SignalR

il WebSocket che tanto ci mancava

#CodeGen

#dotnetconf

@cloudgen_verona

Main sponsor

Community sponsors

Syncfusion®

Packt

MANNING

elmah.io

Udemy™

JET BRAINS

Local Event

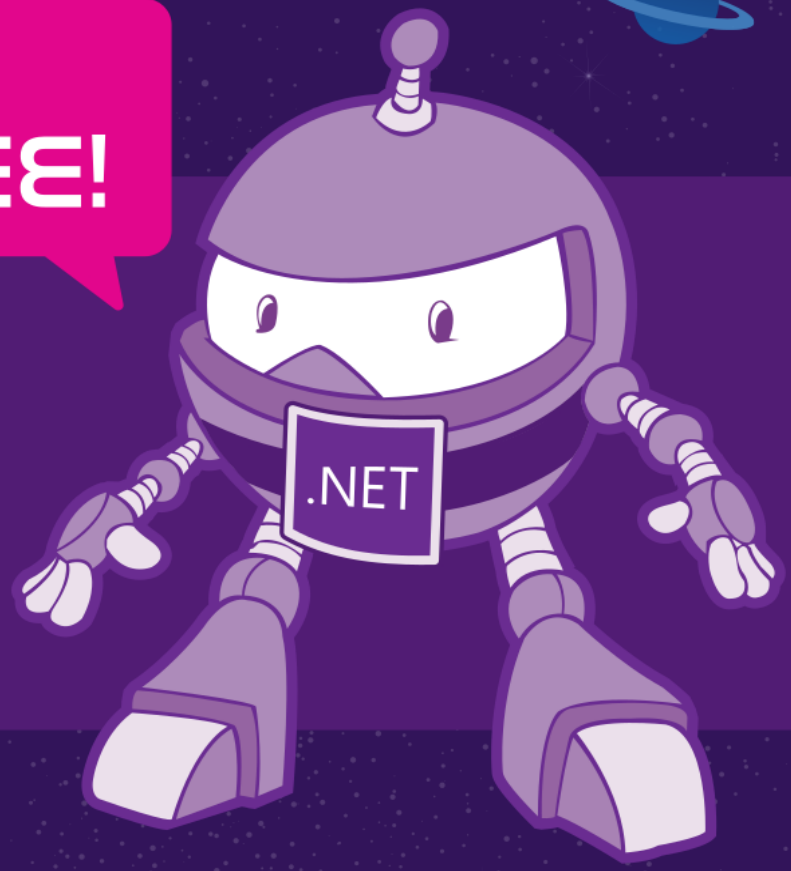
CloudGen
omnidi verso l'evoluzione

It's FREE!

.NET Conf

Discover the world of .NET

September 12-14, 2018



Tune in: www.dotnetconf.net

ANDREA TOSATO

ATosato86

andreatosato

andrea.tosato

CloudGen
ominidi verso l'evoluzione

Storia

Dalla nascita alla ri-nascita

I creatori

David Fowler

Damian Edwards

SignalR nacque nel 2011

Socket.io, release (0.1.0), il 19/03/2010

La storia in ASP .NET

Fu portato nel progetto ASP.NET nel 2013.
In quell'anno WebSocket era un protocollo, appena standardizzato, ma che molti browser non conoscevano.

Si utilizzavano tecnichiche come **polling** e altre tecniche come il **Server-Side Event** non erano ancora completamente implementate dai browser.

SignalR nacque per risolvere il problema e inserire un supporto all'interno dello stack ASP .NET. SignalR divenne una libreria di riferimento per il RealTime.

La storia in ASP .NET

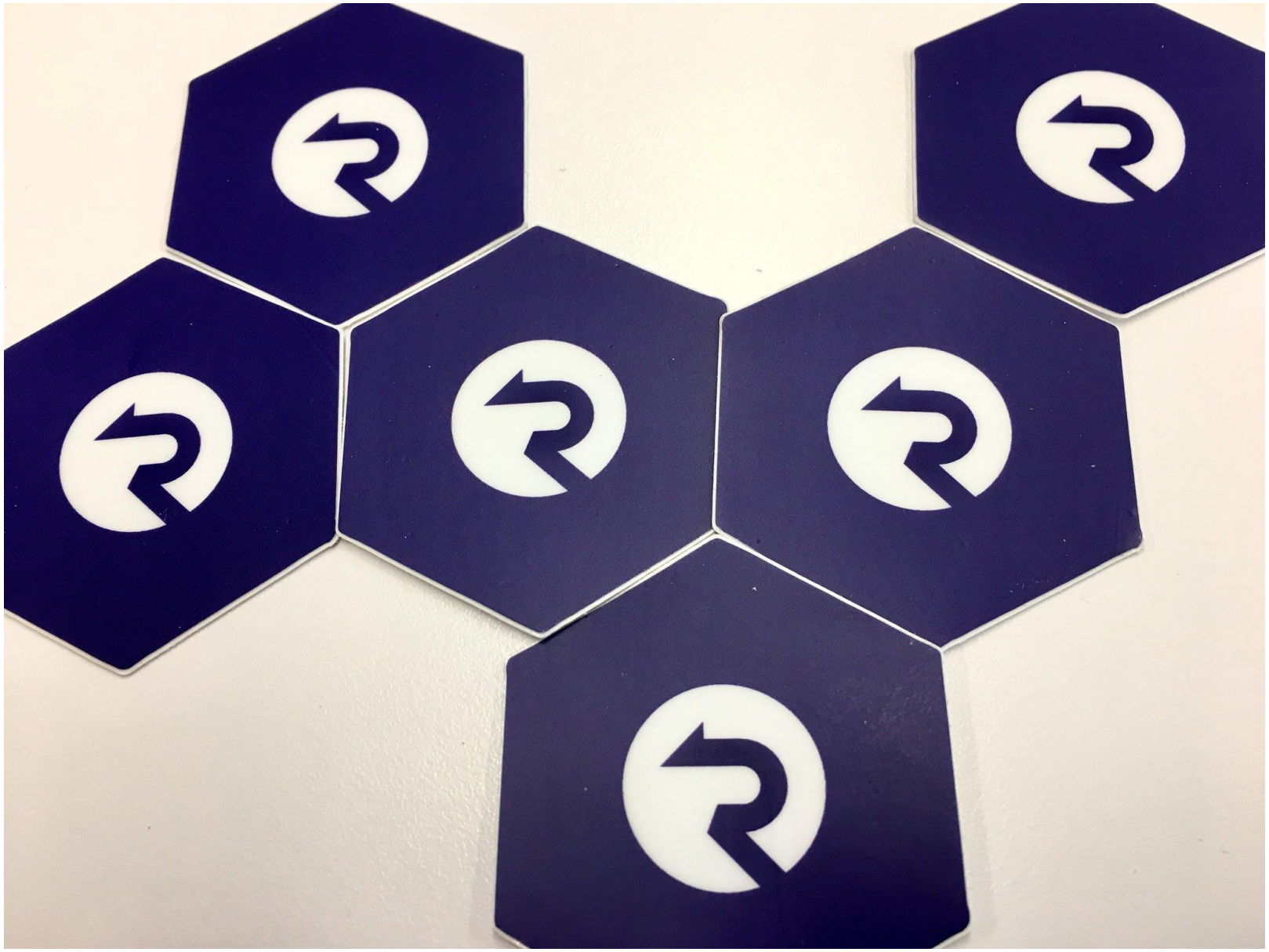
SignalR quindi si fa carico di **negoziare il miglior protocollo** di trasporto disponibile per la comunicazione.

SignalR, alla nascita, era strettamente accoppiato a **jQuery**.
Il web di allora lo era quasi interamente.

Altre funzionalità native dello strumento:

- Riconnessione automatica
- Scalabilità chiavi in mano (Service-Bus)

La rinascita



La rinascita

Nel 2018 viene interamente riscritto.

Nessuna dipendenza con la libreria jQuery.

