SignalR Deep Dive: Building Servers

David Fowler – ASP.NET & Azure SignalR Architect
@davidfowl

Damian Edwards – ASP.NET Program Manager
@damianedwards

What is SignalR?

- Framework for duplex RPC & streaming between client and server
- Provides “real-time” messaging & server-push functionality
- Protocols for transport and messaging layers allow broad reach
- Server support for client addressing & grouping, e.g.:
    - Send to all clients
    - Send to this one client (connection or user)
    - Send to this group of clients

Demo: SignalR app

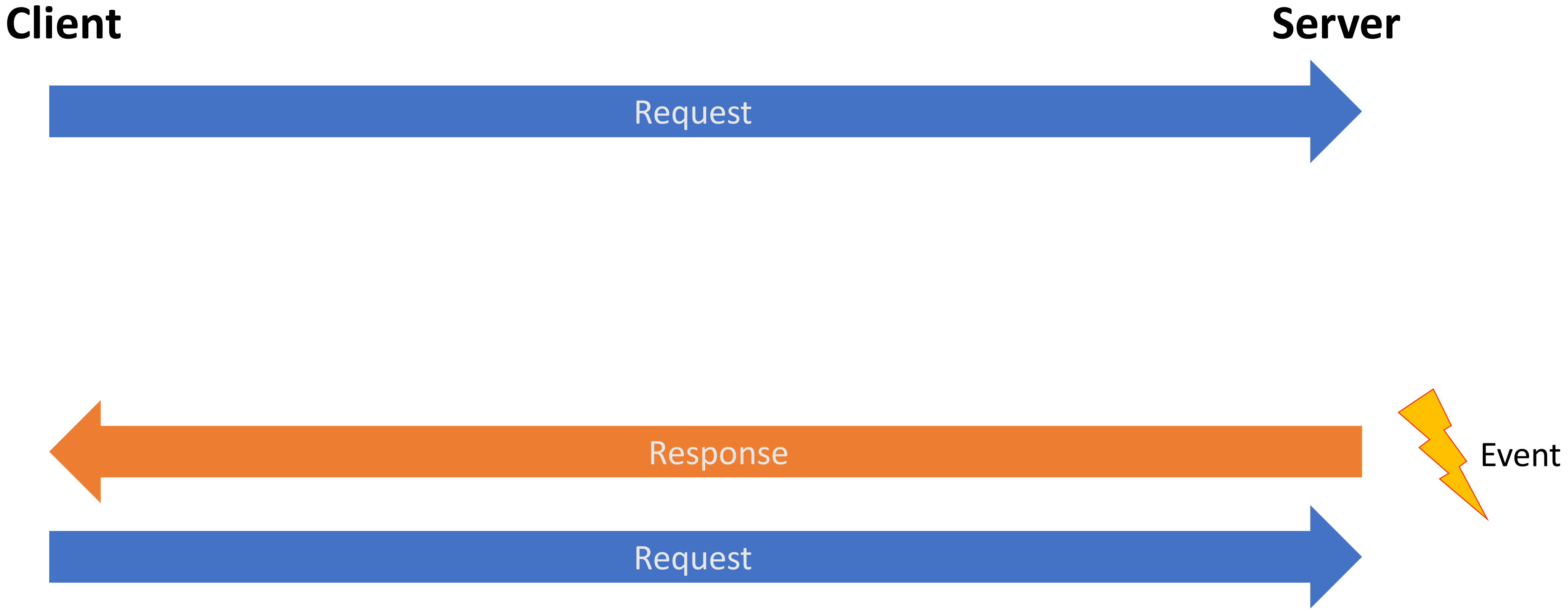
Supporting multiple clients

- SignalR supports different client types
- HTTP transports
  - WebSockets
  - ServerSentEvents
  - Long Polling
- Client SDKs
  - JavaScript
  - .NET
  - Java
  - Swift
  - Python
  - C++ (coming soon)

