIS**?

My favorite SOAP service:

http://api.cba.am/exchangerates.asmx

REST

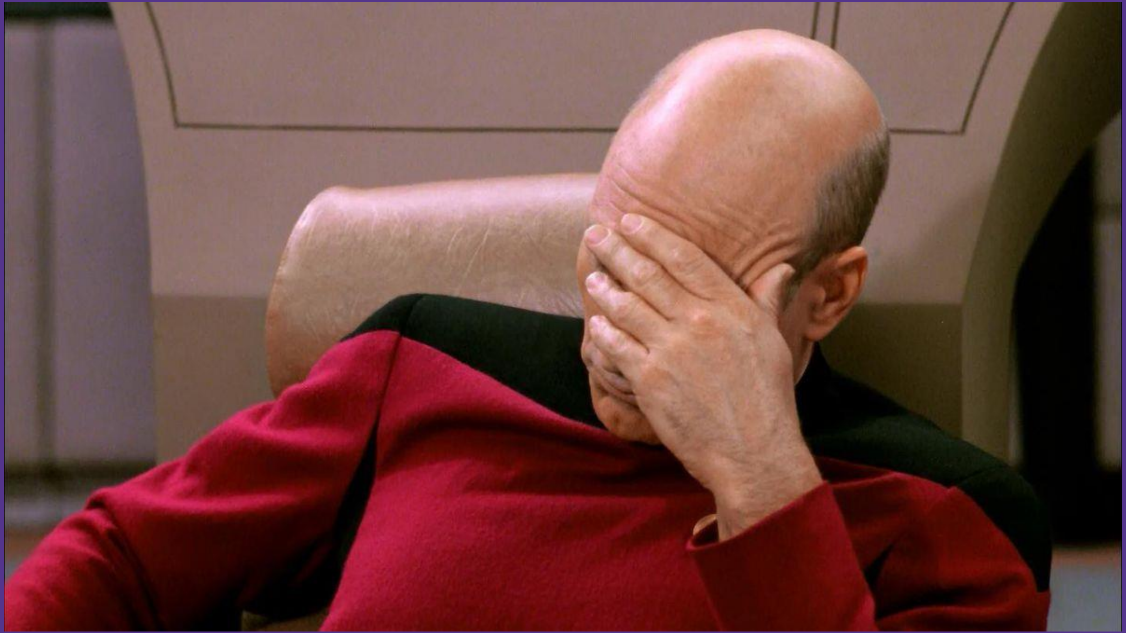
HTTP + JSON

(REST)

HTTP/REST IS GREAT

- Easy to understand (text)
- A lot of infrastructure uses it
- Great tools and development and debug
- Loose coupling between clients/server makes changes (relatively) easy
- High quality HTTP implementation In every language

And what about client libs?



HTTP/REST IS **NOT** GREAT

- No formal (machine-readable) API contract
- Streaming is difficult.
- Bi-directional isn't possible in some languages
- Operations are difficult to model (e.g. restart machine). Emphasizes HTTP
- Inefficient (textual representation aren't optimal for some network)
- Internal API are not always RESTful

gRPC to the rescue

NET by .NET

gRPC = gRPC Remote Procedure Call

Latest technologies:

- HTTP/2
- Protobuf

High performance

Cross platform

‘g’ is for: https://bit.ly/2KaWpfT

gRPC = http2 + protobuf Protocol Buffers

Protobuf (aka Protocol Buffers)

IDL (interface definition language)

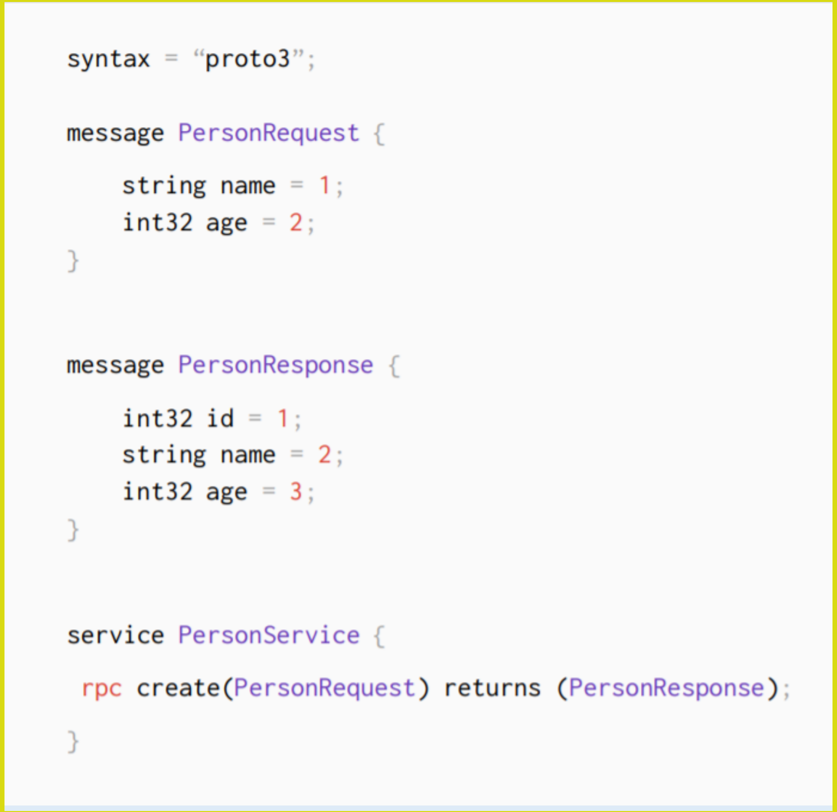
- Describe once and generate interfaces for any language

