IoT rendszerek kommunikációs megoldásai
vitmav22

Mérési adatok szállítása az interneten

MQTT

„MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol.

It was designed as an extremely lightweight publish/subscribe messaging transport.”

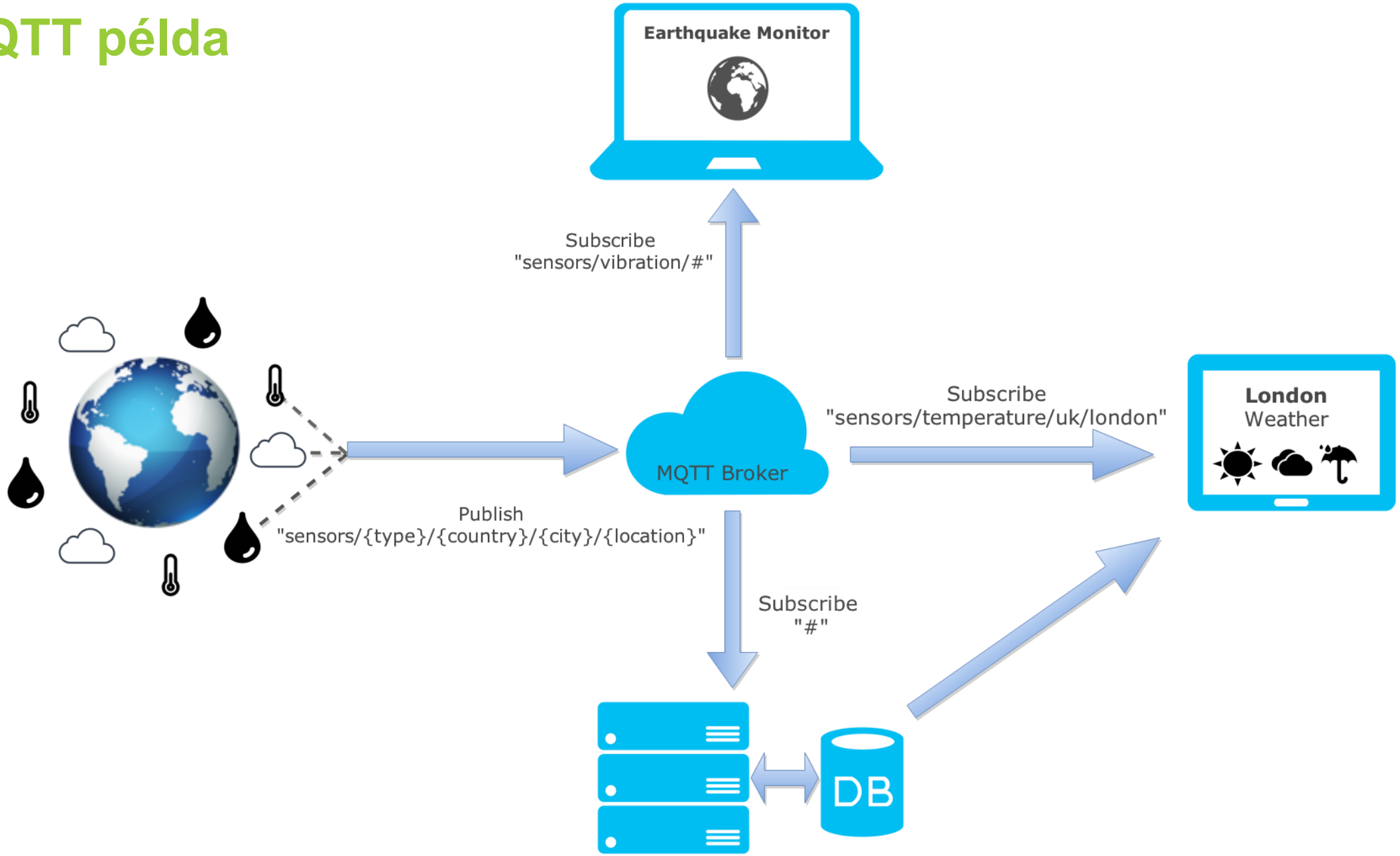
http://mqtt.org/

MQTT

- MQTT = Message Queueing Telemetry Transport
- Nyílt, ingyenes
- publish/subscribe (one-to-many) modell
  - A küldő félnek semmit sem kell tudnia a vevő(k)ről
- Ideális kényszerekkel terhelt hálózatoknál
  - Pl. kis sávszélesség, nagy késleltetés, csomagvesztés, instabil kapcsolatok...
- Szűkös erőforrásokkal rendelkező eszközök számára
  - Kevés memória, kis teljesítményű processzor

- Cél: Megbízható üzenetküldés a lehető legegyszerűbb üzenet-formátumban

MQTT példa


Earthquake Monitor
Subscribe
"sensors/vibration/#"
MQTT Broker
Subscribe
"sensors/temperature/uk/london"
London
Weather
Publish
"sensors/{type}/{country}/{city}/{location}"
Subscribe
"#"
DB

MQTT

- Publish-subscribe
  - A kliensek (*client*) feliratkoznak (*subscribe*) ún. *topic*-okra. Felratkozás után a topic-ra publikált (*publish*) összes üzenetet megkapják.
- Szolgáltatási szintek
  - 3 QoS szint definiált. Magasabb szolgáltatásminőség nagyobb megbízhatóságú üzenet célba juttatást valósít meg – ennek ára a nagyobb sávszélesség és/vagy késleltetés.
- Megőrzött üzenetek
  - A szerver meg is tartja az utoljára elküldött üzenetet. Egy új feliratkozó esetén egyből elküldi az utolsó üzenetet is a kliensek.

MQTT

- Clean session flag
  - Amikor egy MQTT kliens csatlakozik a szerverhez, beállíthat egy ún. *clean-session* flag-et. Ha ez a beállítás „1”-es, a kliens összes feliratkozása törlődik, ha az eszköz lekapcsolódik a szerverről.