Scalabilità (Redis – Azure SignalR Service)

Cross-platform

Estendibile

Disponibile dalla versione ASP .NET Core 2.1

Client: **Javascript – Typescript – .NET - Node.js - Python**

Prossime release: **Java e C++**

Planning: Go, PHP

I pacchetti disponibili per ASP .NET Signlar

ASP .NET: https://www.nuget.org/packages/Microsoft.AspNet.SignalR/
JavaScript: https://www.nuget.org/packages/Microsoft.AspNet.SignalR.JS/

C++: https://github.com/SignalR/SignalR-Client-Cpp

NodeJS: None

PHP: None

Go: None

Python: https://pypi.org/project/signalr-client/

Java: https://github.com/SignalR/java-client

I pacchetti disponibili per ASP .NET Core

ASP .NET Core: *'Microsoft.AspNetCore.SignalR.Client'* on NuGet (https://www.nuget.org/packages/Microsoft.AspNetCore.SignalR/)
Java: *'com.microsoft.aspnet:signalr'* on Maven (https://search.maven.org/artifact/com.microsoft.aspnet/signalr)
JavaScript (including **NodeJS**): '@aspnet/signalr' on NPM (https://www.npmjs.com/package/@aspnet/signalr)
C++: (Prototype) https://github.com/aspnet/SignalR/tree/master/clients/cpp (**no official release planned yet**)
Swift: (**Unofficial**) SwiftSignalRClient on CocoaPods (https://github.com/moozzyk/SignalR-Client-Swift)
PHP ?
Go: Planning
Python: Planning

https://github.com/aspnet/SignalR/tree/release/2.2/specs Per contribuire

Protocolli di trasporto

Browser supportati, scenari legacy

> A transport is required to have the following attributes:
>
> **Duplex**
>
> **Binary-safe**
>
> **Text-safe**

A transport is required to have the following attributes:

Duplex

Able to send messages from Server to Client and from Client to Server

A transport is required to have the following attributes:

Binary-safe

Able to transmit arbitrary binary data, regardless of content

A transport is required to have the following attributes:

Text-safe

Able to transmit arbitrary text data, preserving the content. Line-endings must be preserved but may be converted to a different format. For example **\r\n** may be converted to **\n**. This is due to quirks in some transports (Server Sent Events). *If the exact line-ending needs to be preserved, the data should be sent as a Binary message.*

SignalR

The only transport which fully implements the duplex requirement is **WebSockets**

The others are "half-transports" which implement one end of the duplex connection.

They are used in **combination** to achieve a duplex connection.

SignalR

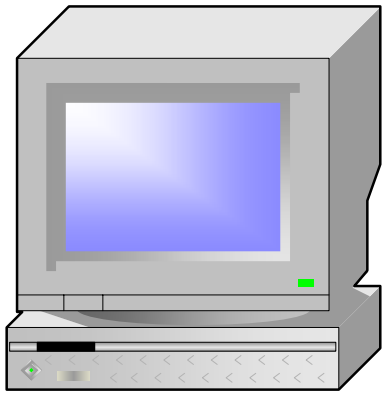
ASP.NET Core SignalR was built to be a general purpose RPC(remote procedure call) based communication library.

It supports:

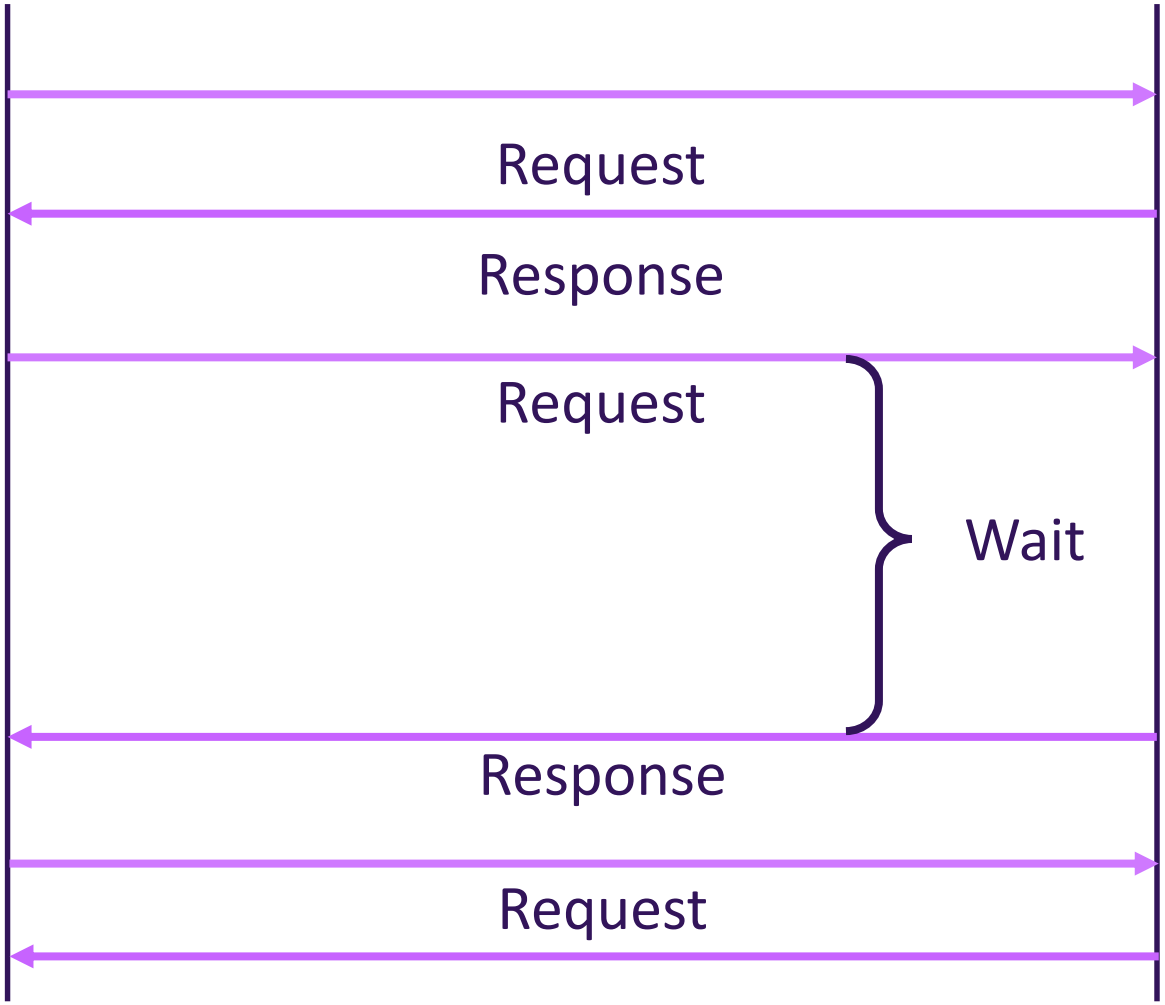
- Multicast
- Broadcast
- Groups
- Bidirectional RPC
- Streaming

