
SOAP & XML • gRPC & Protobuf • WebSocket

Microservice kommunikáció – áttekintés

Hagyományos és modern kommunikációs technológiák összehasonlítása

Miért létezik ennyi kommunikációs technológia?

Miért nem elég „HTTP + JSON” (REST) mindenre?

- különböző igények (és problémák)
- különböző korszakok
- különböző kompromisszumok

SOAP: történeti és üzleti háttér

- **SOAP (Simple Object Access Protocol)**
- 2000-es évek eleje
- enterprise integráció
- platform- és nyelvfüggetlenség
- szigorú szabványok

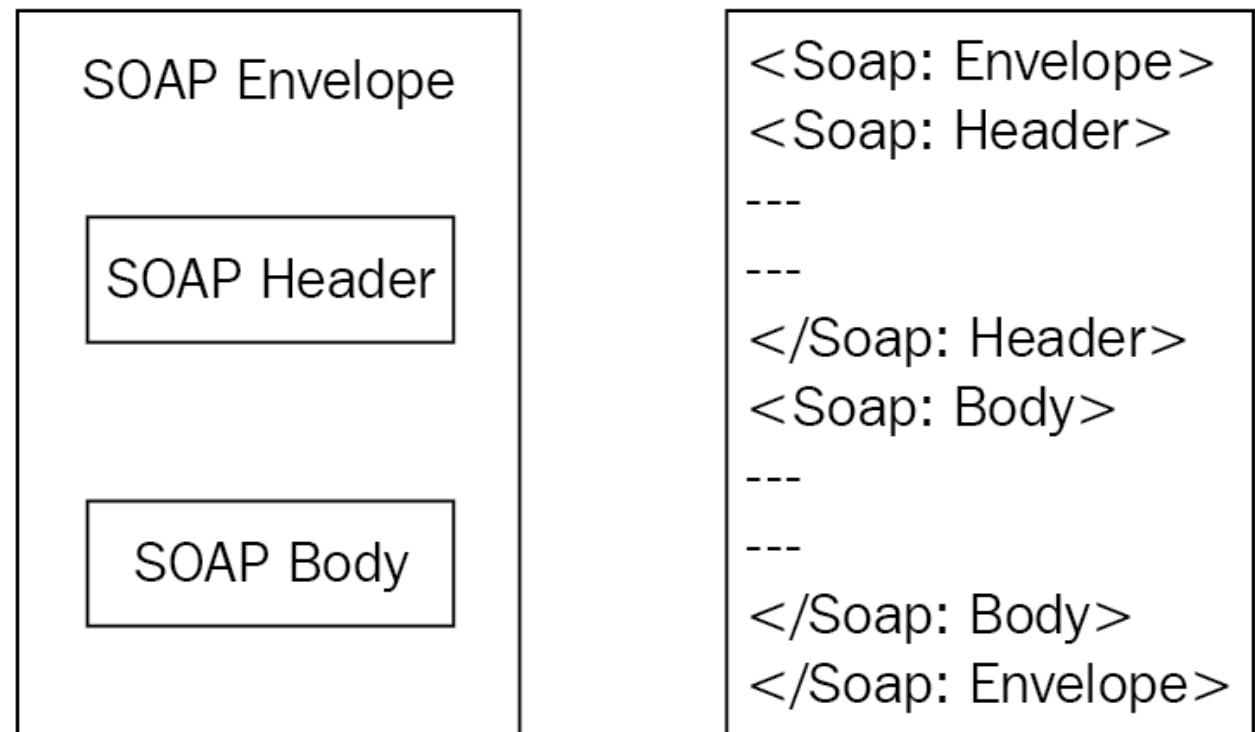
SOAP építőelemei: WSDL és XSD

- **WSDL** – szolgáltatások leírása
- **XSD** – XML adatstruktúrák leírása
- **erős típusosság** jellemzi
- automatikus **kliensgenerálás**

SOAP üzenet felépítése

- **XML** alapú
- **strukturált** felépítés
- Envelope / Header / Body
- bőbeszédű formátum

SOAP Message Structure



SOAP előnyök és hátrányok

- **Előnyök**
- szigorú szerződés (WSDL, XSD)
- stabilitás
- enterprise támogatás
- **Hátrányok**
- Bőbeszédű (verbose) XML
- komplexitás
- lassabb adatfeldolgozás és kommunikáció