  - Ellenkező esetben a kliens előfizetése továbbra is élő marad mindaddig, amíg az vissza nem kapcsolódik. Eközben a magas QoS beállítás esetén érkezett összes üzenet eltárolódik, majd újracsatlakozás után elküldésre kerül.
- Végrendelet (Will)
  - Kérésre a szerver egy üzenetet küld a kliens által előre meghatározott topic-ba, ha a kliens váratlanul lecsatlakozik. (Pl. riasztaás, ha egy érzékelő lecsatlakozik a hálózatról)

MQTT és HTTP összehasonlítás

	MQTT	HTTP
Design orientation	Data centric	Document centric
Pattern	Publish/subscribe	Request/response
Complexity	Simple	More complex
Message size	Small, with a compact binary header just two bytes in size	Larger, partly because status detail is text-based
Service levels	Three quality of service settings	All messages get the same level of service
Extra libraries	Libraries for C (30 KB) and Java (100 KB)	Depends on the application (JSON, XML), but typically not small
Data distribution	Supports 1 to zero, 1 to 1, and 1 to n	1 to 1 only

MQTT kliens

- Egy MQTT kliens (aka kliens alkalmazás)...
  - **adatokat gyűjt** egy telemetriai eszközről, csatlakozik egy MQTT szerverhez, majd egy topic string segítségével **publikálja** az információt, ÉS/VAGY
  - **feliratkozik** egy topic-ra, ahol **fogadja** a publikált adatokat, ÉS/VAGY
  - parancsok segítségével **vezérli** a telemetriai eszköz(öke)t.

- Pl: Eclipe Paho MQTT kliens: http://www.eclipse.org/paho/
  - C és Java alapú implementáció

MQTT bróker

- **MQTT bróker** = Egy szerver, amely implementálja az MQTT protokollt.
- Feladata: Közvetít az MQTT kliensek közötti kommunikációban.

