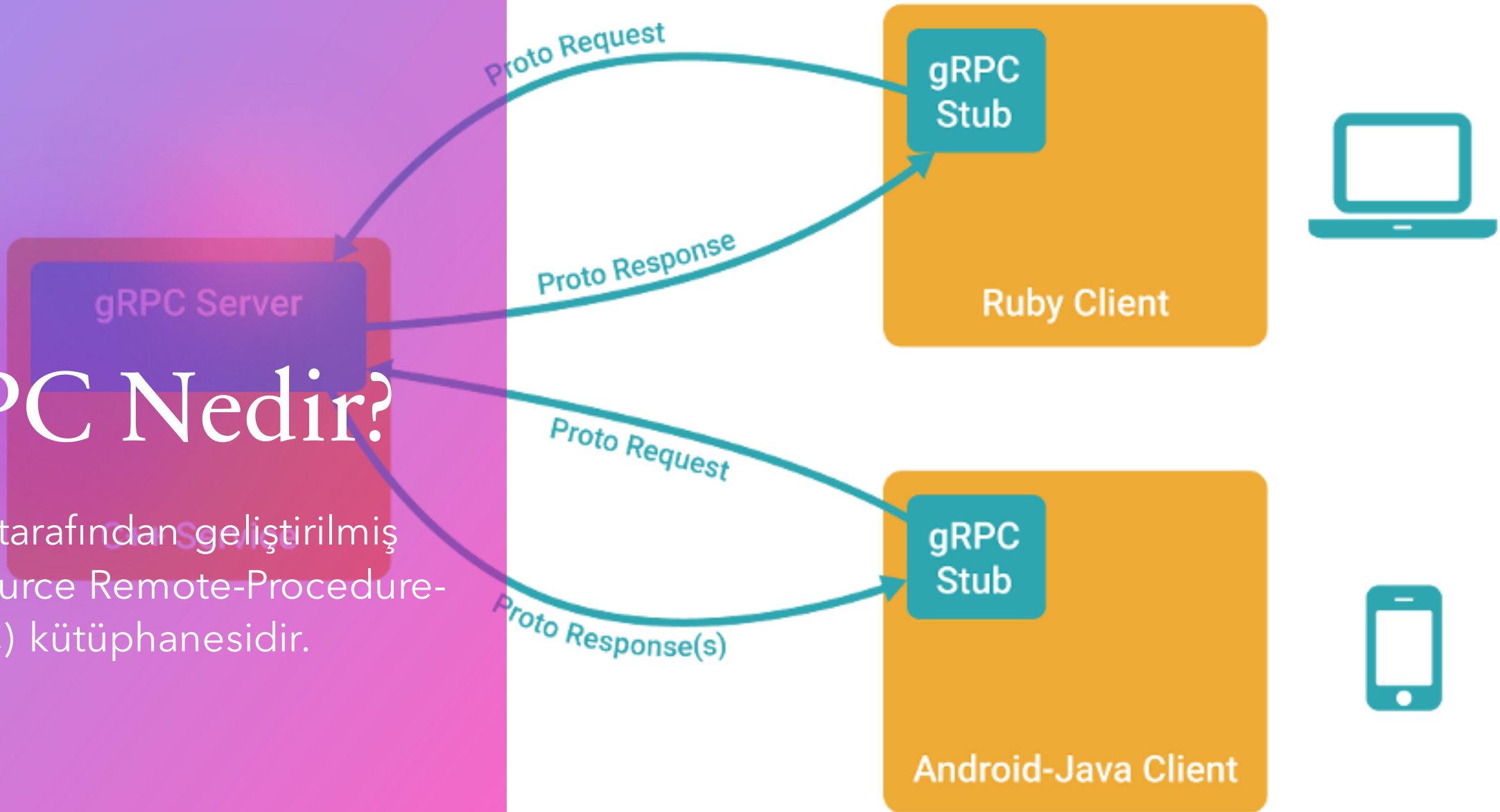


gRPC



gRPC Nedir?

Google tarafından geliştirilmiş open source Remote-Procedure-Call(RPC) kütüphanesidir.



g Harfi Ne Anlama Geliyor?