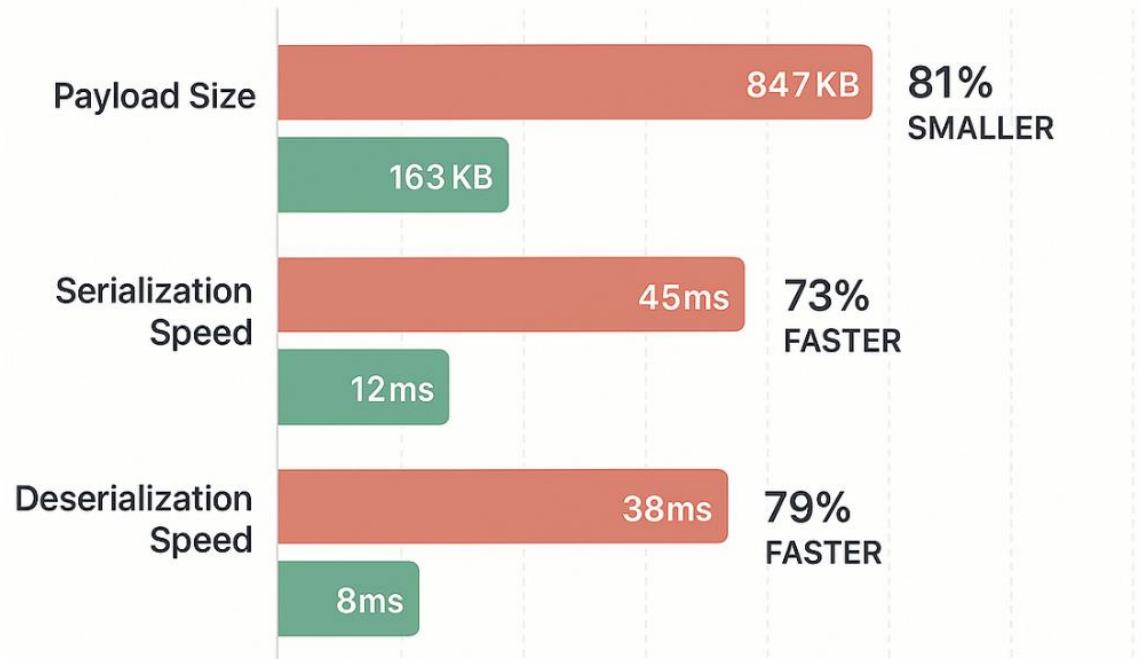
gRPC: mi ez valójában?

- Google által fejlesztett **RPC** framework
- RPC = Remote Procedure Call
- **HTTP/2** felett működik
- **bináris** kommunikáció
- erős **típusosság**

Protobuf: miért bináris?

- gRPC alapja
- strukturált bináris adatformátum
- kisebb üzenetméret
- gyorsabb feldolgozás
- Szigorú séma

JSON vs Protobuf



gRPC kommunikációs módok

HTTP/2 felett működik, és az alábbi módokat támogatja:

- **Unary** (1 kérés – 1 válasz)
- **Server streaming**
- **Client streaming**
- **Bidirectional streaming**

WebSocket: miért kellett?