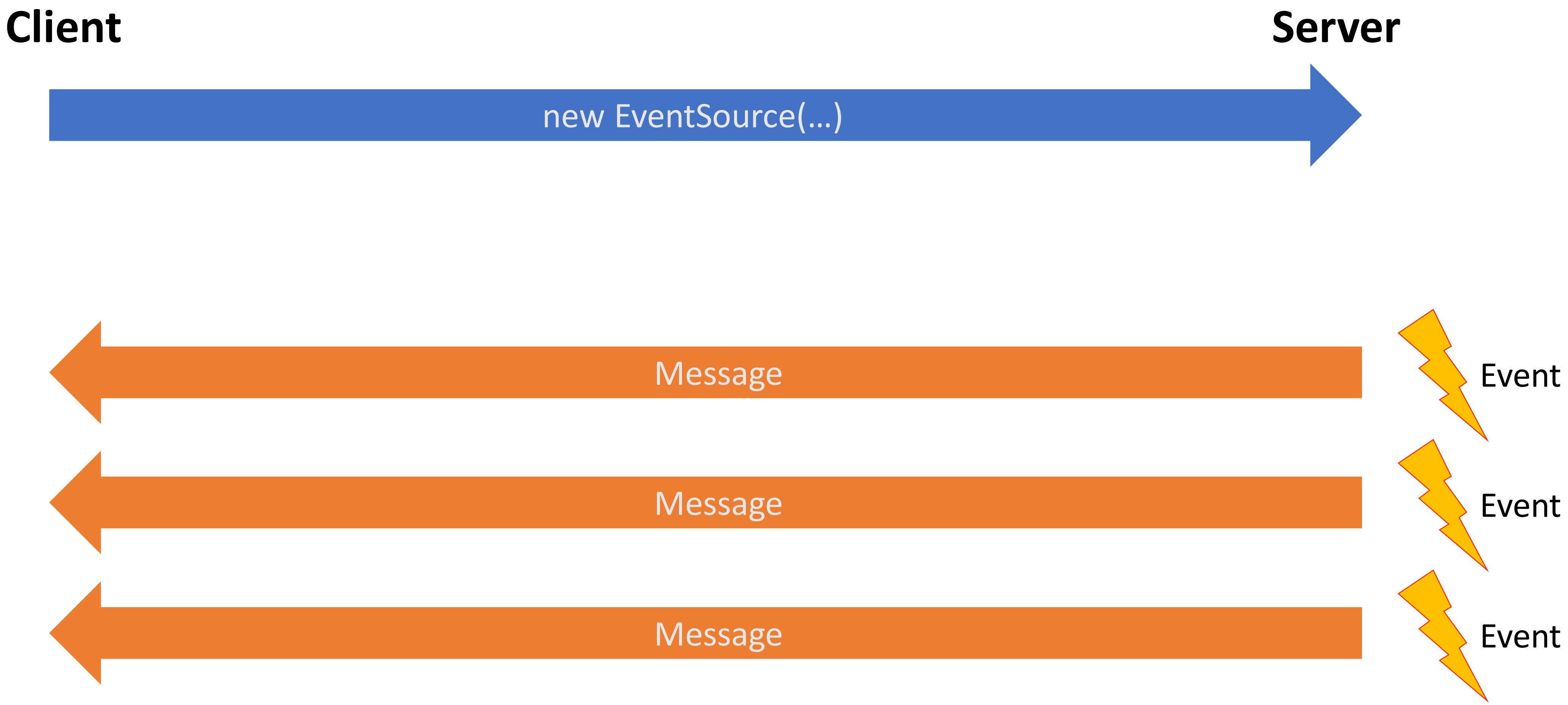
How though?

- This client flexibility is enabled by two protocol layers
- **Transport** protocol provides flexibility for connection types, e.g. HTTP vs. TCP/IP
  - https://github.com/dotnet/aspnetcore/blob/master/src/SignalR/docs/specs/TransportProtocols.md
- **Hub** protocol provides flexibility for message formats, e.g. JSON vs. MessagePack
  - https://github.com/dotnet/aspnetcore/blob/master/src/SignalR/docs/specs/HubProtocol.md

