



# РАЗПРЕДЕЛЕНИ ПРИЛОЖЕНИЯ

Павел Кюркчиев  
Ас. към ПУ „Паисий Хилендарски“  
@rkyurkchiev

GRPC



# Какво е Remote procedure call (RPC)?

- RPC е модел за мрежово програмиране или техника за комуникация между процеси, използвана за комуникация от точка до точка между софтуерните приложения.
- RPC е протокол, който една програма може да използва, за да поиска услуга от друга програма, намираща се на отдалечен компютър, без да е необходимо да разбира подробностите на свързващата ги мрежата.

# Модел на изпълнение

- RPC използва client-server модела.
- Начинът, по който работи RPC е: подател или клиент създава заявка под формата на процедура, функция или повикване към отдалечен сървър, сървъра приема заявката и започва обработката и. Когато отдалеченият сървър обработи заявката изпраща отговор към клиента и той продължава своя процес на работа.

- Използването на „lightweight processes“ или „threads“, които споделят едно и също адресно пространство, позволява няколко RPC заявка да се изпълняват едновременно.

# Какво е gRPC?

- Представява рамка за разработка на RPC системи. Рамката първоначално е изработена от google за техни вътрешни нужди. Сега проекта е с отводен код.

# ОСНОВНИ КОМПОНЕНТИ

- HTTP/2
- Protobuf serialization (proto3)
- Interface Definition Language (IDL)

# Начини на комуникация

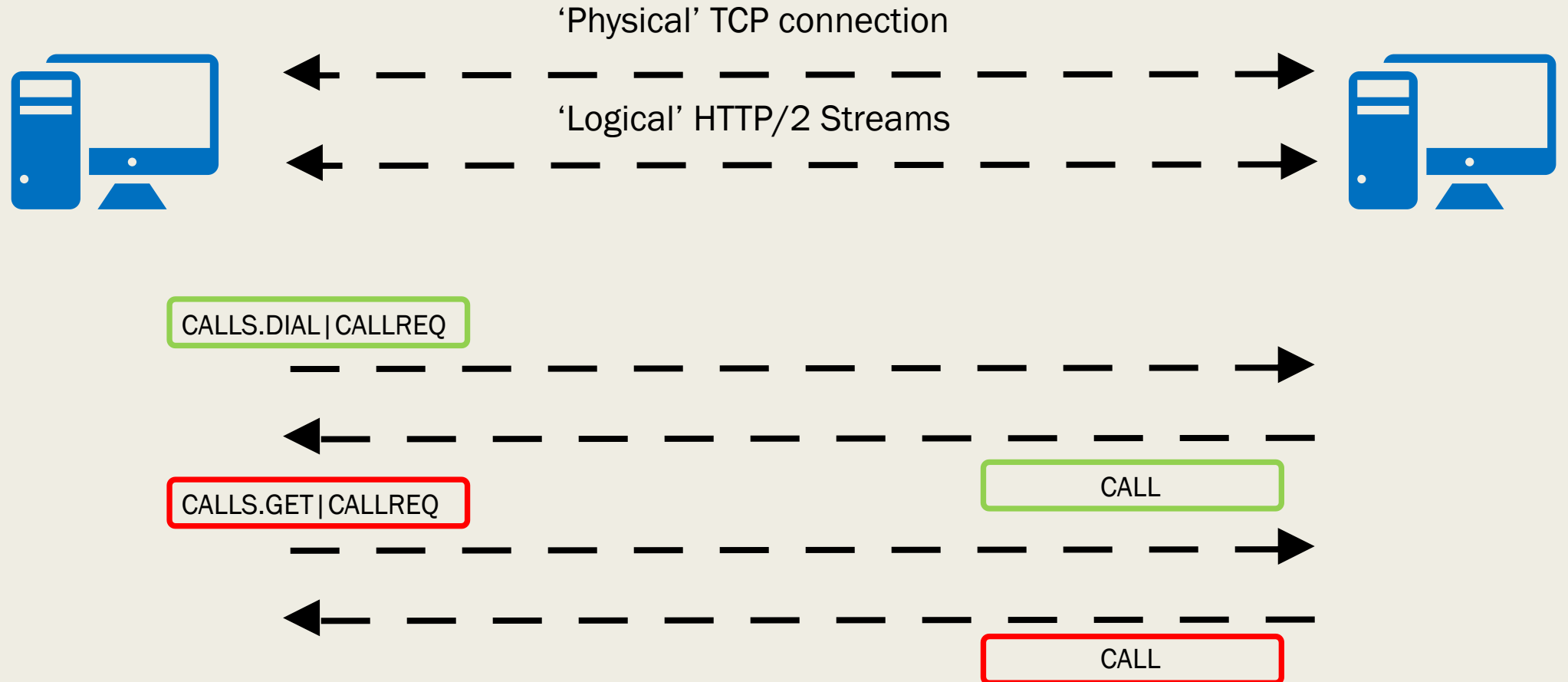
- Unary RPC
- Server streaming RPC
- Client streaming RPC
- Bidirectional streaming RPC
  - Deadlines/Timeouts