- Pl: Mosquitto
  - Egyszerű, nyílt, ingyenes implementáció (MQTT v3.1): http://mosquitto.org

mosquitto.org

Home Downloads Documentation Bugs / Support

Mosquitto

An Open Source MQTT v3.1/v3.1.1 Broker

eclipse

Eclipse Mosquitto™ is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 3.1 and 3.1.1. MQTT provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for "Internet of Things" messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers like the Arduino.

Mosquitto is an iot.eclipse.org project

For the final time...

18

Thursday, March 9th, 2017 | Releases

This guy arrived on Tuesday, two weeks early and weighing 9lb 6oz / 4.26kg. Apologies if I'm a bit out of touch for a while.

Pages

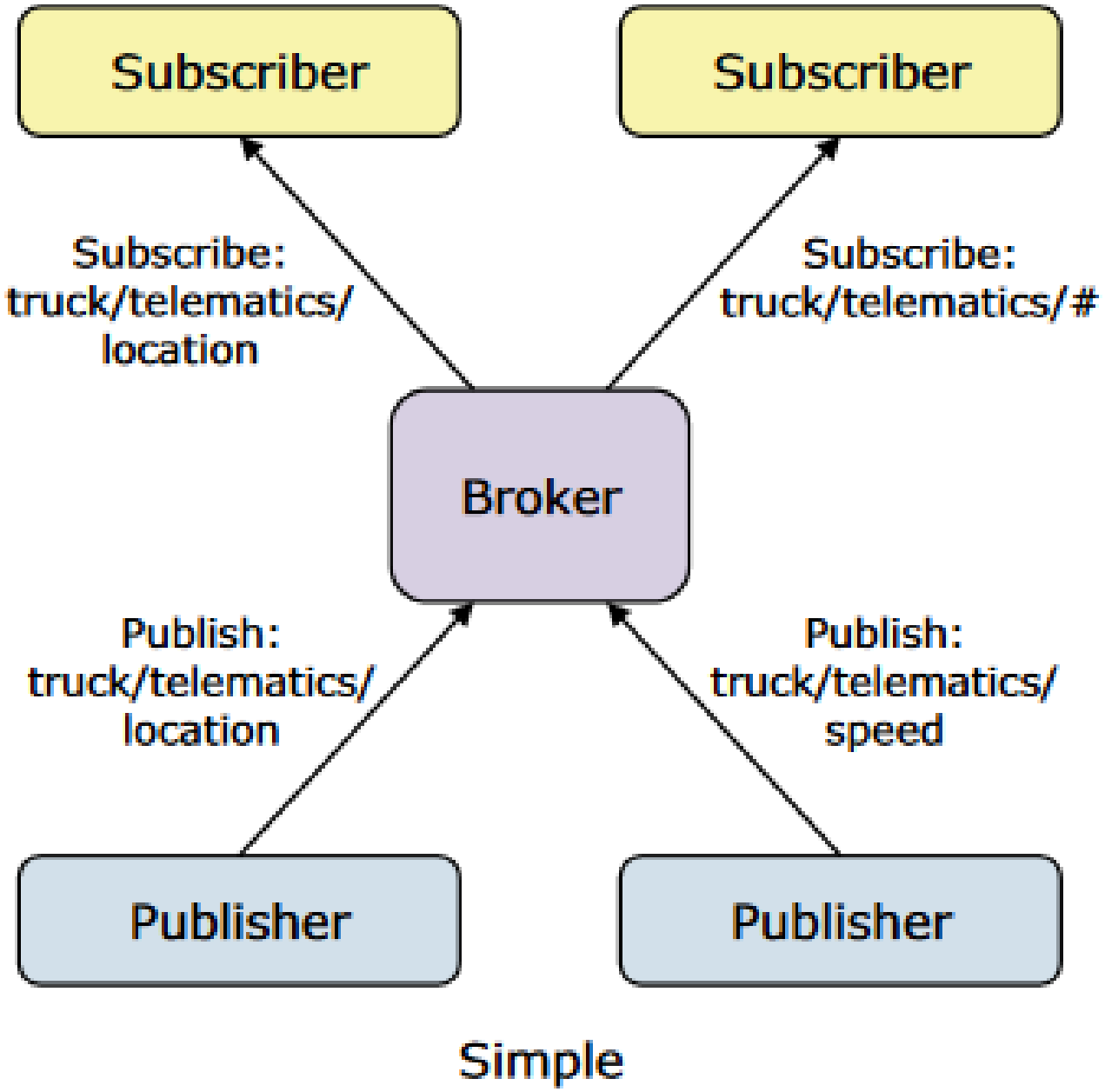
Bugs / Support
Documentation
Python
Downloads

