REALTIME MICROSERVICES

GRPC WITH .NET

OVERVIEW

- About me
- HTTP/2 in one chart
- Protobuf fundamentals
- GRPC = HTTP/2 plus Protobuf
- Nuget Packages and Code Generation
- Live-Coding examples
- Ressourcen / Outlook

ABOUT ME

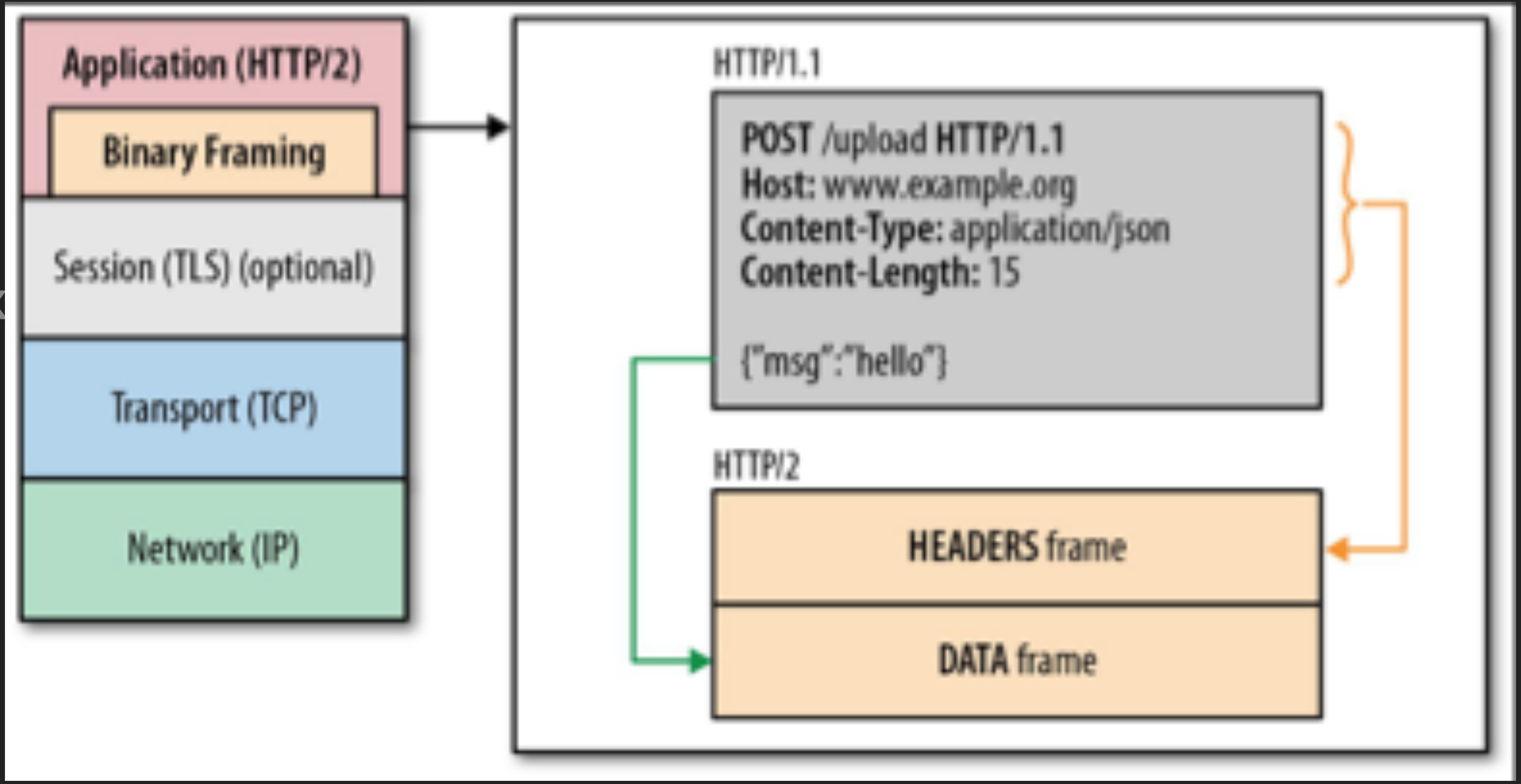
- Management information system studies
- 5 years Accenture
- 5 years ISV for exchange trading
- 10 Jahre CTO at a prop trading company which is a member of Eurex
- daily transformation of 250 - 400 Mio. price messages into orders and trades