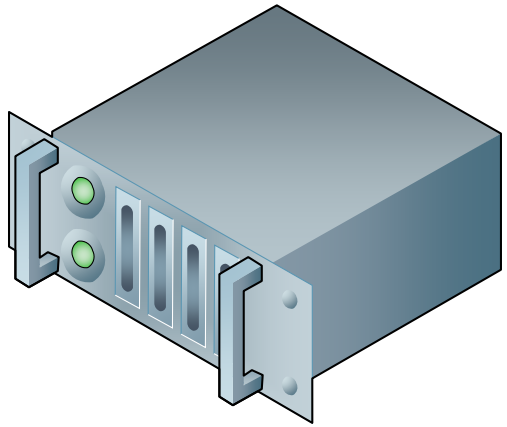
Long Polling

Richieste continue Client-Server

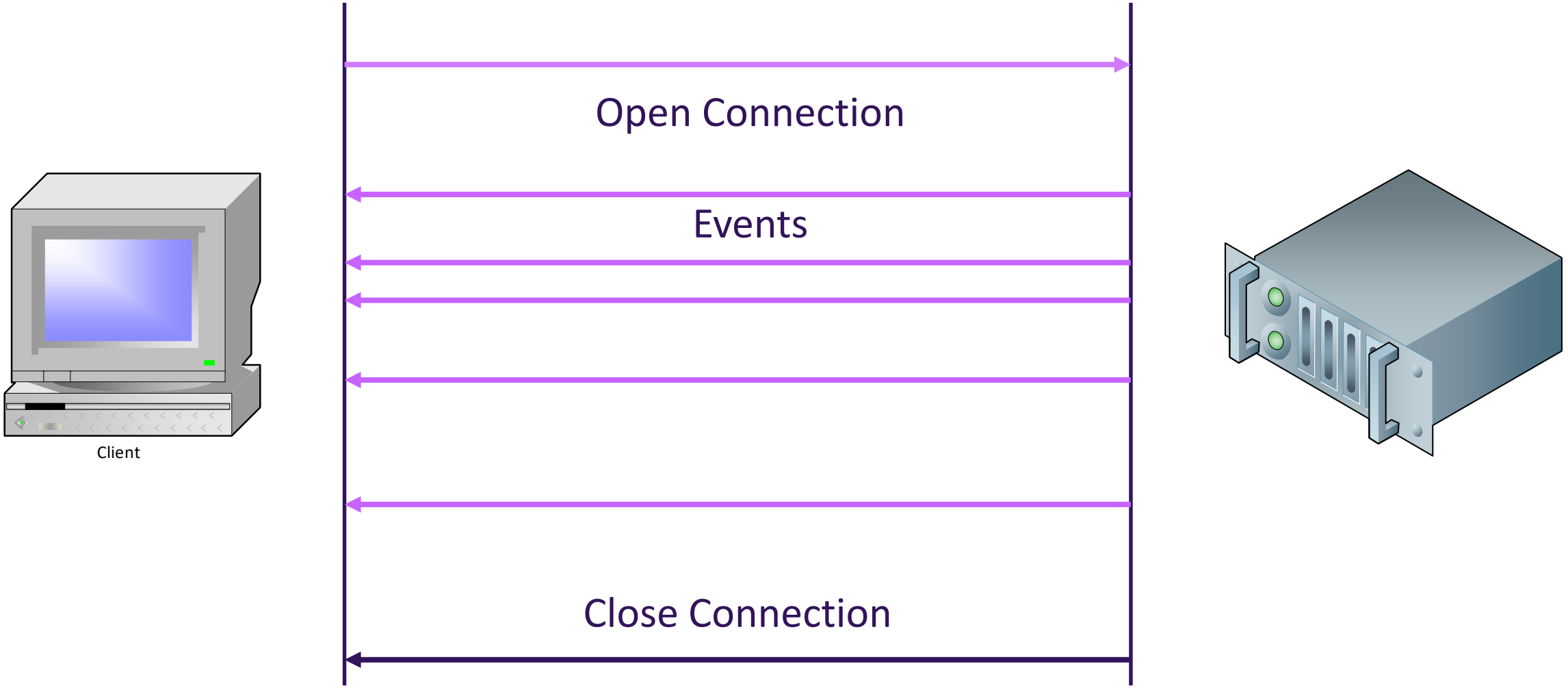


Client





Connessione uniderezionale tra Server e Client



Server-Sent Events

Server-sent events - LS

Usage % of all users

Global 88.08%

Method of continuously sending data from a server to the browser, rather than repeatedly requesting it (EventSource interface, used to fall under HTML5)

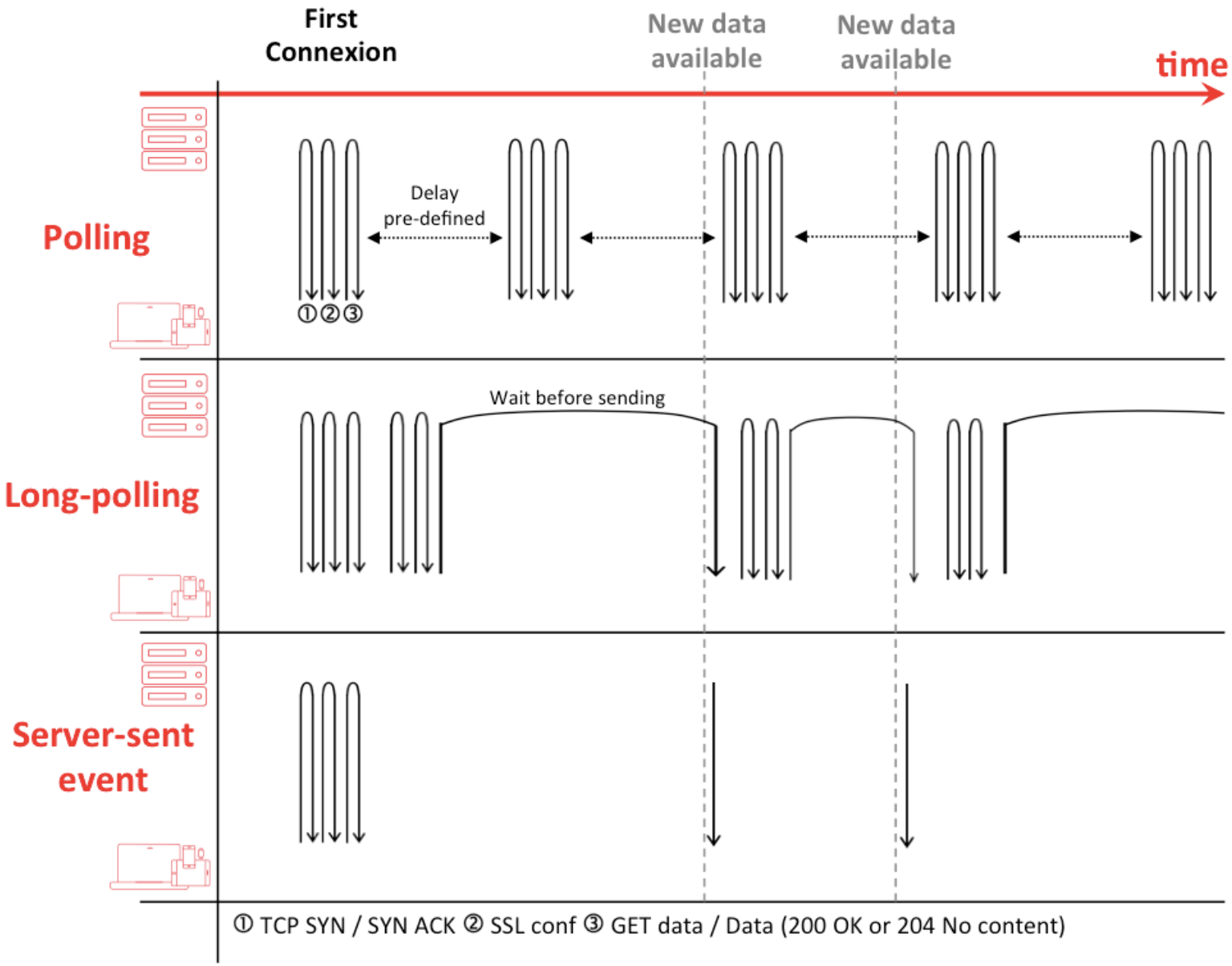
Current aligned | Usage relative | Date relative | Show all

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			63						
			66		10.3				
			67		11.2				4
11	17	61	68	11.1	11.4	all	67	11.8	7.2
	18	62	69	12	12				
		63	70	TP					
			71						

Server-Sent Events

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android *	Blackberry	Opera Mobile *	Chrome Android	Firefox Android	IE Mobile	UC for Android	Samsung Internet	QQ	Baidu
		44	51		36												
		45	52		37												
		46	53	3.1	38												
		47	54	3.2	39												
		48	55	4	40	3.2											
		49	56	5	41	4.1											
		50	57	5.1	42	4.3											
		51	58	6	43	5.1											
		52	59	6.1	44	6.1		2.1									
		53	60	7	45	7.1		2.2									
		54	61	7.1	46	8		2.3									
		55	62	8	47	8.4		3									
6	12	56	63	9	48	9.2		4									
7	13	57	64	9.1	49	9.3		4.1									
8	14	58	65	10	50	10.2		4.3							4		
9	15	59	66	10.1	51	10.3		4.4		12					5		
10	16	60	67	11	52	11.2		4.4.4	7	12.1			10		6.2		
11	17	61	68	11.1	53	11.4	all	67	10	46	67	60	11	11.8	7.2	1.2	7.12
	18	62	69	12		12											
		63	70	TP													
			71														

Avidità dei protocolli



Web Socket

WebSocket è una tecnologia web che fornisce canali di comunicazione **full-duplex** attraverso una singola connessione **TCP**. L'API del WebSocket è stata standardizzata dal W3C e il protocollo WebSocket è stato standardizzato dall'IETF come RFC 6455.