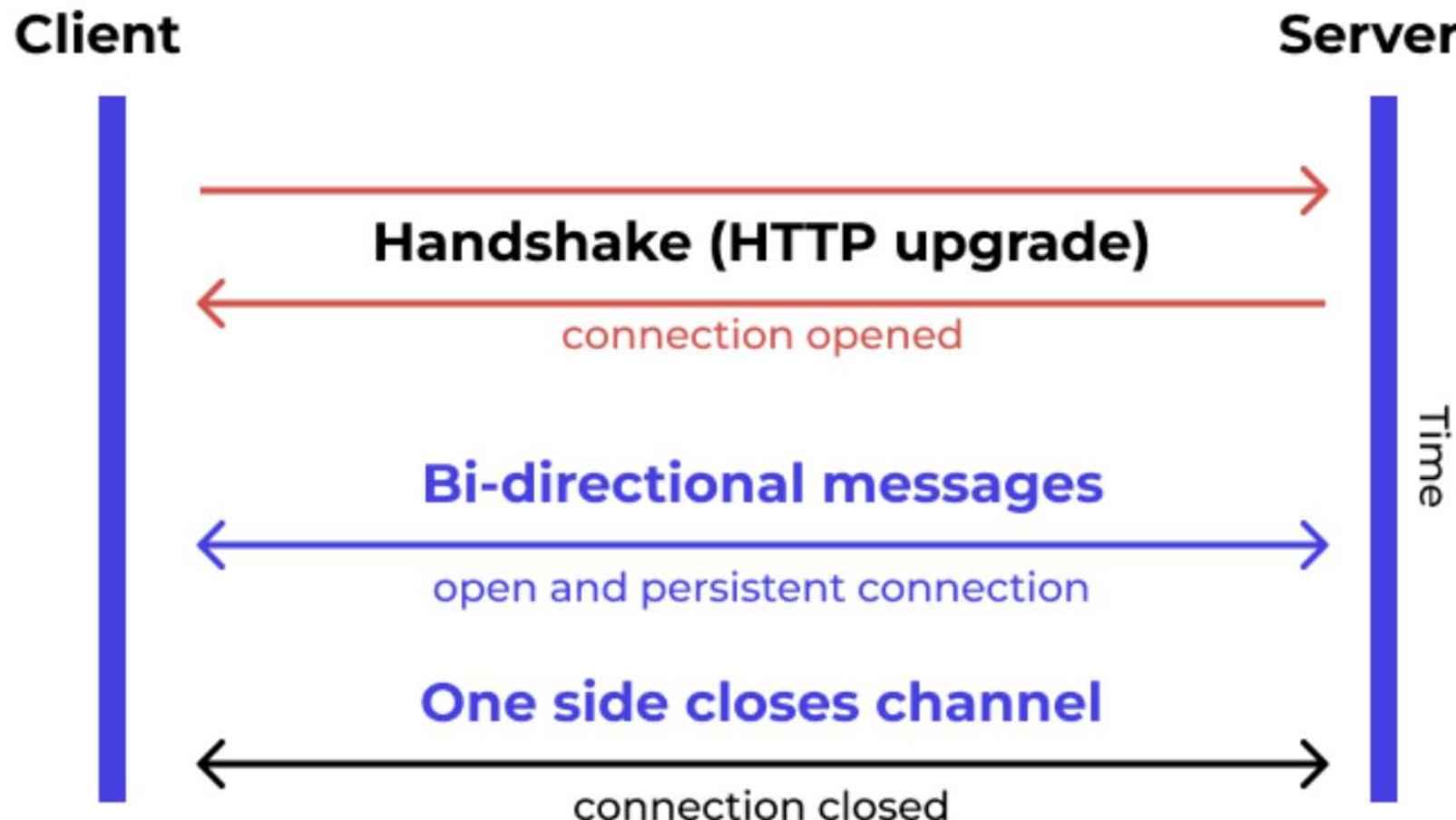
A HTTP korlátai real-time esetén

- HTTP **stateless**
- kérés–válasz modell
- **polling** → pazarló
- nincs natív szerver → kliens **push**

WebSocket megoldás

- **egyszeri** kapcsolatfelépítés
- **állandó, kétirányú** kapcsolat
- **alacsony** késleltetés (real-time)

WebSocket működése



WebSocket előnyök és korlátok

- **Előny**
 - real-time kétirányú kommunikáció
 - alacsony késleltetés (low latency)
- **Hátrány**
 - Állandó (stateful) kapcsolat
 - skálázási és üzemeltetési nehézségek

Microservice: mi ez egyáltalán?