Links

Mosquitto test server
MQTT community
mqtt2cosm
Paho

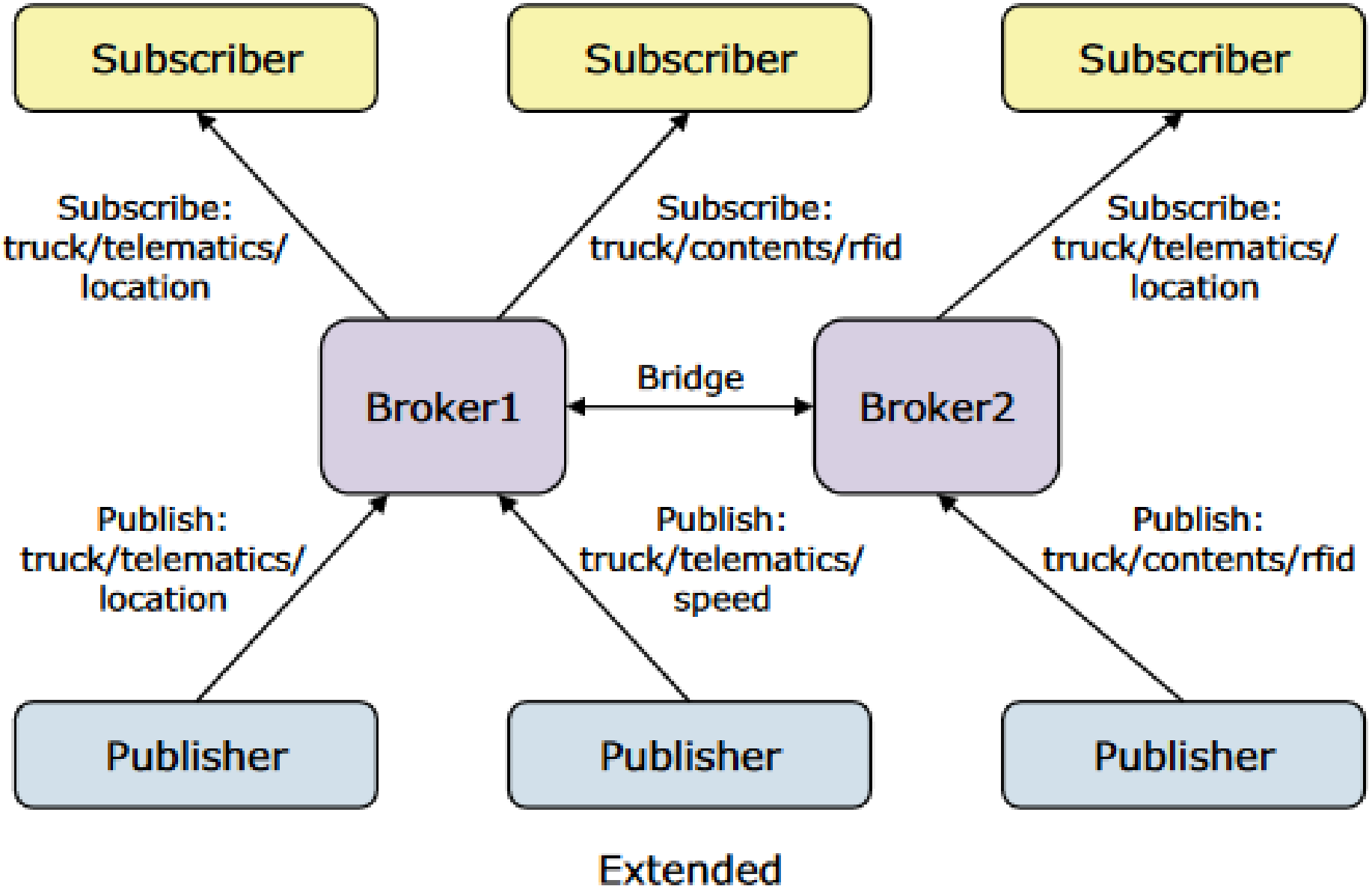
Donate

MQTT üzenetküldés



Simple

MQTT üzenetküldés



MQTT topikok, fák, sztringek

- topic
  - Az üzenetek osztályozását szolgálja, definiálja az üzenet tartalmát.
  - Pl: „truck”
- topic-tree
  - Tipikusan a topic-ok hierarchikusan szervezettek.
  - A „/” karakter használatával hozhatók létre al-topikok.
  - Pl: „truck/contents”, „truck/contents/rfid”
- topic-string
  - Egy karakterlánc, amely meghatározza az üzenet topikját. (wildcard használható!)
  - Pl: „truck/contents/#”, „truck/+”

MQTT programozás

- Client ID
  - 23 bájtos egyedi(!) sztring a kliens azonosítására.
- Megőrzött üzenet(ek)
  - *retention attribute* = true/false
  - Megőrzött üzenetek elküldése: **legalább egyszer** (QoS=1), **pontosan egyszer** (QoS=2), vagy **legfeljebb egyszer** (QoS=0).

Példa: mosquitto használata

- Publikálás
  - Pl: mosquitto_pub -t samples/topic01 –l
  - Pl: mosquitto_pub -h fake.mqtt.server -t samples/topic01 -i localGhost –l
- Feliratkozás
  - Pl: mosquitto_sub -t samples/topic01
  - Pl: mosquitto_sub -h fake.mqtt.server -t samples/topic01