WebSocket è disegnato per essere implementato sia lato browser che lato server, ma può essere utilizzato anche da qualsiasi applicazione client-server.

Fonte: https://it.wikipedia.org/wiki/WebSocket

Web Socket

Web Sockets - LS

Bidirectional communication technology for web apps

Usage % of all users

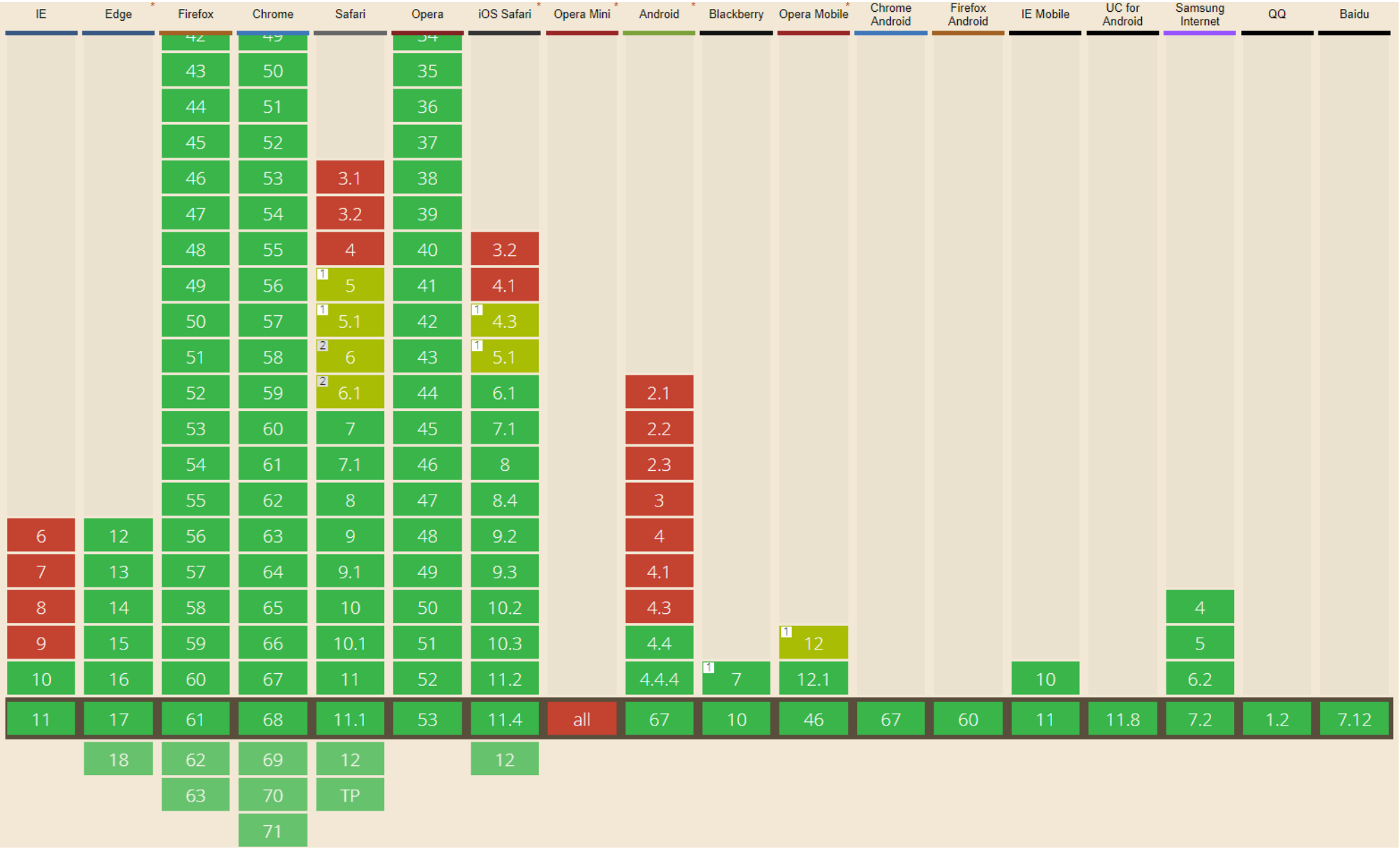
Global 92.64% + 0.21% = 92.84%

unprefixed: 92.64% + 0.17% = 92.8%

Current aligned | Usage relative | Date relative | Show all

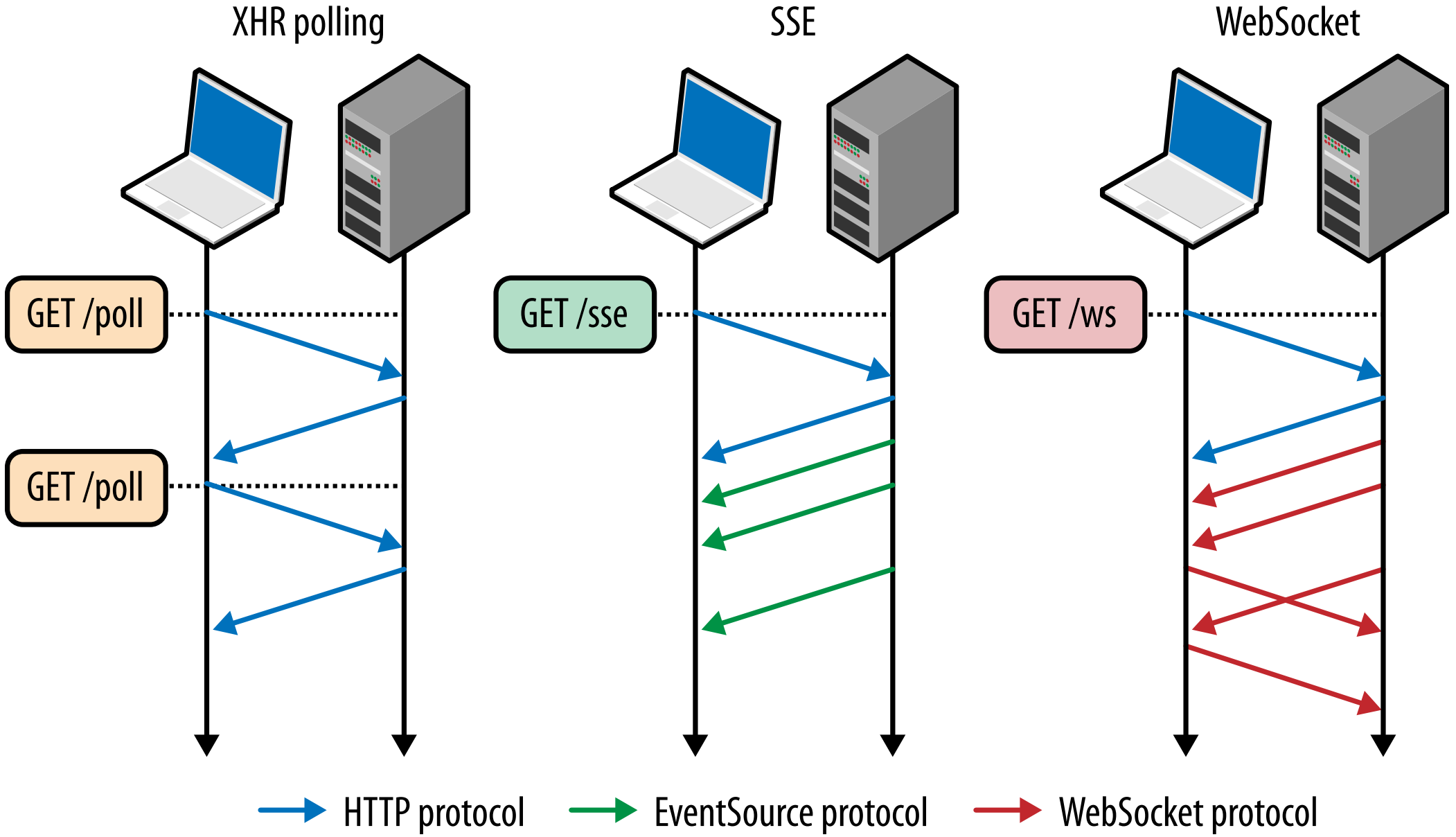
IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			63						
			66		10.3				
			67		11.2				4
11	17	61	68	11.1	11.4	all	67	11.8	7.2
	18	62	69	12	12				
		63	70	TP					
			71						