Service model

- Service method and structure of the request and the response

Wire format

- Binary format for network transmission



```
syntax = "proto3";

message PersonRequest {
    string name = 1;
    int32 age = 2;
}


message PersonResponse {
    int32 id = 1;
    string name = 2;
    int32 age = 3;
}


service PersonService {
 rpc create(PersonRequest) returns (PersonResponse);
}
```

DEMO

Create gRPC service using template

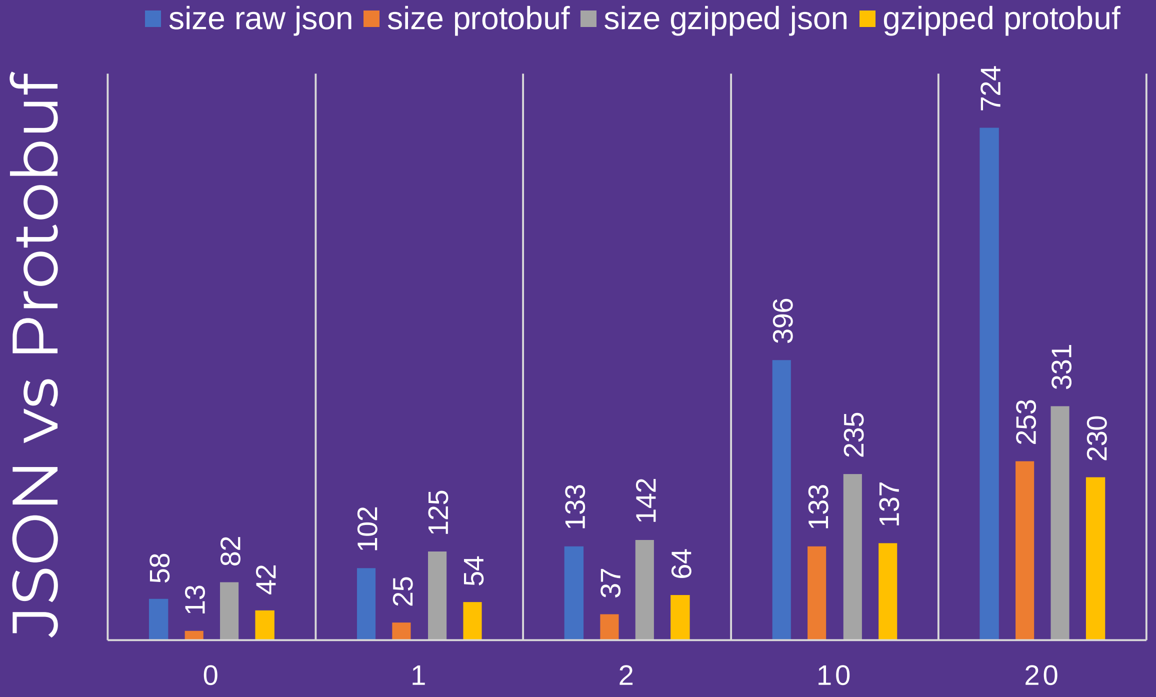
Create gRPC client

Call service

REST vs gRPC

REST		gRPC
Resource/Content first		Contract first - protobuf
Human Readable content	VS	Human Readable contract
Test Serialization – JSON		Binary Serialization
Heavily leverages HTTP		**Hides transport layer – HTTP/2**