- Genellikle üreticisi Google gibi düşünülebilmektedir lakin değildir!
- Google tarafından çıkarılan her bir sürüm için baş harfi 'g' ile başlayan ayrı anlamı ifade etmektedir. Bu anlamları aşağıdaki adresten daha rahat görebilirsiniz.
- https://grpc.github.io/grpc/core/md_doc_g_stands_for.html

'g' stands for something different every gRPC release

- 1.0 'g' stands for 'gRPC'
- 1.1 'g' stands for 'good'
- 1.2 'g' stands for 'green'
- 1.3 'g' stands for 'gentle'
- 1.4 'g' stands for 'gregarious'
- 1.6 'g' stands for 'garcia'
- 1.7 'g' stands for 'gambit'
- 1.8 'g' stands for 'generous'
- 1.9 'g' stands for 'glossy'
- 1.10 'g' stands for 'glamorous'
- 1.11 'g' stands for 'gorgeous'
- 1.12 'g' stands for 'glorious'
- 1.13 'g' stands for 'gloriosa'

RPC Ne Demektir?

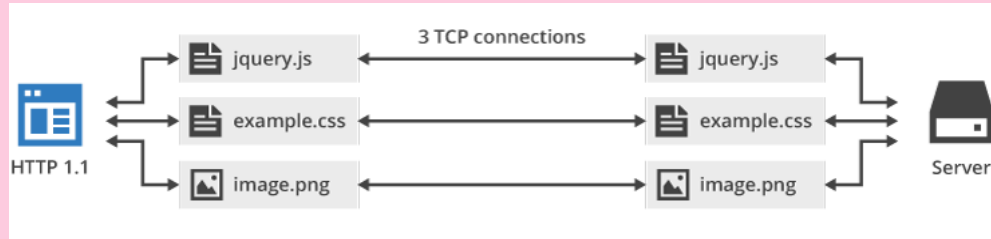
- Remote-Procedure-Call(RPC) 'Uzak Yordam Çağırısı'dır.
- RPC; uzak sunucudaki metotları sanki kendi ortamının birer parçasıymış gibi çağırabilen sistemdir.



- gRPC; transport/iletiřim/veri iletimi için Http/2 protokolünü kullanmaktadır.
- Dolayısıyla gRPC'nin faydalarını anlayabilmek için öncelikle Http/2'nin Http/1'e nazaran getirilerini anlamak gerekmektedir.

Http/1

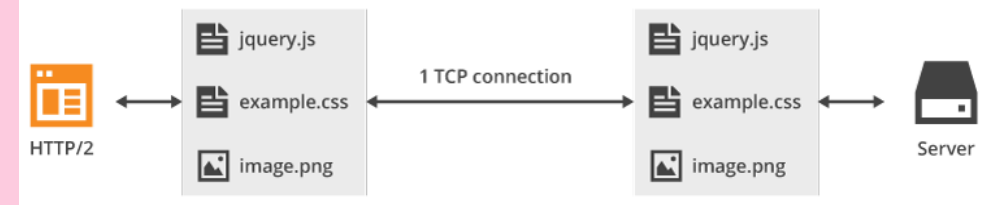
Her bir statik dosya için(.css, .js, .png vs.) ayrı istek göndermektedir. Bu durum ise yük ve maliyeti arttıracacağından dolayı ektradan bekleme süresinin artmasına sebep olmaktadır.