Web Socket



IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android *	Blackberry	Opera Mobile *	Chrome Android	Firefox Android	IE Mobile	UC for Android	Samsung Internet	QQ	Baidu
		43	50		35												
		44	51		36												
		45	52		37												
		46	53	3.1	38												
		47	54	3.2	39												
		48	55	4	40	3.2											
		49	56	1 5	41	4.1											
		50	57	1 5.1	42	1 4.3											
		51	58	2 6	43	1 5.1											
		52	59	2 6.1	44	6.1		2.1									
		53	60	7	45	7.1		2.2									
		54	61	7.1	46	8		2.3									
		55	62	8	47	8.4		3									
6	12	56	63	9	48	9.2		4									
7	13	57	64	9.1	49	9.3		4.1									
8	14	58	65	10	50	10.2		4.3							4		
9	15	59	66	10.1	51	10.3		4.4		1 12					5		
10	16	60	67	11	52	11.2		4.4.4	1 7	12.1			10		6.2		
11	17	61	68	11.1	53	11.4	all	67	10	46	67	60	11	11.8	7.2	1.2	7.12
	18	62	69	12		12											
		63	70	TP													
			71														

Web Socket


XHR polling
SSE
WebSocket
GET /poll
GET /poll
GET /sse
GET /ws
HTTP protocol
EventSource protocol
WebSocket protocol

Forever frame

CloudGen
ominidi verso l'evoluzione

Protocolli di messaggio

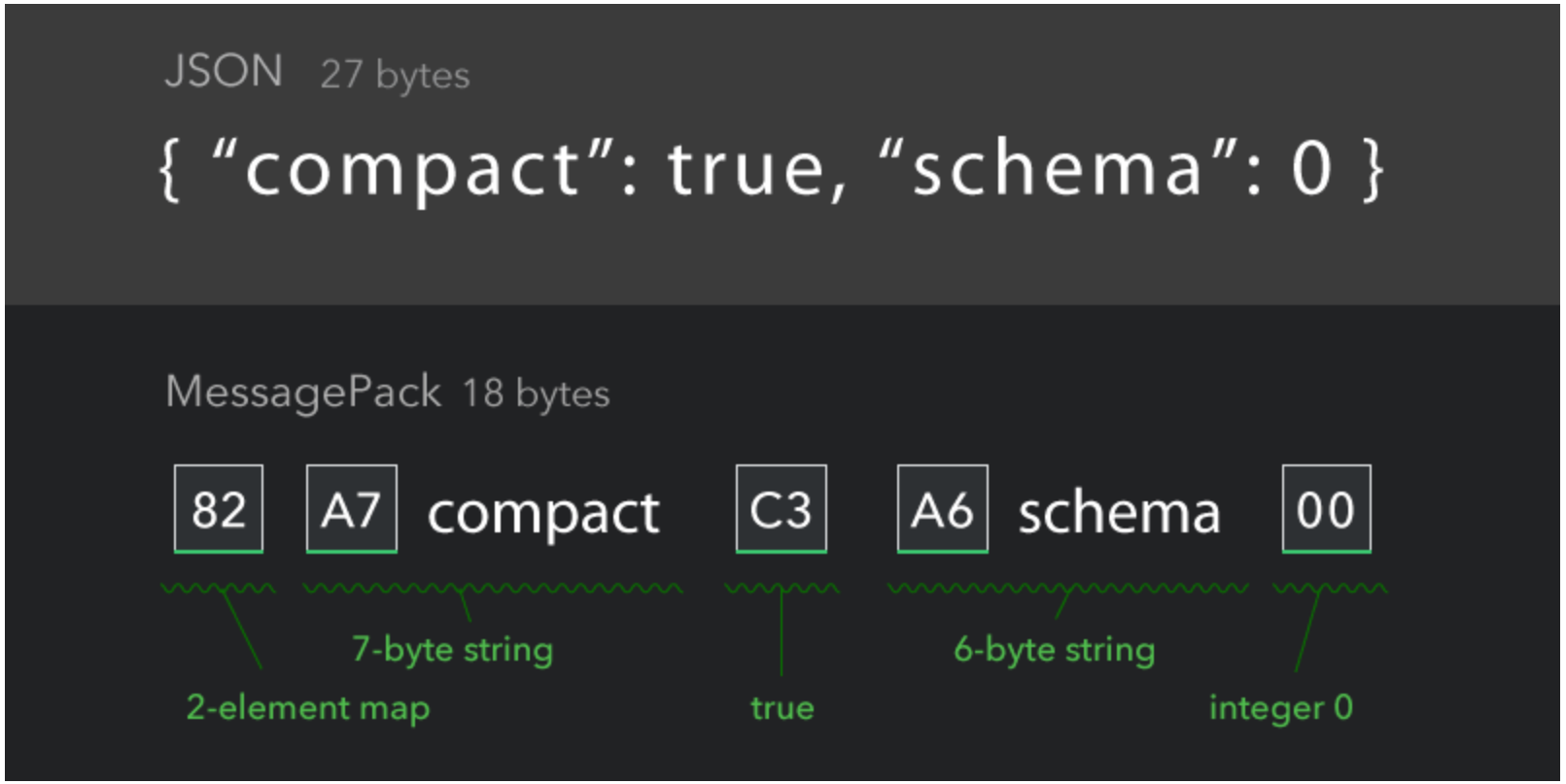
Json, msgpack, custom

JSON

JSON


```
{
  "arguments":[
    {
      "To":{
        "Username":"demo",
        "ConnectionId":"cqvsDNFlhjht7c6-NyWgvQ"
      },
      "From":{
        "Username":"andrea.tosato",
        "ConnectionId":"JLn11llArunDqCMmVqFrbg"
      },
      "TextMessage":"Buongiorno e benvenuto"
    }
  ],
  "target":"AddPrivateMessage",
  "type":1
}
```

MessagePack



https://msgpack.org/

Message Pack vs Json

JSON 235 bytes

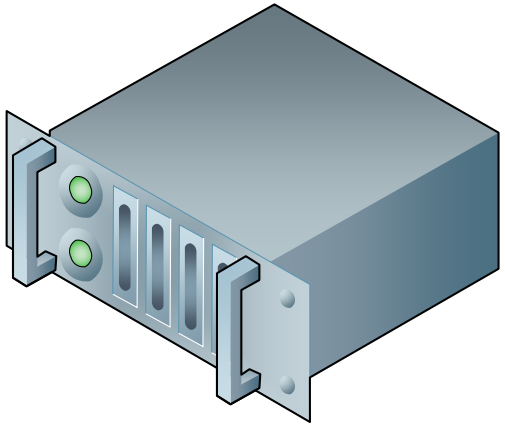
{"arguments":[{"To":{"Username":"demo","ConnectionId":"cqvsDNFlhjht7c6-NyWgvQ"},"From":{"Username":"andrea.tosato","ConnectionId":"JLn11lArunDqCMmVqFrbg"},"TextMessage":"Buongiorno e benvenuto"}],"target":"AddPrivateMessage","type":1}