További info: http://www.redbooks.ibm.com/redbooks/pdfs/sg248054.pdf

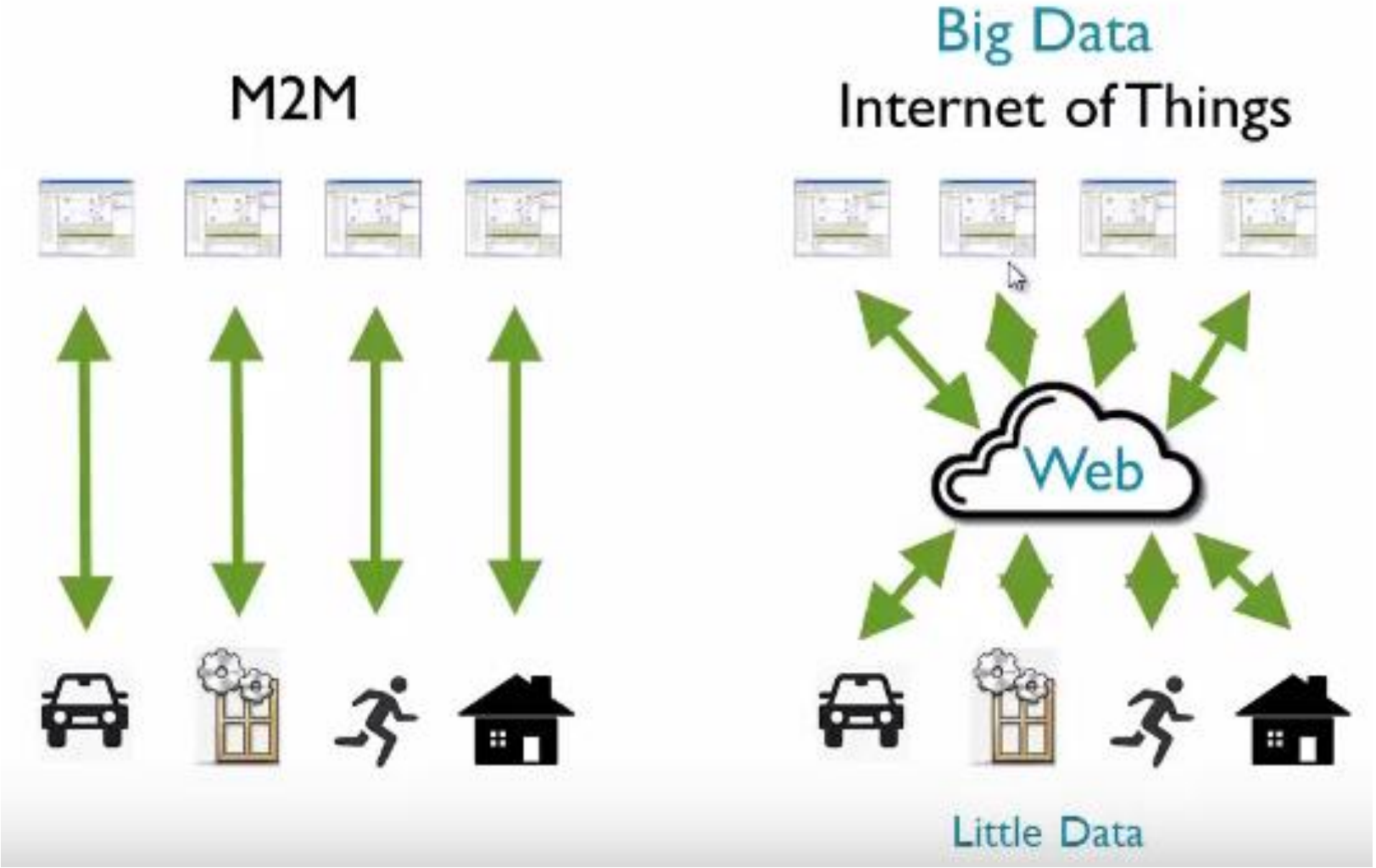
CoAP

“The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things.

The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation.”