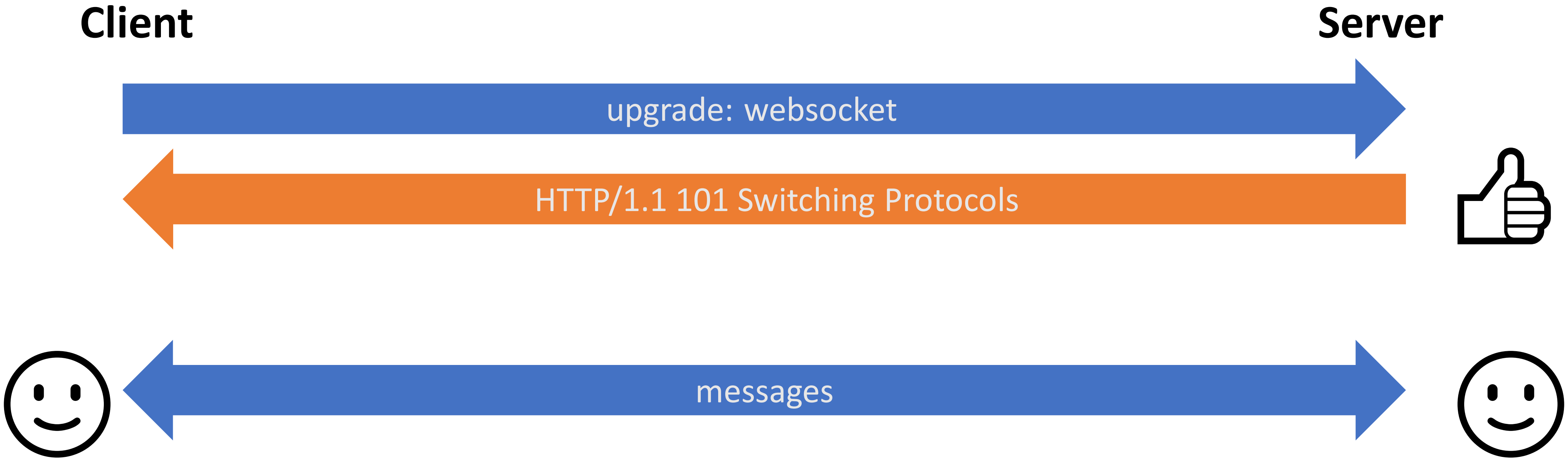
Transports: Long polling



Transports: Server-sent events



Transports: WebSockets



Client

Server

upgrade: websocket

HTTP/1.1 101 Switching Protocols

messages

Negotiate request

```
REQUEST: POST <endpoint-base>/negotiate[?negotiateVersion=<version>]

RESPONSE: {
  "connectionToken":"05265228-1e2c-46c5-82a1-6a5bcc3f0143",
  "connectionId":"807809a5-31bf-470d-9e23-afaee35d8a0d",
  "negotiateVersion":1,
  "availableTransports":[{
      "transport": "WebSockets",
      "transferFormats": [ "Text", "Binary" ]
  }]
}
```

Hub protocol message types

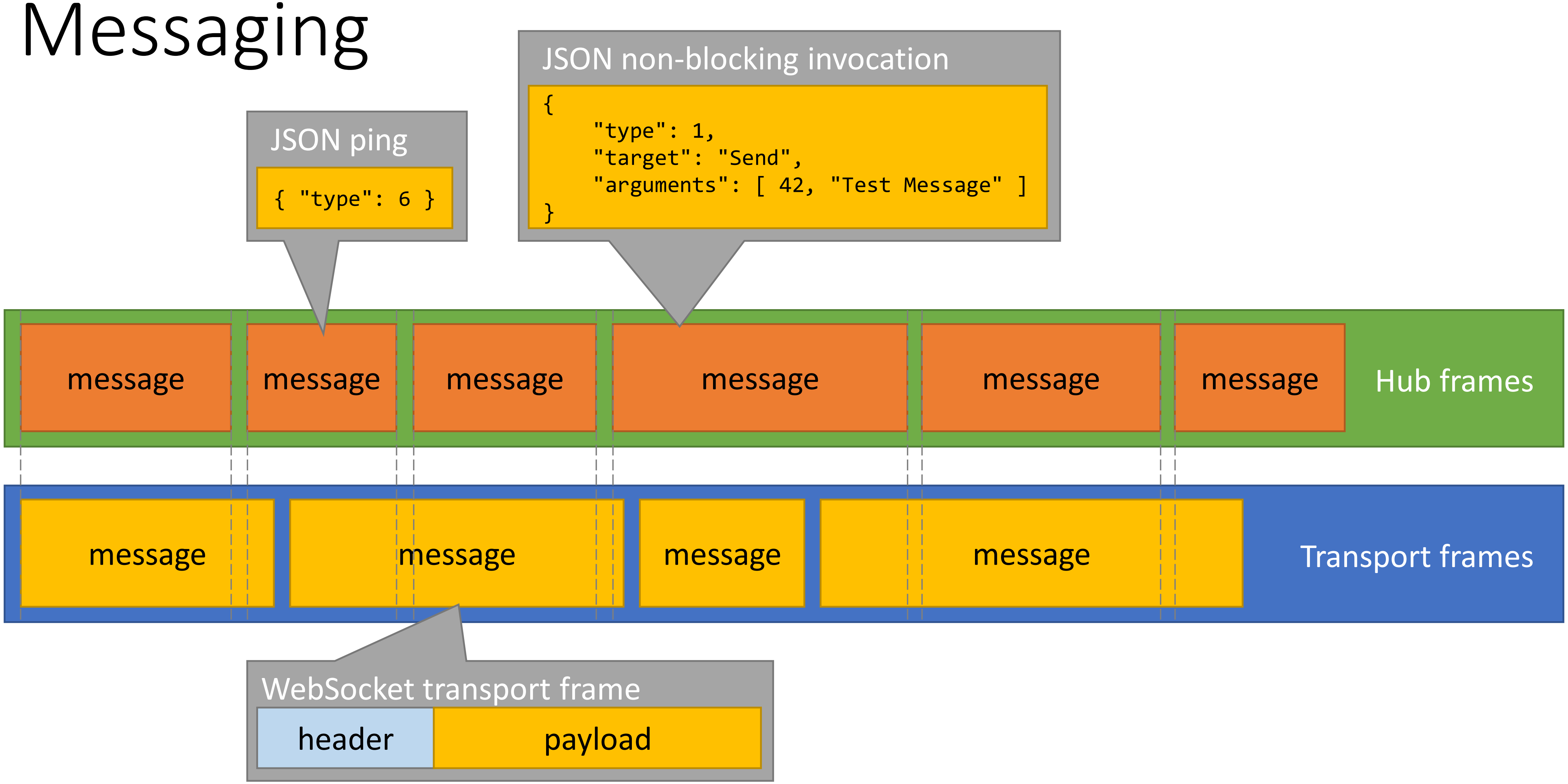
Message Name	Sender
`HandshakeRequest`	Client
`HandshakeResponse`	Server
`Close`	Callee, Caller
`Invocation`	Caller
`StreamInvocation`	Caller
`StreamItem`	Callee, Caller
`Completion`	Callee, Caller
`CancelInvocation`	Caller
`Ping`	Caller, Callee