MessagePack (hex) 198 bytes 84 %

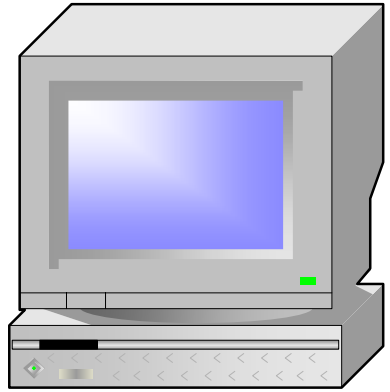
83 a9 61 72 67 75 6d 65 6e 74 73 91 83 a2 54 6f 82 a8 55 73 65 72 6e 61 6d
65 a4 64 65 6d 6f ac 43 6f 6e 6e 65 63 74 69 6f 6e 49 64 b6 63 71 76 73 44
4e 46 6c 68 6a 68 74 37 63 36 2d 4e 79 57 67 76 51 a4 46 72 6f 6d 82 a8 55
73 65 72 6e 61 6d 65 ad 61 6e 64 72 65 61 2e 74 6f 73 61 74 6f ac 43 6f 6e
6e 65 63 74 69 6f 6e 49 64 b6 4a 4c 6e 31 31 6c 6c 41 72 75 6e 44 71 43 4d
6d 56 71 46 72 62 67 ab 54 65 78 74 4d 65 73 73 61 67 65 b6 42 75 6f 6e 67
69 6f 72 6e 6f 20 65 20 62 65 6e 76 65 6e 75 74 6f a6 74 61 72 67 65 74 b1
41 64 64 50 72 69 76 61 74 65 4d 65 73 73 61 67 65 a4 74 79 70 65 01

MessagePack

Microsoft.AspNetCore.SignalR.Protocols.MessagePack



```
<script src="~/lib/signalr/signalr.js"></script>
<script src="~/lib/msgpack5/msgpack5.js"></script>
<script src="~/lib/signalr/signalr-protocol-msgpack.js"></script>
```



Client

@aspnet/signalr-protocol-msgpack

> If you want to implement a custom message protocol, ASP.NET Core SignalR has extensibility points that allow new protocols to be plugged in

Demo
Heart Rate

Connessione e trasferimento dati

Negoziazione della connessione, invio e ricezione dati

Negotiate


```
{
  "connectionId":"nw-Mk2QvXGxS5WyffMao4A",
  "availableTransports":[
    {
      "transport":"WebSockets",
      "transferFormats":["Text", "Binary"]
    },
    {
      "transport":"ServerSentEvents",
      "transferFormats":["Text"]
    },
    {
      "transport":"LongPolling",
      "transferFormats":["Text", "Binary"]
    }
  ]
}
```

Streaming

ASP.NET Core SignalR supporta streaming valori restituiti dei metodi del server.

Ciò è utile per scenari in cui verranno inviati frammenti di dati nel corso del tempo.

Quando un valore restituito viene trasmesso al client, non appena diventa disponibile, anziché attendere che tutti i dati diventino disponibili.

Streaming – oggetti da utilizzare

ChannelReader<T>

Restituisce un valore non appena disponibile.

ChannelWriter<T>

Consente la scrittura di dati all'interno di uno stream.

Demo Streaming

Streaming – Cooming soon

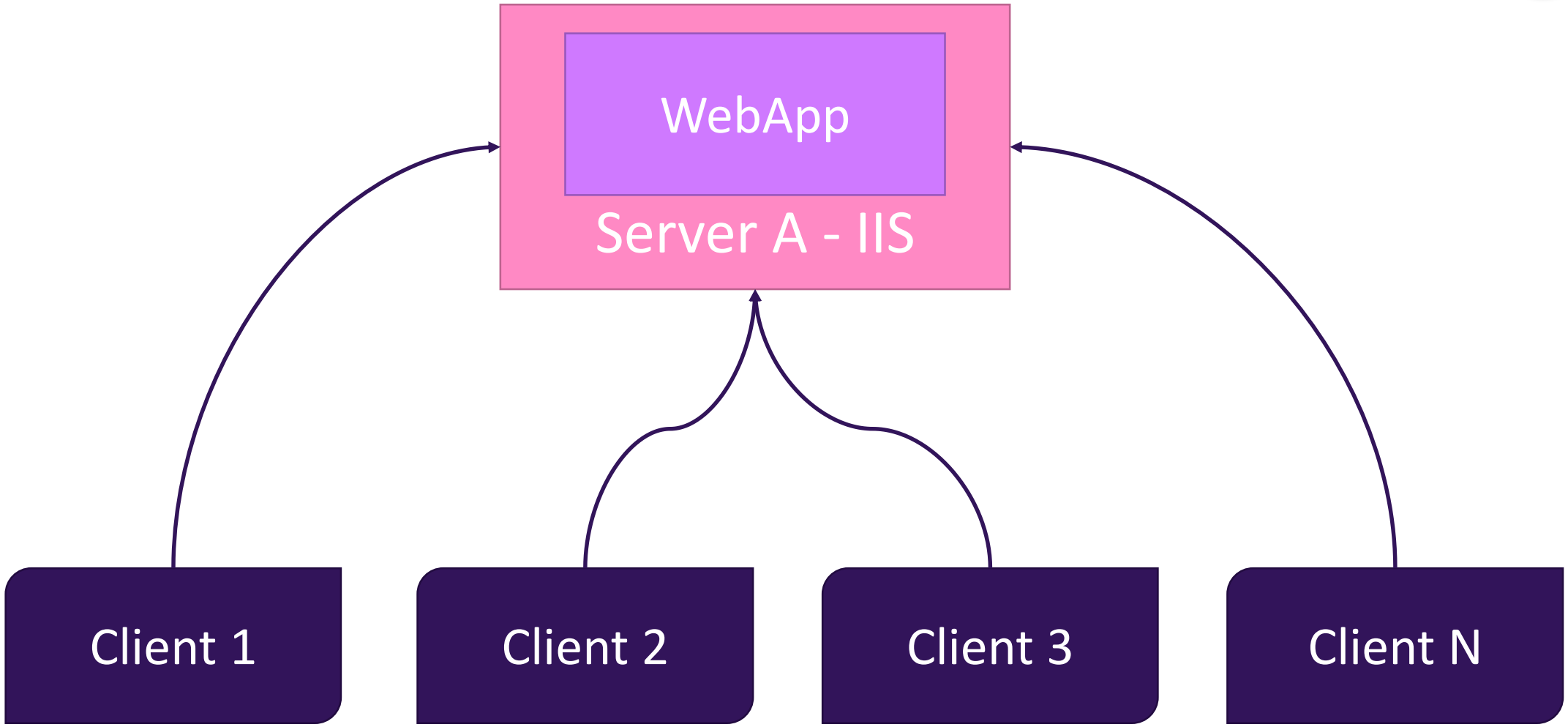
Server To Client (Work)

Client To Server (Cooming soon)