Key features - Performance

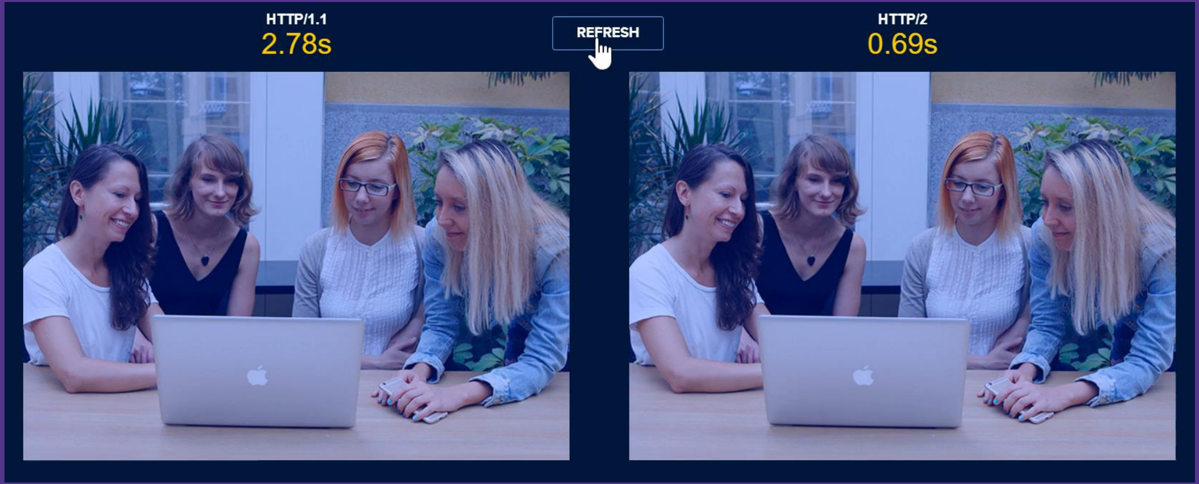


https://nilsmagnus.github.io/post/proto-json-sizes/

Key features - Performance

HTTP/2 multiplexing

- Multiple calls via a TCP connection
- Avoid head-of-line blocking*



http://www.http2demo.io/

https://http1.golang.org/gophertiles

~3.4 times faster
in this particular test

Key features – Code Gen.

All gRPC libraries have first-class code generation support

```
syntax = "proto3";
option csharp_namespace = "GrpcService1";
package greet;

service Greeter {
rpc SayHello (HelloRequest) returns
(HelloReply);
}

message HelloRequest {
  string name = 1;
}

message HelloReply {
  string message = 1;
}
```

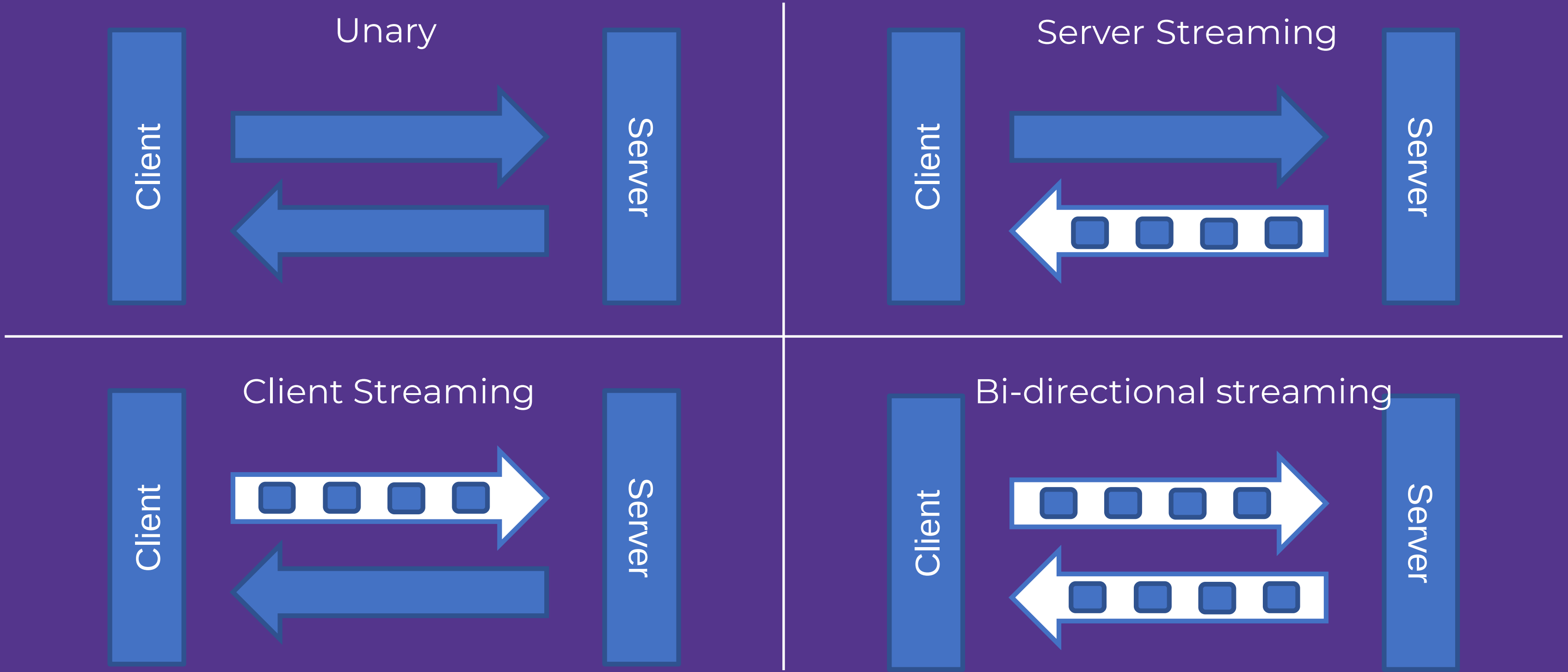
```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>netcoreapp3.1</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <Protobuf Include="Protos\greet.proto" GrpcServices="Server" />
  </ItemGroup>

  <ItemGroup>
    <PackageReference Include="Grpc.AspNetCore" Version="2.28.0" />
    <PackageReference Include="Grpc.AspNetCore.Server" Version="2.28.0"/>
    <PackageReference Include="Grpc.AspNetCore.Web" Version="2.28.0-pre2"
/>
</ItemGroup>

</Project>
```