Demo: Build a server… in Go

Further considerations

- Thread safety 🤔
- Error handling
- Groups
- Client-to-server streaming
- Cancellation
- Frame inter-leaving

Messaging



JSON ping

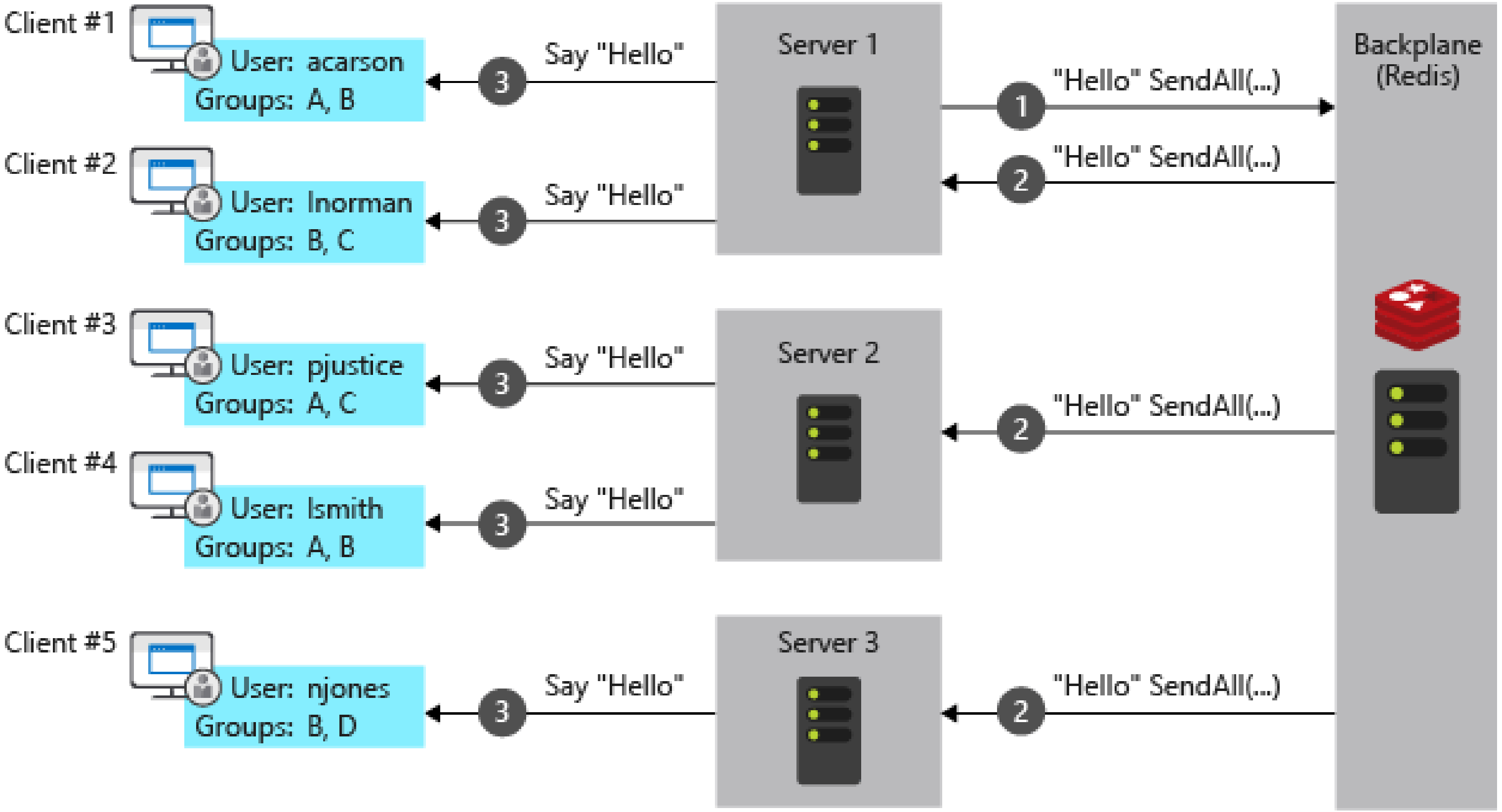
```
{ "type": 6 }
```

JSON non-blocking invocation

```
{
    "type": 1,
    "target": "Send",
    "arguments": [ 42, "Test Message" ]
}
```