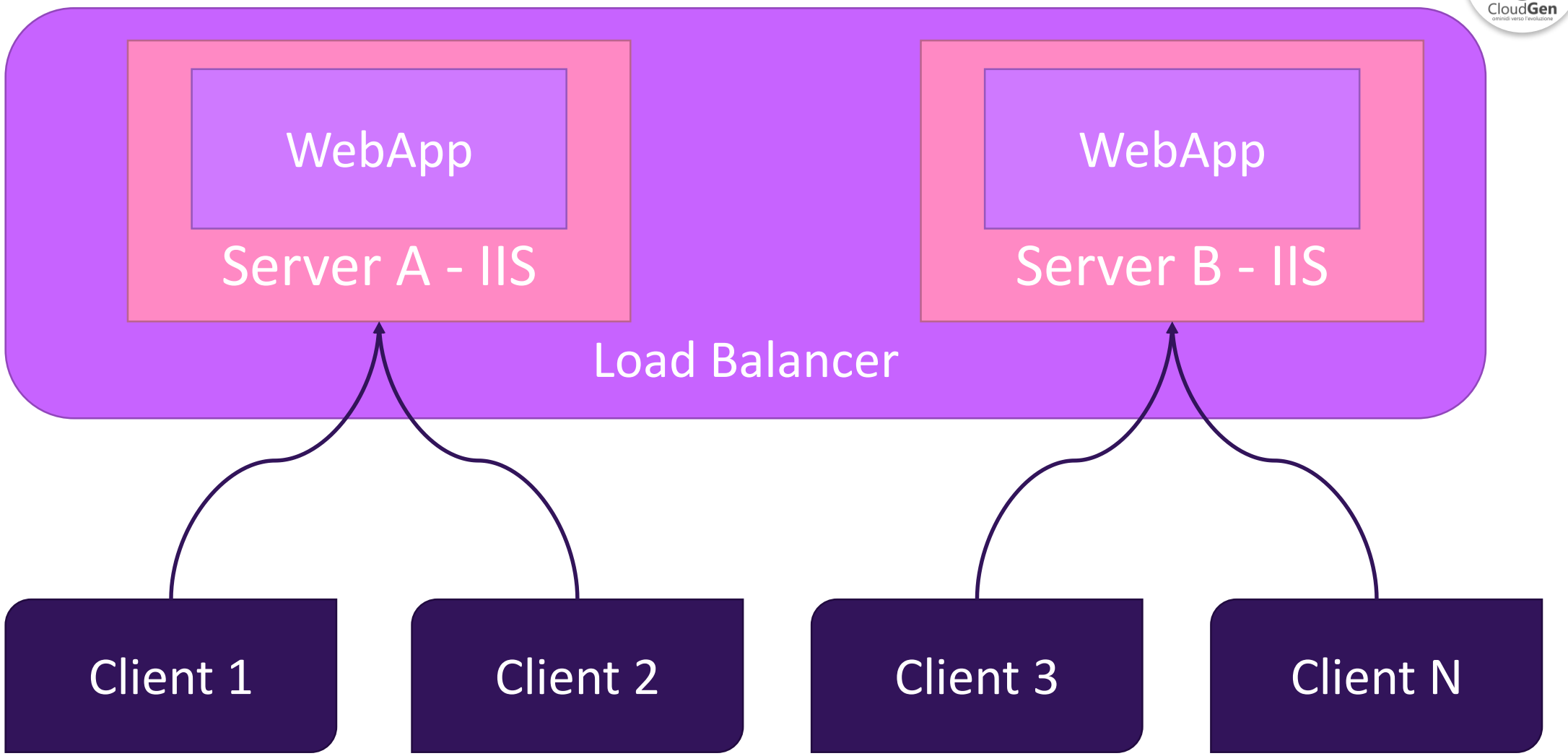
Scalare SignalR

Scenari On-Premise

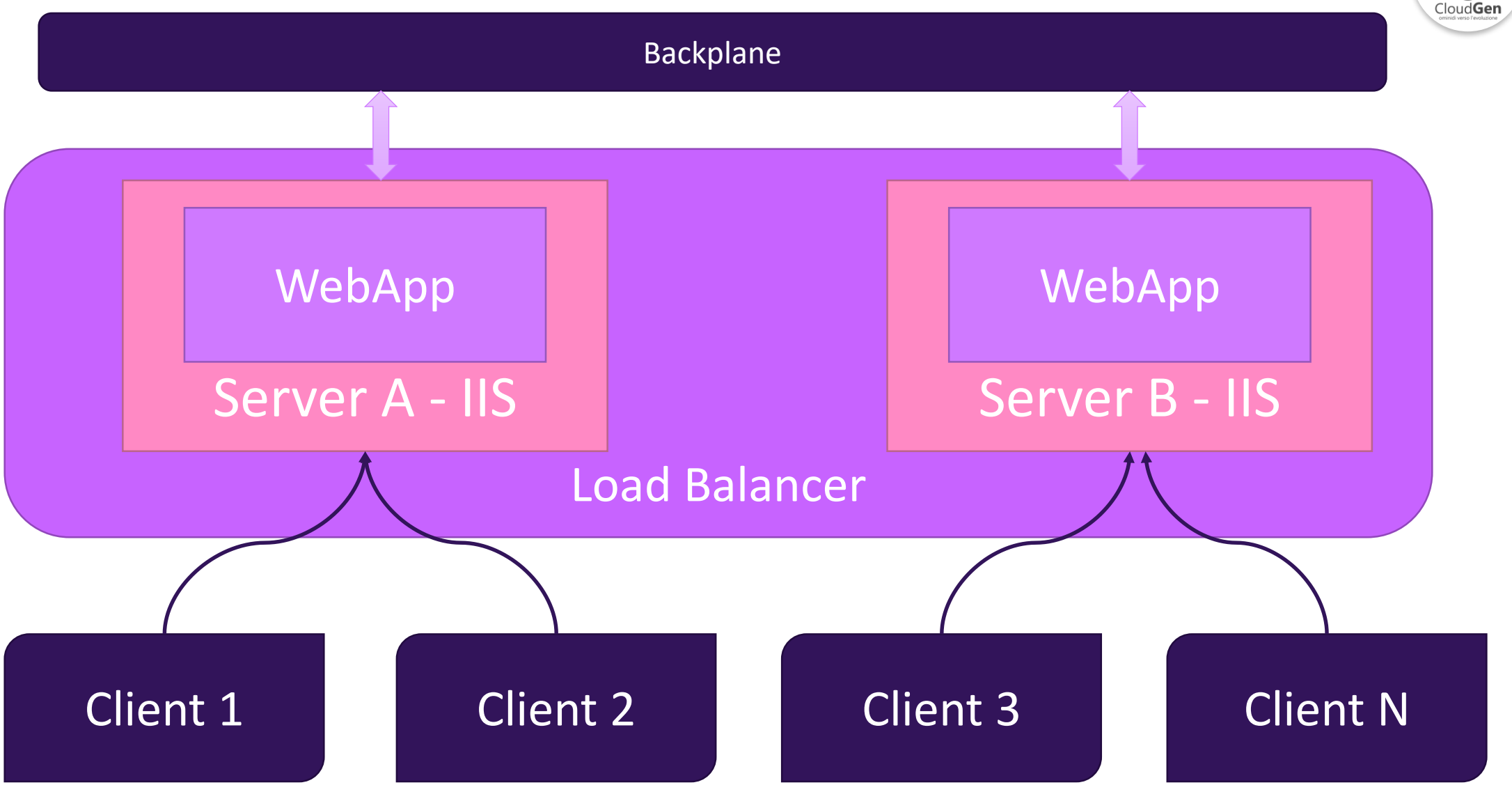
Perchè Scalare?



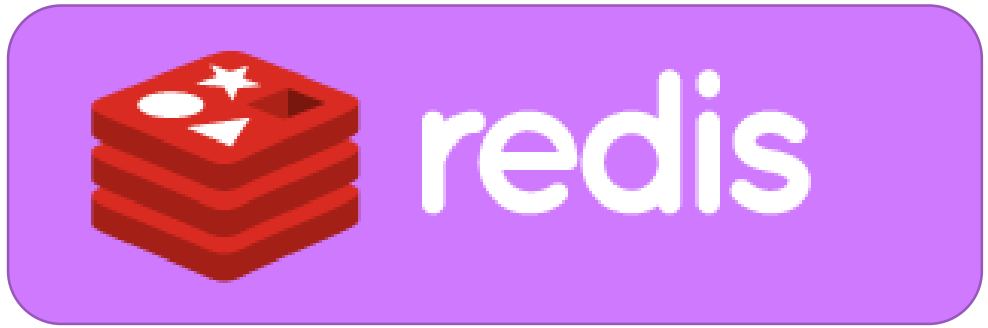
Perchè Scalare?



Perchè Scalare?