Http/2

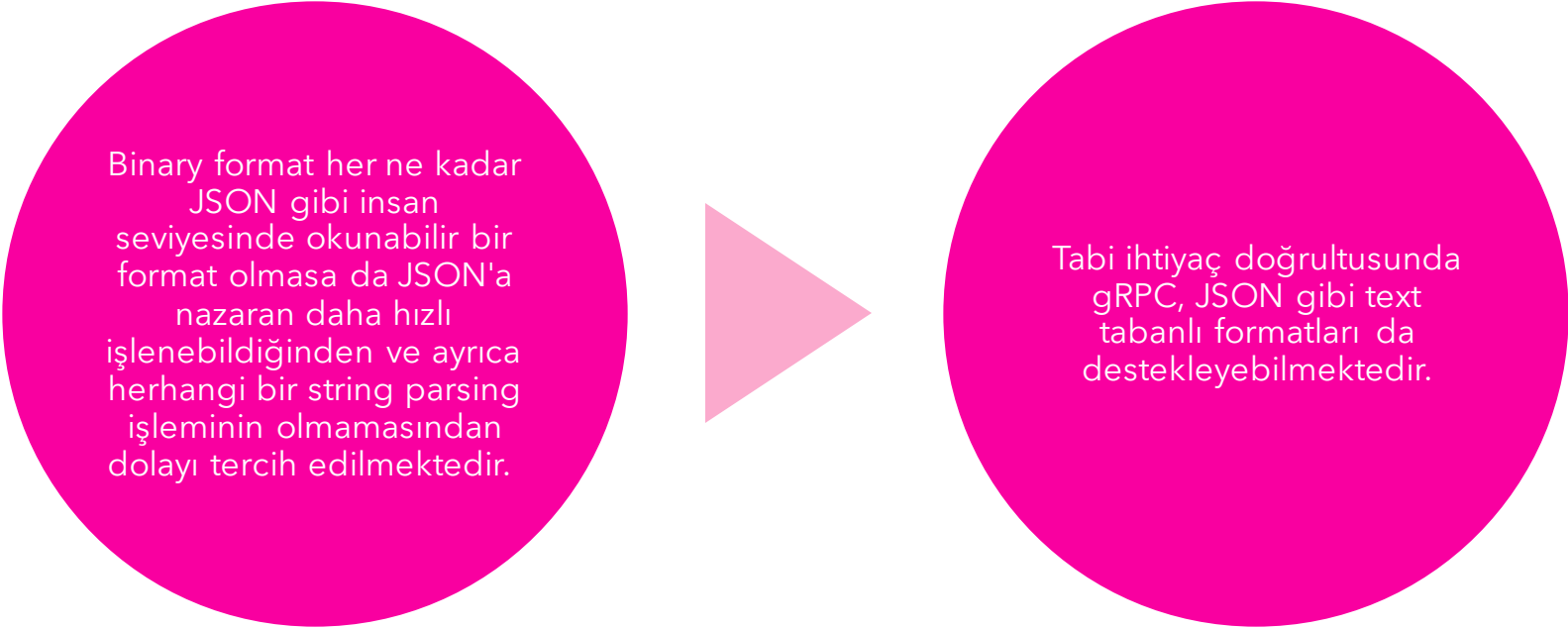
Dosyalar için tüm istekler tek seferde toplu olarak yapılabilir. Böylece açılış hızı artmakta, süresi düşürülmektedir..

Multiplexing



Bu duruma **Multiplexing** denmektedir. Teknik olarak, tek bir TCP bağlantısı üzerinde birden çok ve paralel request ve response yeteneği olarak yorumlanabilir.

Http/1	Http/2
Metin tabanlı(text based) protokoldür.	Client ile server arasındaki iletişim binary formatta ki küçük frame'lere ayrılmaktadır. Bu duruma Binary Protokol denmektedir.
Her request'te sıkıştırılmamış vaziyette header gönderilir.	Her request'te header'lar HPACK ile sıkıştırılarak gönderilmektedir. Bu duruma Header Compression denmektedir.
Bir request'e bir response döner.	Bir request'e karşılık birden fazla response alınabilir. Server Push



Binary format her ne kadar JSON gibi insan seviyesinde okunabilir bir format olmasa da JSON'a nazaran daha hızlı işlenebildiğinden ve ayrıca herhangi bir string parsing işleminin olmamasından dolayı tercih edilmektedir.

Tabi ihtiyaç doğrultusunda gRPC, JSON gibi text tabanlı formatları da destekleyebilmektedir.

Neden gRPC Kullanmalıyız?