http://coap.technology/

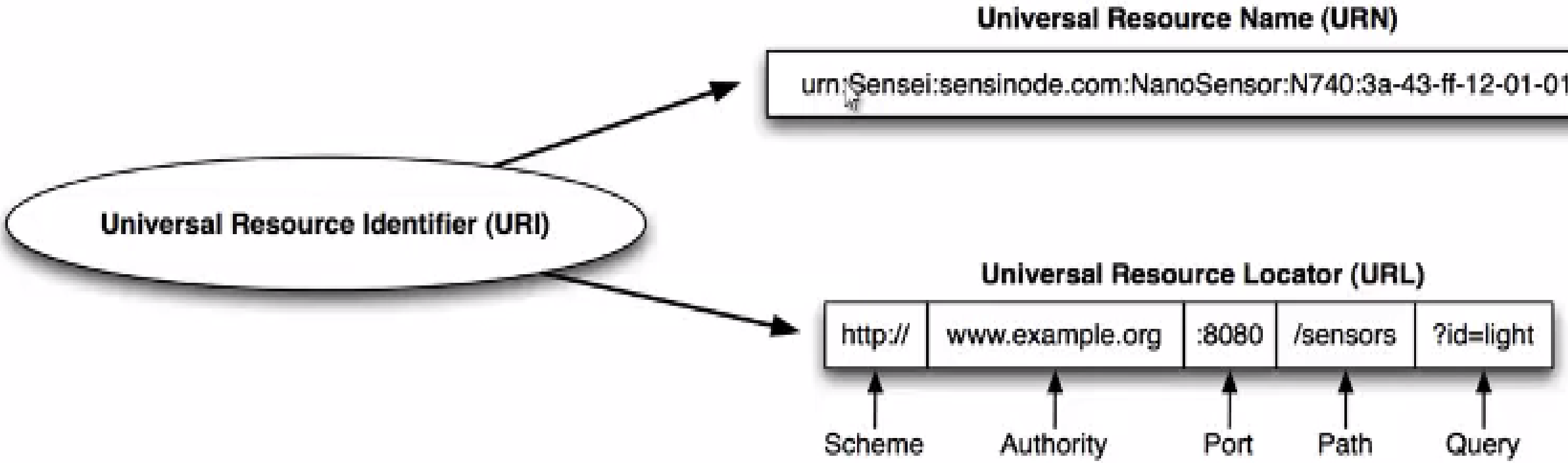
M2M kommunikációtól az IoT felé



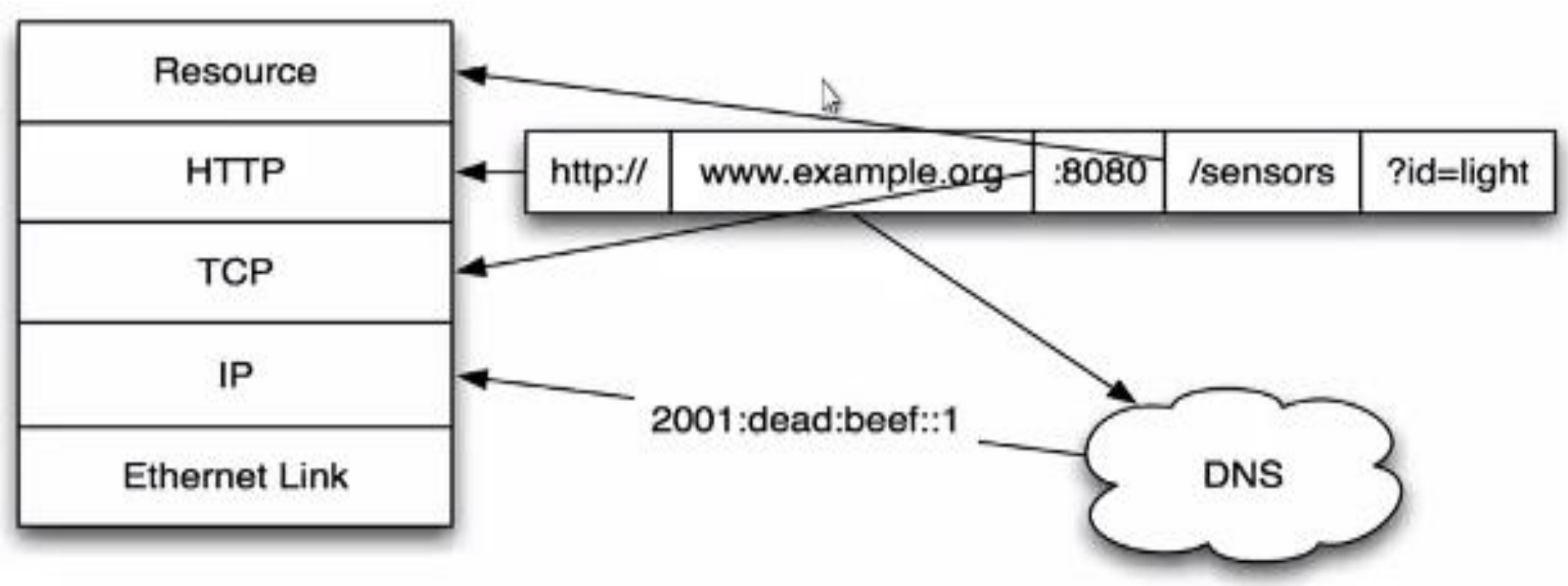
CoAP

- CoAP = RFC 7252 Constrained Application Protocol
- IETF ajánlás (CoRE Workgroup – Constrained RESTfil Environment)
- Aka: „CoAP: The Web of Things Protocol”
- Alkalmazás réteg-beli protokoll: web (dokumentum) átviteli protokoll kis erőforrású (alacsony erőforrású) eszközök részére, veszteséges átvitelű („lossy”) hálózatok felett.
- REST modell: A szerverek erőforrásokat tesznek elérhetővé URL-ek használatával. A kliensek elérhetik az erőforrásokat metódusok hívásával: GET, PUT, POST, és DELETE.