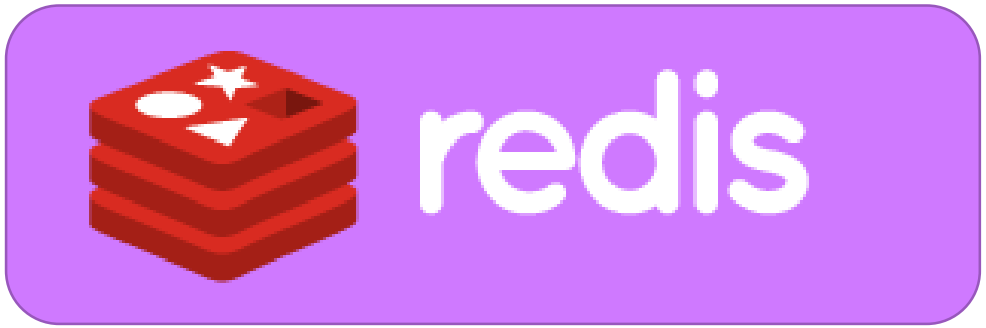


Backplane



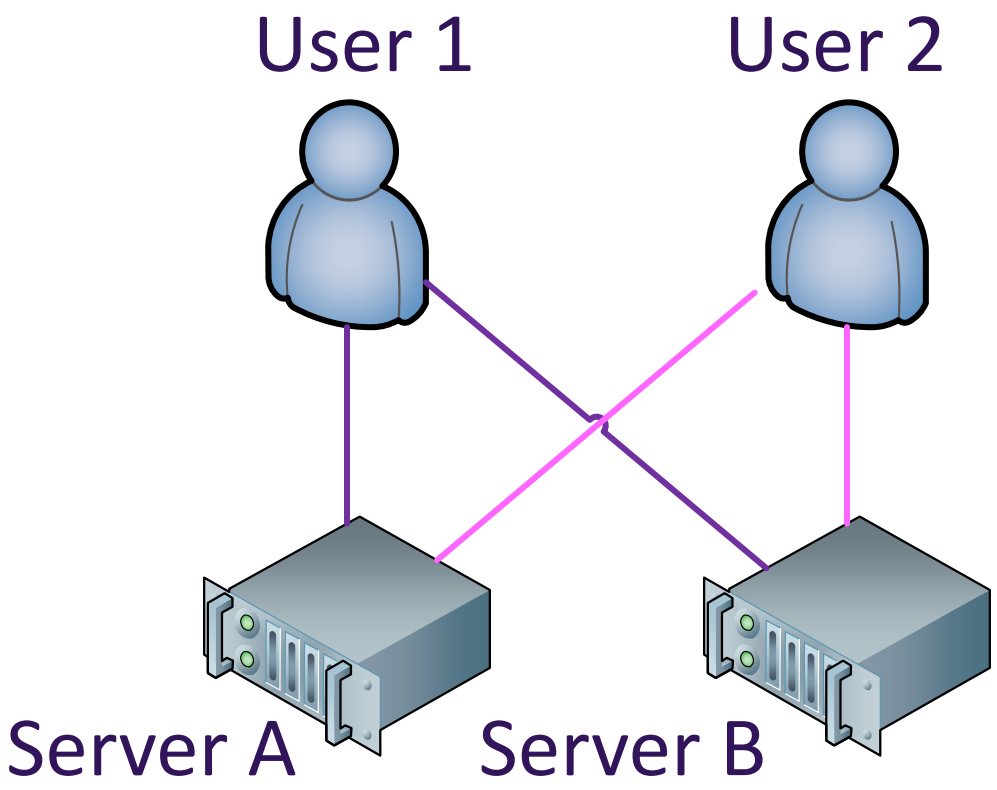
Redis - Pub/Sub
https://redis.io/topics/pubsub

Backplane

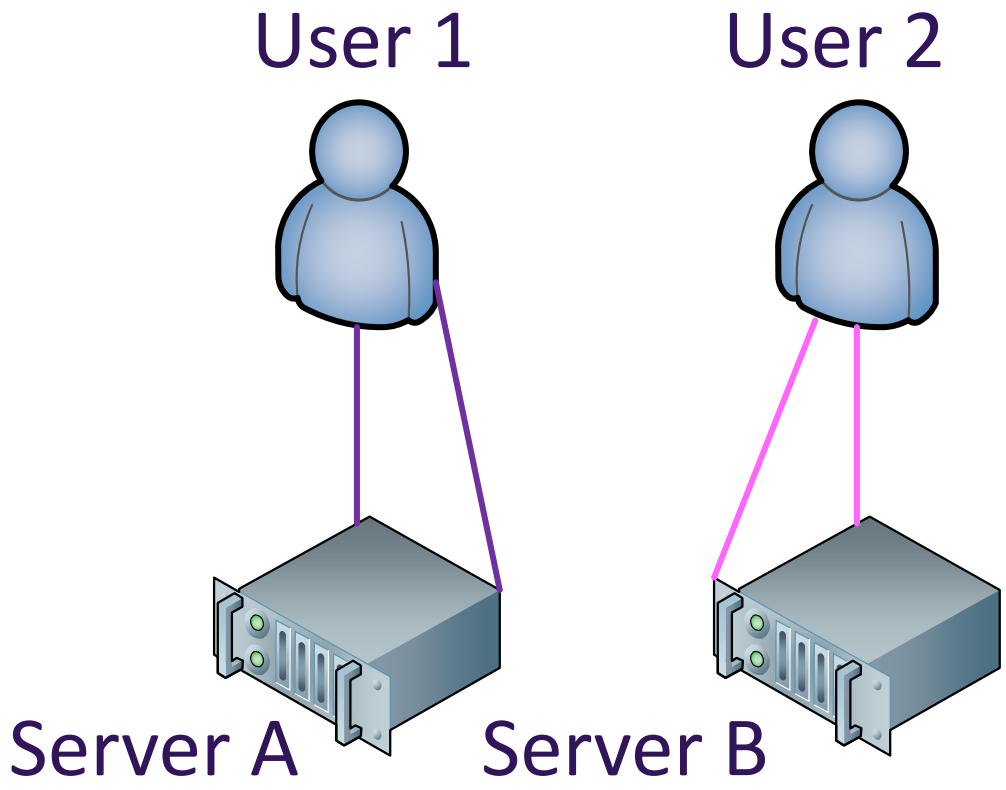


Sticky Sessions

Backplane



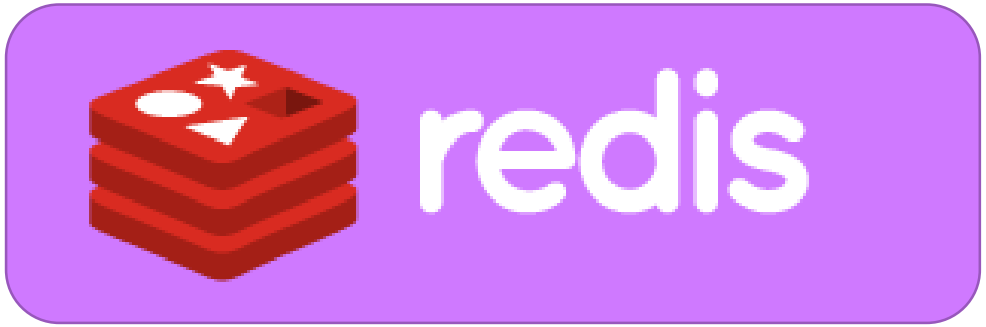
Round Robin



Round Robin
with Sticky Session

Backplane

Microsoft.AspNetCore.SignalR.Redis




```
services
        .AddSignalR()
        .AddRedis("ConnectionString");
```

Scalare SignalR

Scenari Cloud

Azure SignalR Services

SignalR è un servizio completamente gestito.

puoi implementarlo in un **ambiente multiserver** senza preoccuparti di:

- hosting,
- scalabilità,
- bilanciamento del carico
- autenticazione.

API REST

SWAGGER

https://editor.swagger.io/

SWAGGER DOC

https://github.com/Azure/azure-signalr/blob/dev/docs/swagger.json

Demo
Azure Signalr Service

Azure SignalR Service

ASP .NET versione 2.4

Supporto Azure SignalR per scalabilità chiavi in mano

Coming in 2019

CloudGen
ominidi verso l'evoluzione

Serverless

Scenario Serverless

SignalR - Serverless

Grazie a una estensione per Azure Functions non è necessario avere la parte server di SignalR su una applicazione ASP .NET Core.

L'estensione ha grossi limiti, ma per alcuni sceri può essere utile.

Current limitations
•Only supports broadcasting at this time, cannot invoke methods on a subset of connections, users, or groups
•Functions cannot be triggered by client invocation of server methods (clients need to call an HTTP endpoint or post messages to a Event Grid, etc, to trigger a function)

https://github.com/anthonychu/AzureAdvocates.WebJobs.Extensions.SignalRService

Demo
Serverless - Desktop

Esempio completo

Creare una chat con SignalR

Demo
Chat

Grazie

Domande?

andreatosato

ATosato86

Andrea.Tosato