- Günümüzde yoğun bir şekilde uygulama geliştirme yaklaşımı olarak benimsenen microservice yapılanması, sistemin gelişim sürecini her ne kadar dinamize etsede bir o kadar da servisler arası iletişim hızını monolithic yaklaşıma binaen oldukça düşürmektedir.
- Bu durum ise bizlere klasik Restful mimarisinden ziyade daha hızlı bir altyapı ihtiyacı hissettirmekte ve mümkün mertebe iletişim hızımızı güçlendirecek farklı teknolojilere yönlendirmektedir.
- Nihayetinde Restful service'ler çoğunlukla external(dış) bir client tarafından consume edilmek için daha uygun bir fitrata sahiptir ve text-based messaging'e dayandığı için internal(iç) service iletişimine pekte yatkın bir seçenek değildir. Çünkü burada hız önplandadır.
- Haliyle dış dünya tarafından tüketilmeyen internal projelerde Restful'dan ziyade binary-based messaging'i benimseyen gRPC'yi kullanmamız, bizleri iletişim hızı açısından monolithic'e en yakın seviyeye çıkarmaktadır diyebiliriz.



- gRPC sayesinde Restfull'un etkisini yitirdiğini düşünmek yanlış olacaktır. En nihayetinde external client'lar tarafından consume edilen uygulamalarda Restfull mimarisi hala en kolay, kullanışlı ve efektif çözümdür.

IoT(Internet of Things) çağında cihazlar arası iletişim içinde gRPC iyi bir seçenek diyebiliriz.

gRPC Avantajları Nelerdir?

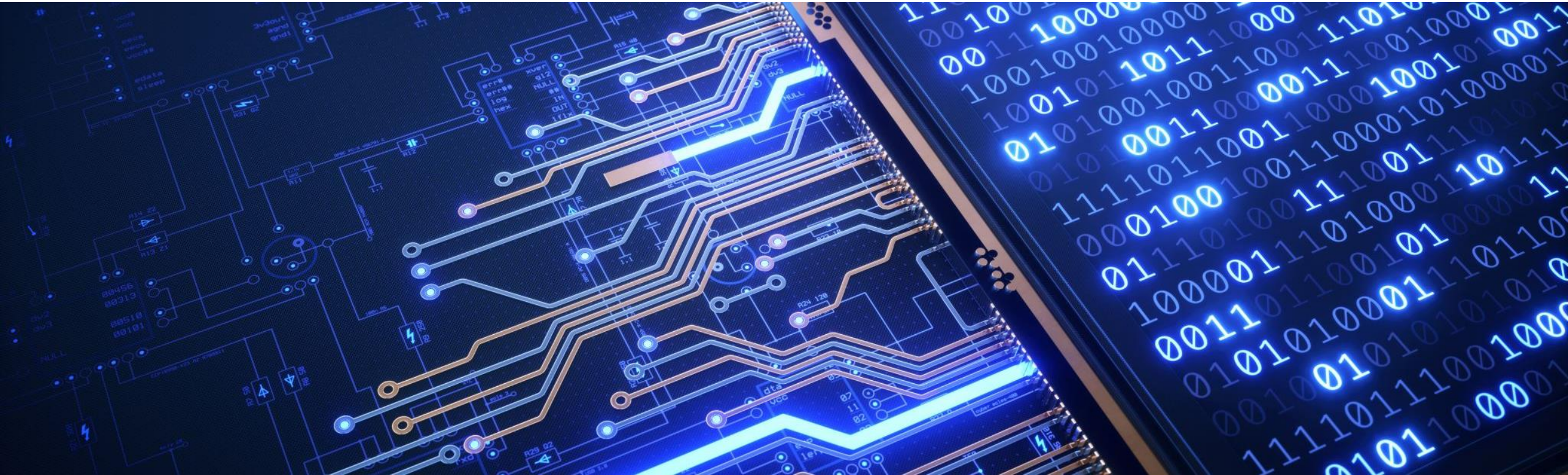
- Http/1'e nazaran Http/2'yi kullanmaktadır. Böylece Http/2 stream desteği verebilmektedir.
- TCP socket haberleşmesi sağlayabilmektedir.
- Http/2 sayesinde binary serialization kullanmaktadır. Böylece text-based mesajlaşmaya nazaran oldukça hızlıdır.
- Yapılan optimizasyonlar neticesinde Http/1 kullanan Restfull servislere nazaran 2.5 kat daha hızlı olduğu tespit edilmiştir. Http/1 ile Http/2 arasındaki hız farkını daha net görebilmek için http2demo.io adresindeki demoyu inceleyebilirsiniz.

gRPC Avantajları Nelerdir?