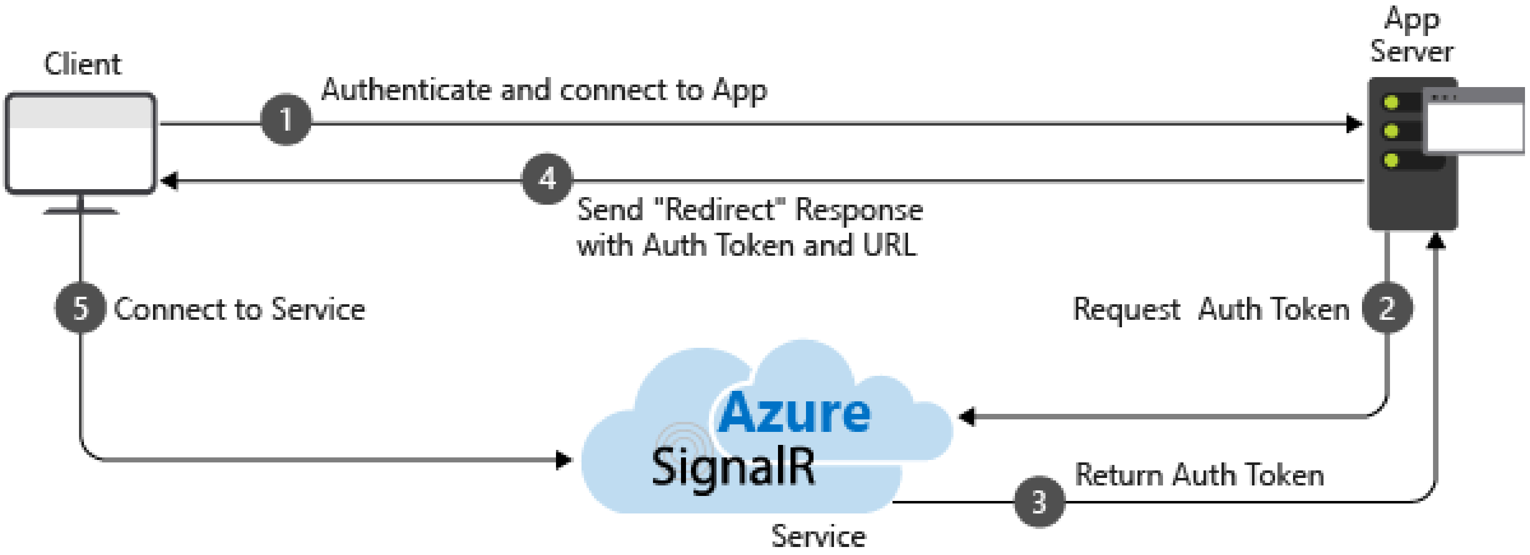
Scale-out

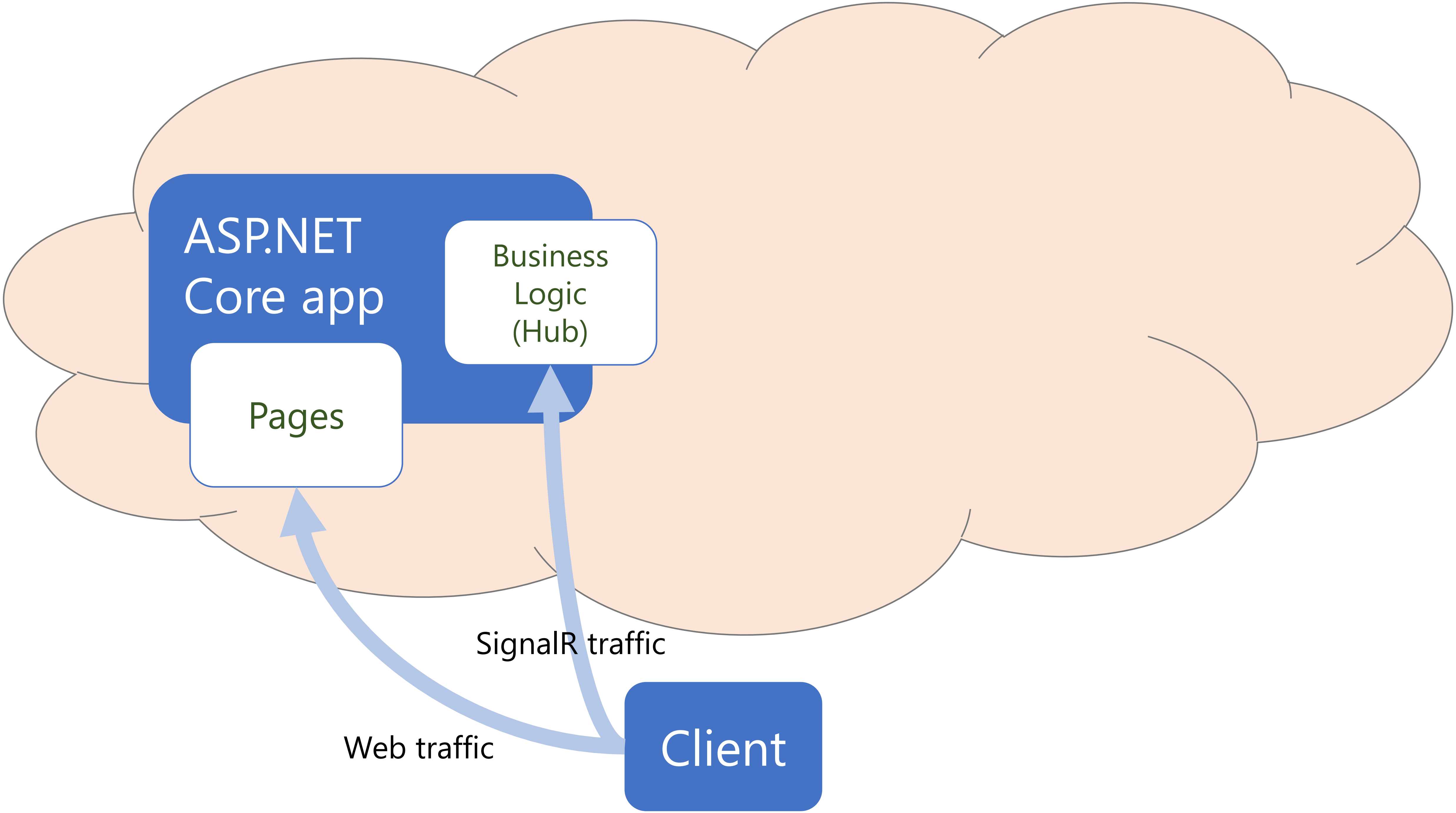
- https://docs.microsoft.com/en-us/aspnet/core/signalr/scale