Microservice = architektúra, nem technológia és nem protokoll

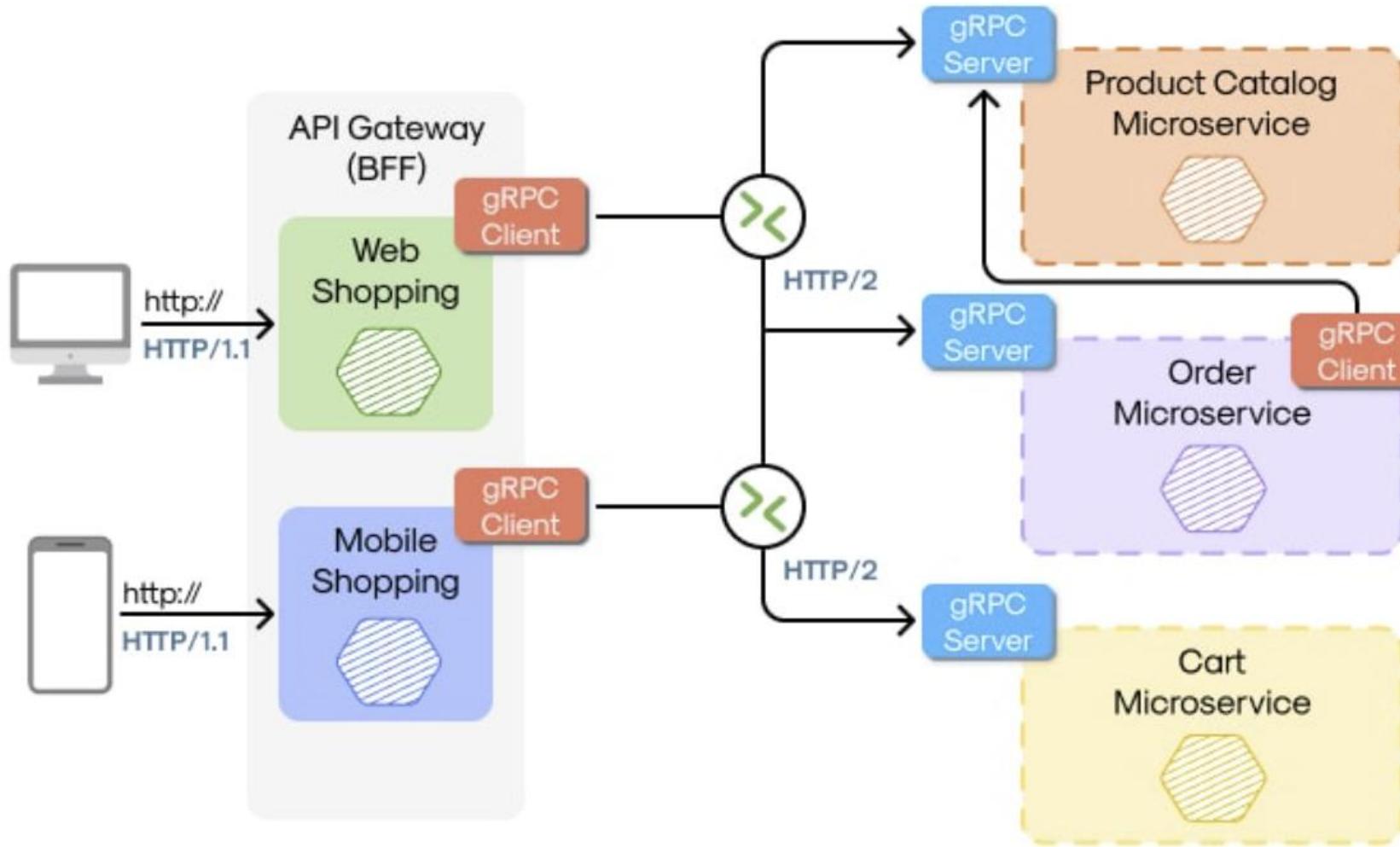
- **Sok kis**, jól körülhatárolt **szolgáltatás**
- **Önállóan** fejleszthető és telepíthető szolgáltatások
- Saját adatkezeléssel (business domain) rendelkezik
- Hálózaton kommunikálnak egymással