GRPC OVERVIEW

- Public successor of Stubby at Google for Microservices
- A RPC/Remote Procedure Call library and framework
- GRPC = HTTP/2 + Protocol Buffer

HTTP/2 IN ONE SLIDE

- One TCP Verbindung
- Request -> Stream
  - Streams with Multiplex
  - Streams with Priority
- Binary core layer
  - Periodically
  - Flow control
  - Server push
- Header compression



```
POST /upload HTTP/1.1
Host: www.example.org
Content-Type: application/json
Content-Length: 15

{"msg":"hello"}
```

PROTOBUF

- Structure representation of data
- Googles general description of data
    - 48.000 messages
    - 12000 Proto files
- Evolutionary development (currently version 3)

WHY PROTOBUF

- Efficiency in size
- Clear compatibility rules
- Idiomatic - one file for multiple languages(C++, C#, Java, Java Script, Objective-C, Ruby, Python, Go)
- Strongly typed for performance and safety

MESSAGE FORMAT (PROTO 3)

- Explicitly numbered fields Felder
- Typed
- Hierarchical
- Arrays
- Extensible through sub structures

```
syntax =„proto3"

message Person {

string Name =1;

int32 Id = 2;

}
```

PROTOBUF SCALAR VALUE TYPES

- double
- float
- int32/int64
- bool
- string
- bytes

PROTOBUF ENUMS

- Enums similiar to C#

```
enum Corpus {
  UNIVERSAL = 0;
  WEB = 1;
  IMAGES = 2;
 }
 Corpus corpus = 4;
```

PROTOBUF COMPOSITION

- Structs may contain other structs
- .proto may reference each other

```
message SearchResponse {
  repeated Result results = 1;
}
message Result {
  string url = 1;
  string title = 2;
  repeated string snippets = 3;
}
import "myproject/other_protos.proto";
```

PROTOCOL BUFFER COMPILER

https://github.com/google/protobuf

GRPC

- Service Definition via .proto files
- Code generation in 10 languages- in .NET via Nuget package using a C-core
- Using Http/2
- Point to point connection
- Streaming / Uni- and Bidirectional
- Authentification via SSL/TLS and OAUTH2

GRPC VS SIGNALR

- SignalR works via websockets
- SignalR ist a pure .NET architecture
- GRPC has advantages if you want to be language-anostic Vorteile, wenn man sprach-unabhängig sein will
- GRPC is unfortunately not supported straight in the browser => there are bride solutions in development

PROJECT STRUCTORE OF GRPC

- ServiceDefinition Projekt
  - .Proto with definitions
  - Generated Code
  - Potentially add some basic logic in partial classes
- Client/Service Consumer project
- Server project

```
syntax = "proto3";

package helloworld;

service Greeter {

 rpc SayHello (HelloRequest)
returns (HelloReply) {} }

message HelloRequest { string
name = 1;}

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

MY EXPERIENCE

- Server API is amodernes async/await .NET API
- Lean and mean implementation
- No decimals
- No nullables

LIVE CODING STRUCTURE

- 3 Projects will be created
- Nuget packages for Grpc.Core, Grpc.Tools and Google.Protobuf need to be added
- .proto file as first step
- code generation
- Add reference to generated code in ServiceDefinition project
- In client and server project add a reference to service definition
- Implement interfaces
- Present using command line

GRPC AND WPF

- In Client Code do not forget „ContinueWith“
- asynchrone Read methods should be called in the WindowLoaded Event, if you want a continous push during the full app-lifetime and await or Task.AwaitAll should be used, to avoid deadlocks

- The streamWriter inside the server is bound to the calling async task
- Blocking Collection is not adeqaute because it is only synchronous
- TPL Data Flow provides BufferBlock =:> FIFO Queue with synchronous und asynchronous methods for Post/SendAsync and Receive/ReceiveAsync

RESSOURCEN

- http://www.grpc.io/
- https://github.com/grpc/grpc
  - Important: One universal project for all languages
  - General and .NET specific directories