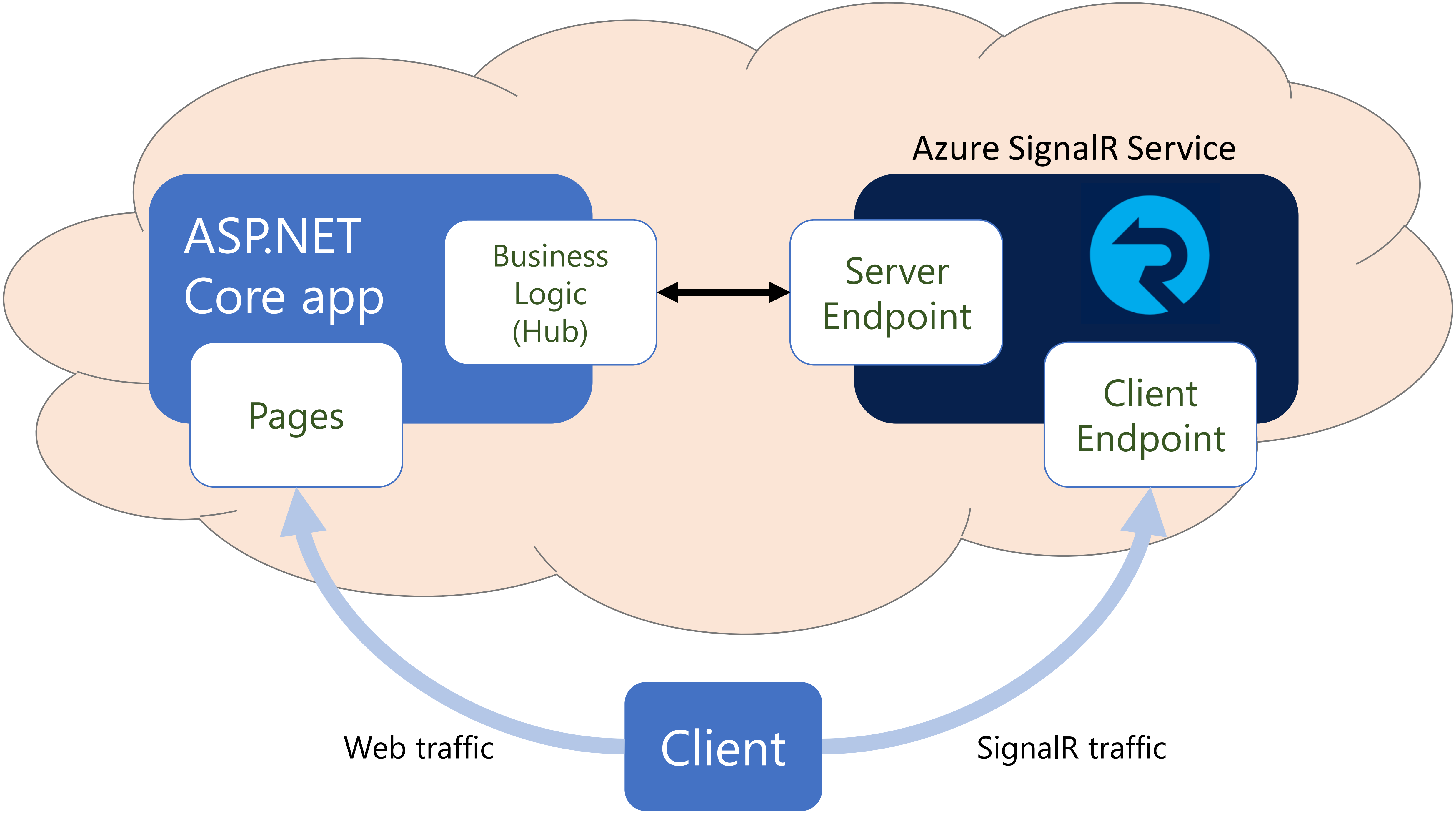


Azure SignalR Service

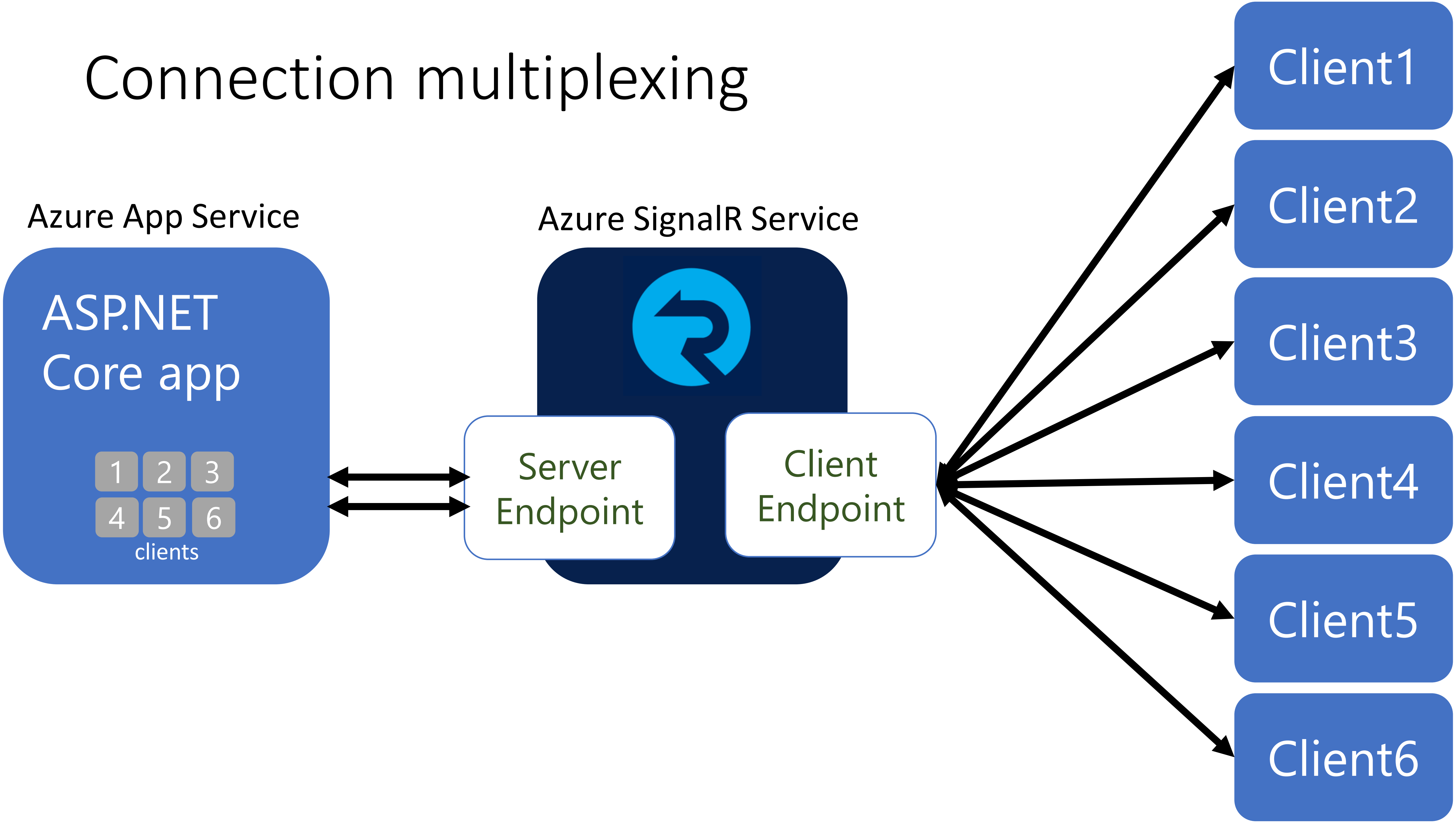
- Provides SignalR scale-out as a service
- Offloads SignalR connection handling to the service
- Relays client traffic to/from your app over few WebSockets
- https://github.com/Azure/azure-signalr/blob/dev/specs/ServiceProtocol.md