Key features – Multiple langs.

python™
Java
node js
Microsoft® .NET
GO
Perl
Swift
R
Dart
php

Key features – Streaming

gRPC uses HTTP/2 to enable streaming

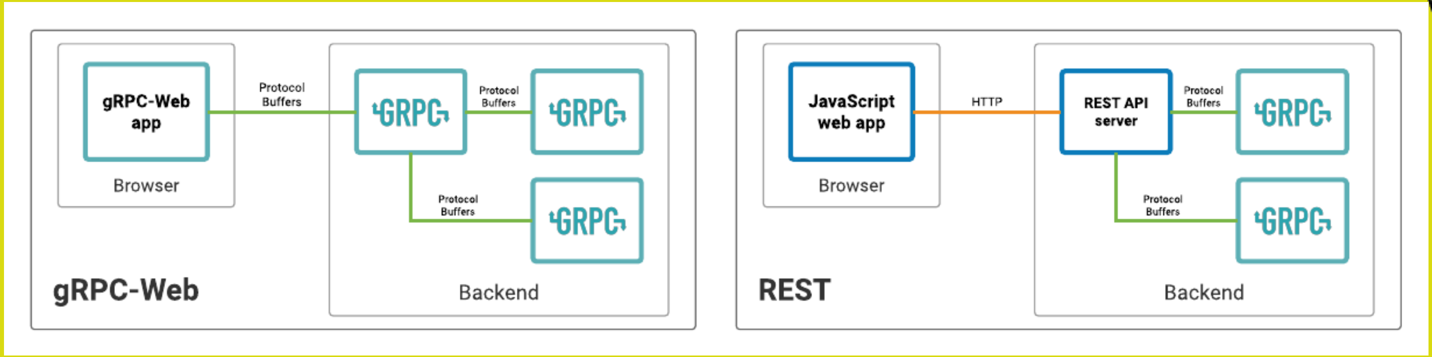


DEMO

Update proto with server streaming call
Implement on server
Call from client

Disadvantages – Limited browser support

- Browsers have great HTTP/2 support
- Browser JavaScript APIs lack HTTP/2 support
- gRPC-web provides limited support for calling gRPC



https://habr.com/en/company/microsoft/blog/486450/

Disadvantages – Not human readable

- HTTP/2 and protobuf are binary protocols
- Additional tools required to debug calls

BloomRPC: https://github.com/uw-labs/bloomrpc

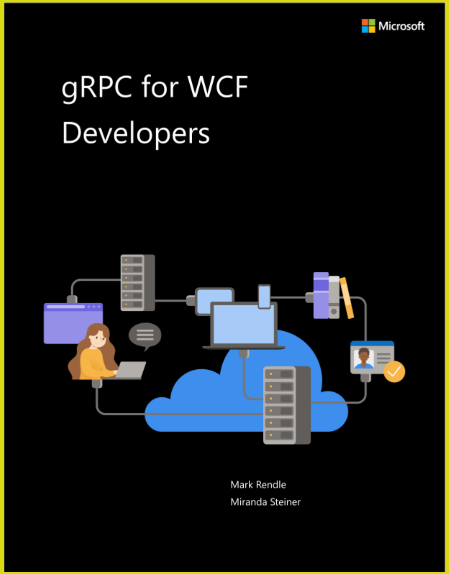
https://www.wireshark.org/

References

gRPC docs - https://docs.microsoft.com/aspnet/core/grpc

- gRPC with ASP.NET Core authentication
- Logging and diagnostics
- HTTPClientFactory Integration

gRPC for WCF Developers https://docs.microsoft.com/dotnet/architecture



Questions?

NET by .NET

Thank you

Contact Details

Telegram

Linkedin

Facebook

Feedback

Want to be a speaker?

NET by .NET