- Aynı bağlantı üzerinden birden fazla paralel request desteği sağlanmaktadır. Http/1'de ise bir request'e nazaran bir response söz konusudur.
- Client ile server arasında çift yönlü iletişim vardır.
- Moderndir.
- Birlikte birçok niteliğinden dolayı yüksek performanslıdır.
- Default olarak Protocol Buffers kullanarak dilden bağımsız olacak şekilde birçok uygulama tarafından kullanılabilir.

Protocol Buffers(Protobuf) Nedir?

- Google'ın geliştirdiği ve hala geliştirmekte olduğu bir binary serialization protokolüdür.
- Özünde bir Arayüz Tanımlama Dili/Interface Definition Language(IDL)'dir.
- Kullanılan platform ve programlama dili farkını gözetmeksizin, client ve server arasında haberleşmeyi sağlayabilmek için IDL compiler sayesinde her iki tarafa da(client ve server) 'stub' ismi verilen gerekli arayüzlerin oluşturulmasını sağlayan bir dildir.
- İçerik olarak gRPC servis tanımlarını ve import edilen paket tanımlarını ve iletişim sürecinde kullanılacak olan mesaj tanımlarını tutmaktadır.



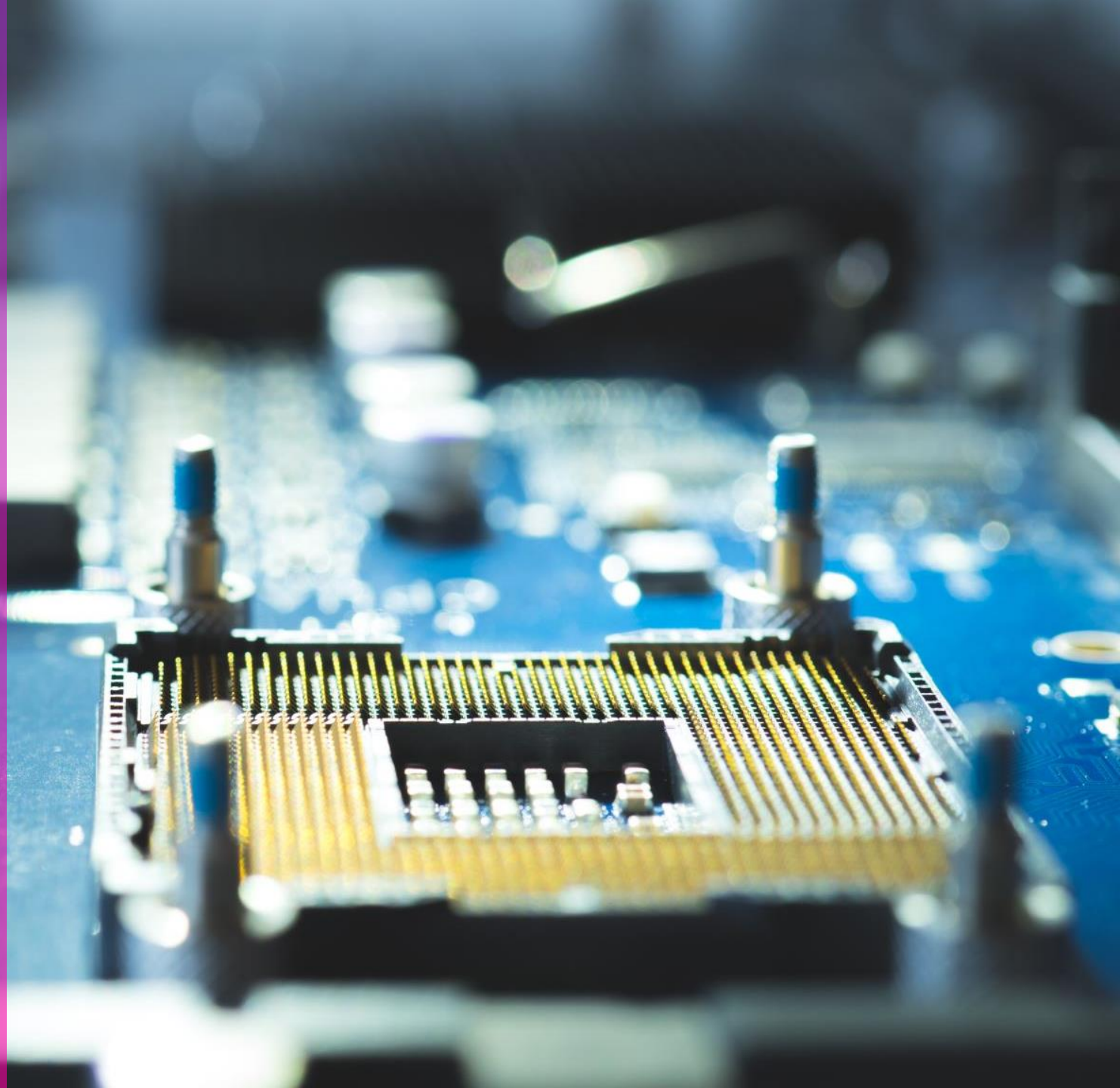
gRPC vs Restfull

- Rest servislerde yapılan request neticesinde response'un alınabilmesi için gönderilen tüm dataların topyekün işlenmesi gerekmektedir. Halbuki gRPC'de ise yapılan request neticesinde response, tüm dataların işlenmesini beklemeksizin alınabilmekte ve veriler parça parça işlendikçe bütünden bağımsız bir şekilde response edilmektedir. Bu durum gRPC'de **Data Stream** olarak nitelendirilmektedir.