Kommunikáció microservice architektúrában

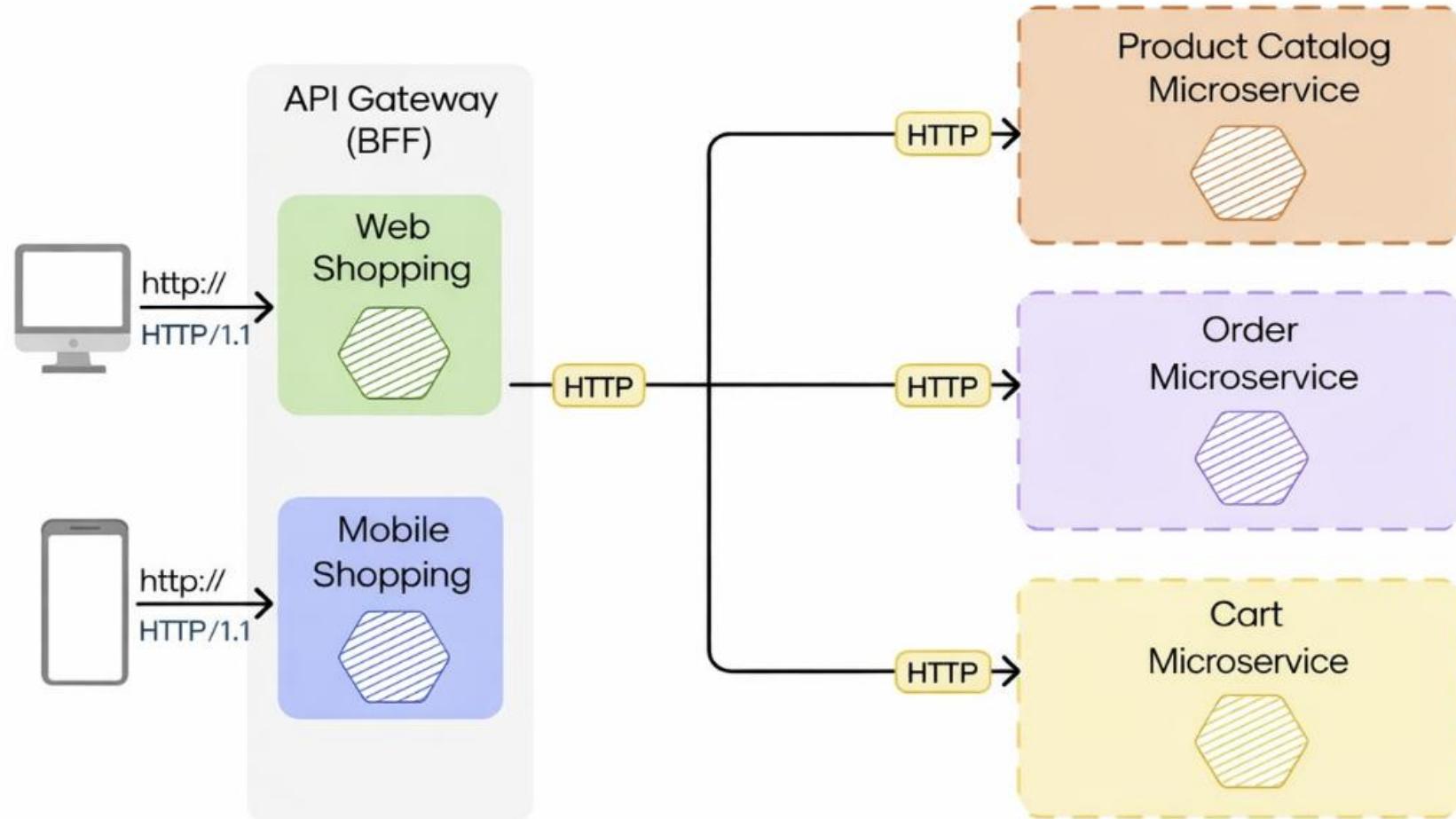
Egy modern rendszerben több protokoll együtt él

- REST – külső API
- böngészők, mobil kliensek
- publikus, („emberbarát”) szöveges
- HTTP/JSON
- **gRPC – belső kommunikáció**
- microservice ↔ microservice
- nagy teljesítmény
- bináris, erősen típusos
- **WebSocket – real-time**
- Böngészők, mobil kliensek
- események, push értesítések
- folyamatos kétirányú kapcsolat
- alacsony késleltetés

gRPC in Microservices Communication



HTTP in Microservices Communication



Összefoglalás

Technológia	Tipikus cél
SOAP	legacy enterprise rendszerek integrációja
gRPC	belső microservice kommunikáció
WebSocket	real-time web kommunikáció
REST	publikus Web API