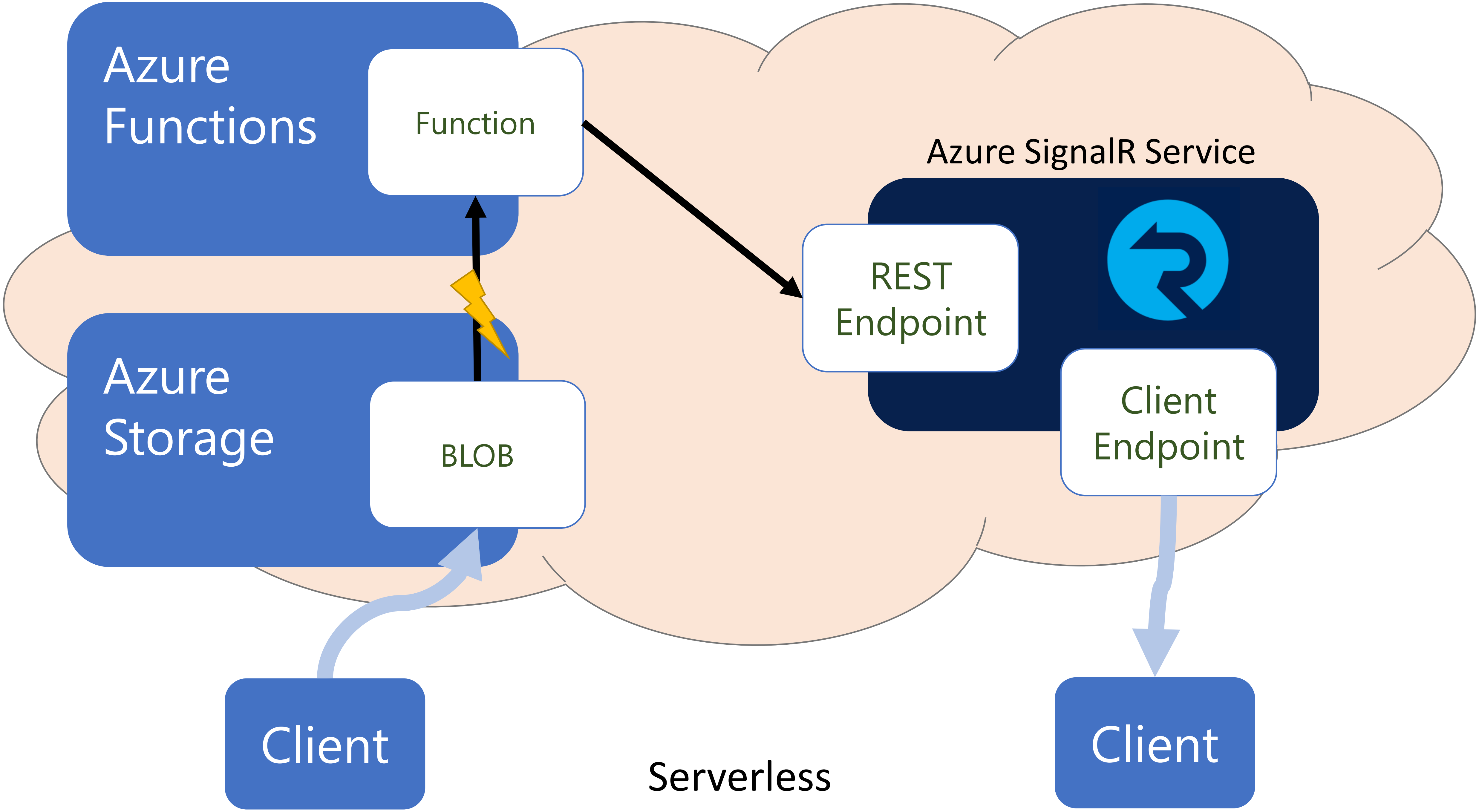



ASP.NET Core app
Business Logic (Hub)
Pages
SignalR traffic
Web traffic
Client


Azure SignalR Service
ASP.NET Core app
Business Logic (Hub)
Server Endpoint
Pages
Client Endpoint
Web traffic
Client
SignalR traffic

Connection multiplexing




Azure Functions
Function
Azure SignalR Service
REST Endpoint
Client Endpoint
Azure Storage
BLOB
Client
Serverless
Client

Demo: Build a SignalR service

Questions

@davidfowl

@damianedwards