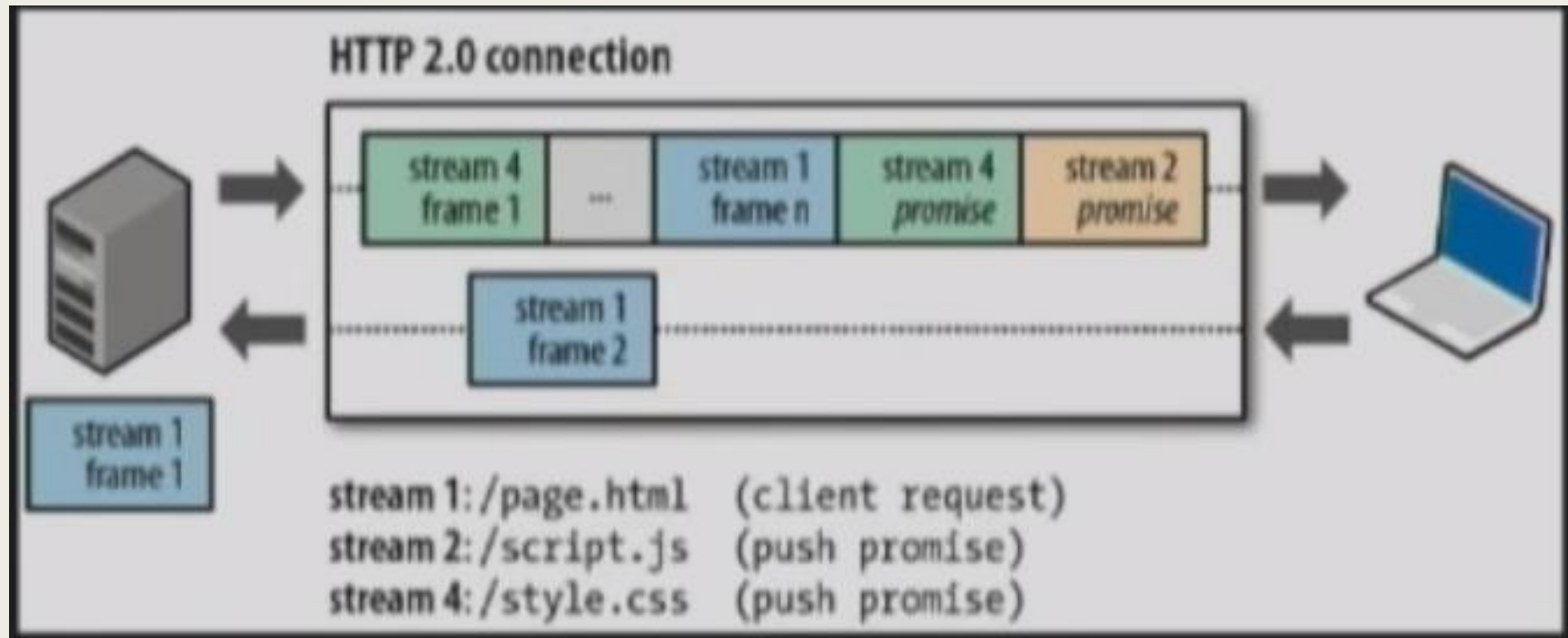
# Легенда

- **Unary** - клиента изпраща едно запитване до сървъра и получава един отговор
- **Stream** – клиента изпраща запитване до сървъра и получава стрeam за четене (последователност от съобщения)

# GRPC: on the wire 1



# GRPC: on the wire 2



# Верификация

- gRPC поддържа TLS и токън базирана верификация.
- TLS е криптографски протокол за предаване на информация по компютърната мрежата.

# Имплементация

- 3 високо производителни събитийно базирани имплементации
  - C
    - Ruby, Python, node.js, PHP, C#, Objective-C, C++ всички са “C” базирани
  - *Java*
    - Netty + boringSSL via JNI
  - Go
    - Чиста имплементация на Go използваща stdlib crypto/tls пакет

# RESTful vs gRPC

- Ресурс базиран
  - IDL опционален
  - Синхронен по подразбиране
  - Unary
  - Перфектен за serverless системи
- API базиран
  - IDL насочен
  - Асинхронен по природа
  - Streaming или Unary
  - Най – важна е производителността

# RESTful vs gRPC

- Използвай: когато трябва бързо да се сглоби нещо или ако трябва да е лесно разбираемо от крайния потребител.
- Не използвай: когато се изисква някакъв вид проверка на типа данни или трябва да се намали размера на предавана информация.
- Използвай: когато има комуникация между `microservices` намиращи се в един клъстер или се изисква бързодействие.
- Не използвай: когато трябва да се предава информация между браузър и `back-end` услуги.

# DEMO GRPC

examples/GrpcGreeter



ВЪПРОСИ ?