- gRPC isteklerinde encoding ve decoding işlemleri istemcide gerçekleştirilmektedir. Böylece bu işlemlerin yükü serverdan arındırılmış olmaktadır.
- gRPC'de farklı platformlar ve diller arası tür dönüşümleri için serialization ve deserialization yapmaya gerek yoktur. Bunun nedeni protokol üzerinde veri tipinin önceden belirlenmiş olmasıdır ve hedef dile ait kodun ilgili protokol üzerinden(protobuf) üretilmesindendir. Nihayetinde hem client'ın hem de server'ın kodlarını inşa eden protobuf protokolü her iki uygulamanın da diline uygun servisleri inşa edecek ve böylece iletişim sürecinde ekstra bir dönüşüme gerek duyulmayacaktır.
- gRPC avantajları olduğu kadar şimdilik dezavantajlara da sahiptir. IIS ve Azure App Service üzerinde barındırılmamakta lakin Kestrel tarafından desteklenmektedir.
- Restfull servislerde tarayıcı desteği varken, gRPC'de kısmi bir destek söz konusudur.

gRPC vs API

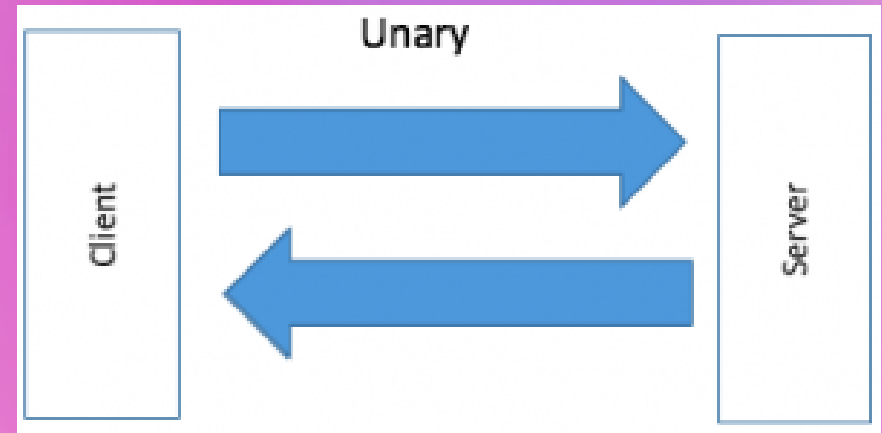
	gRPC	API
Protokol	Http/2	Http/1
Kontrakt	Var	Yok
Veri Türü	Protobuf(Binary)	Json
Security	TLS	TLS
Browser Desteği	Kısıtlı	Var

gRPC Konseptleri ve Client/Server Arasındaki İletişim Tipleri Nelerdir?



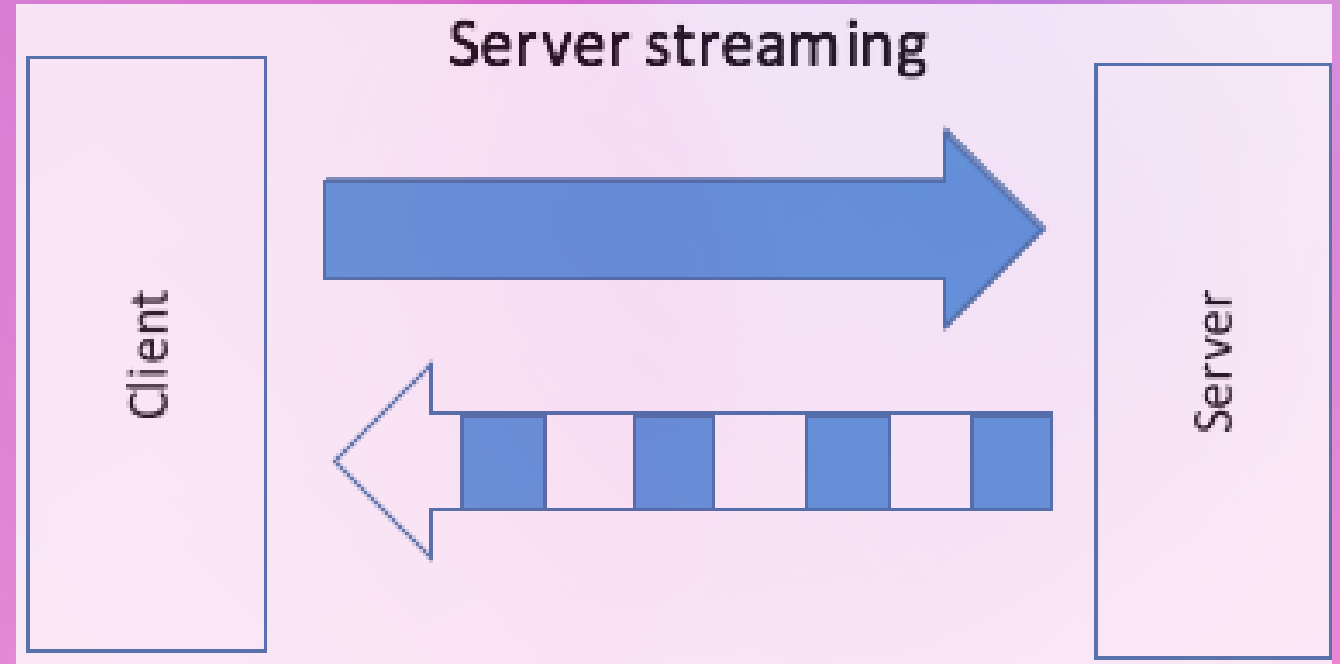
Unary

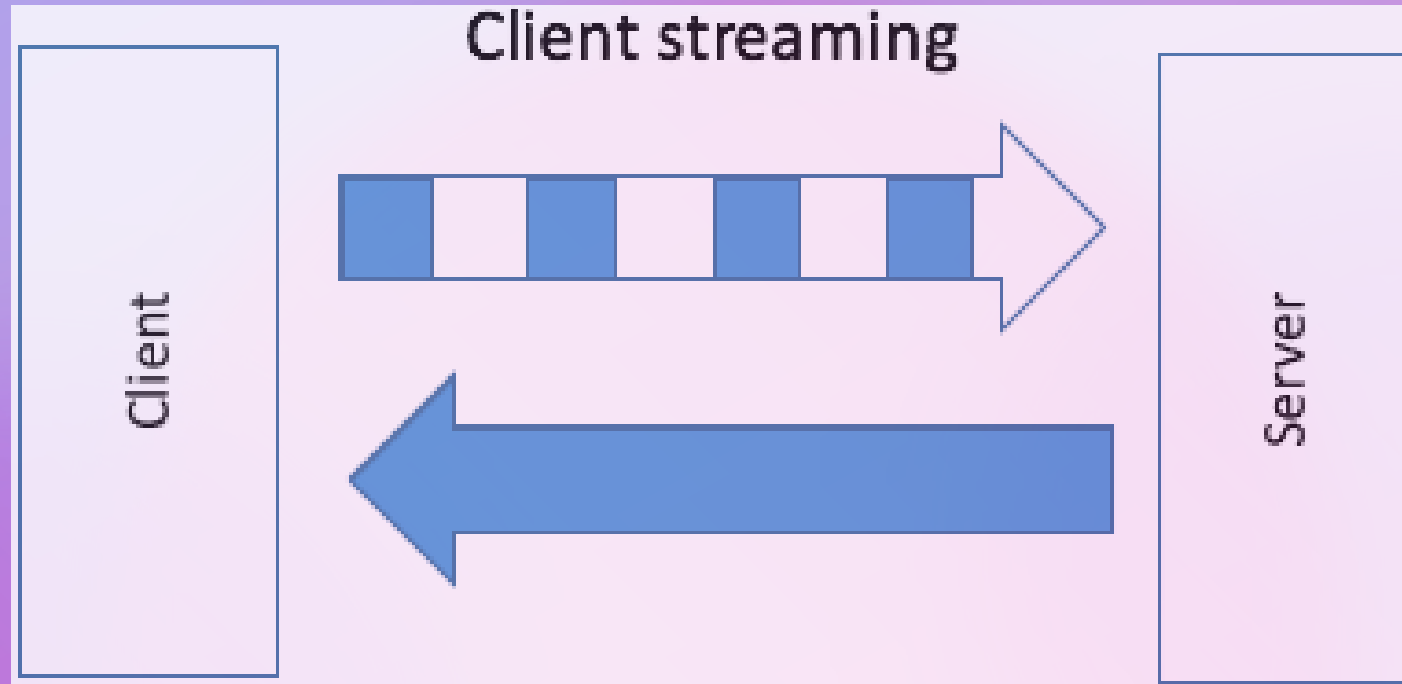
- Client'ın server'a tek bir istek gönderdiği ve normal bir işlem çağrısı gibi tek bir yanıt geri aldığı RPC türüdür.



Server Streaming

Client'in server'a tek bir istek gönderdiği ve server'ın stream dönmeye başladığı RPC türüdür.



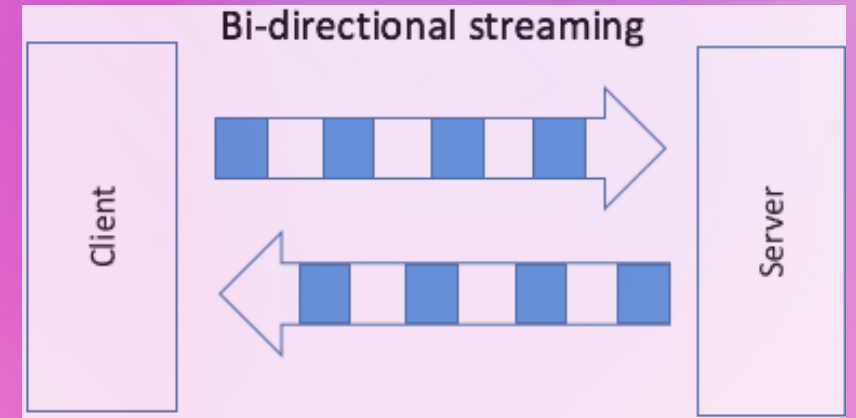


Client Streaming

- Client'ın server'a stream mesaj gönderdiği ve server'ın tek bir response döndürüğü RPC türüdür. Yani client birden çok mesaj gönderiyor, server buna karşılık tek bir cevap döndürüyor. Anlayacağınız Server Streaming'in tam tersi bir akış söz konusudur.

Bi-directional Streaming

- Client'ın server'a stream mesaj gönderdiği ve server'ın stream response döndürdüğü RPC türüdür. Yani hem client hem de server karşılıklı message streaming gerçekleştirmektedir ve böylece birden çok mesaj transferi sağlanabilmektedir. bknz: Duplex messaging



gRPC Yaşam Döngüsü (Lifecycle)