REST, URI/URN/URL



REST, URI/URN/URL

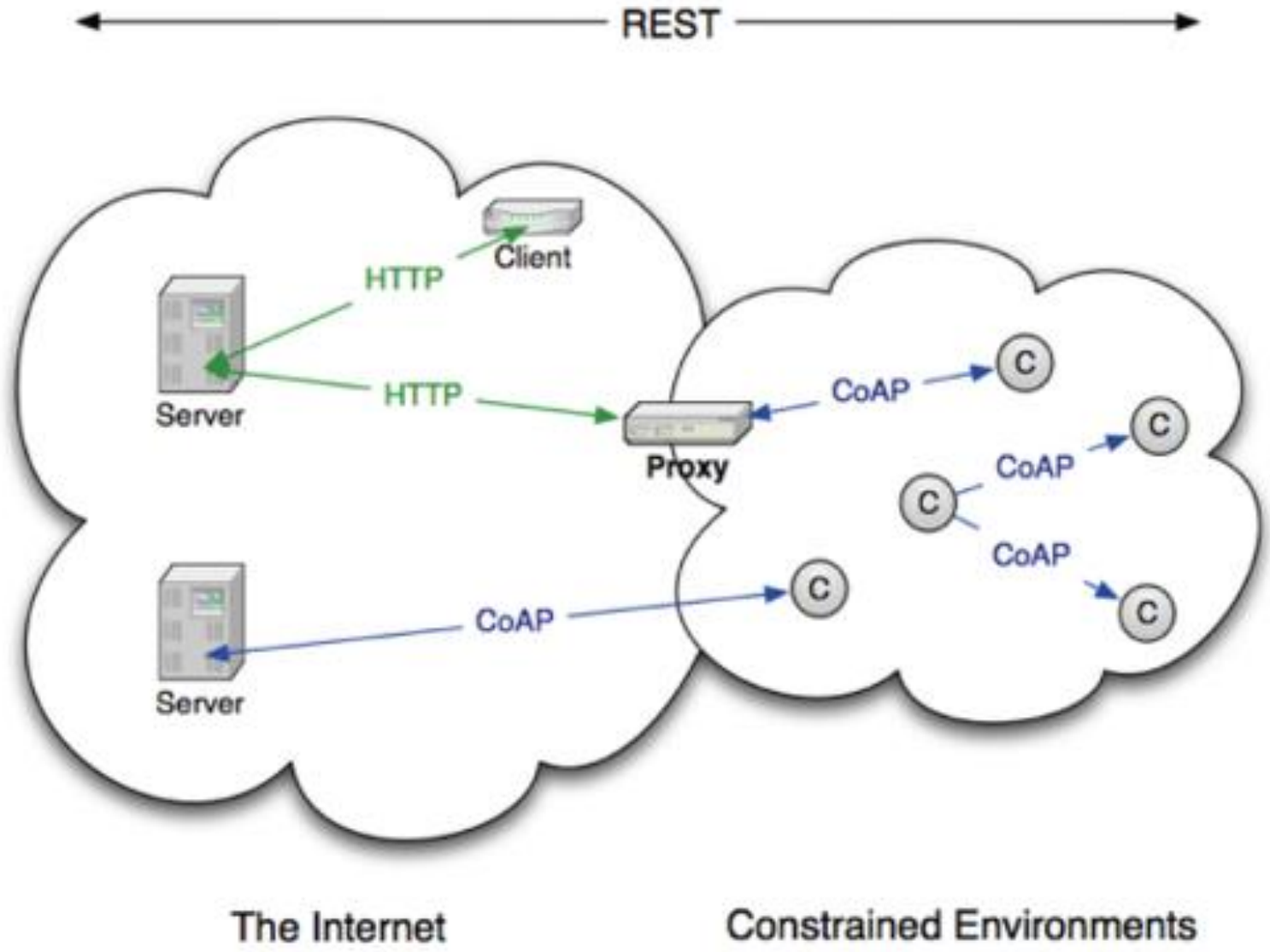


CoAP kontra HTTP

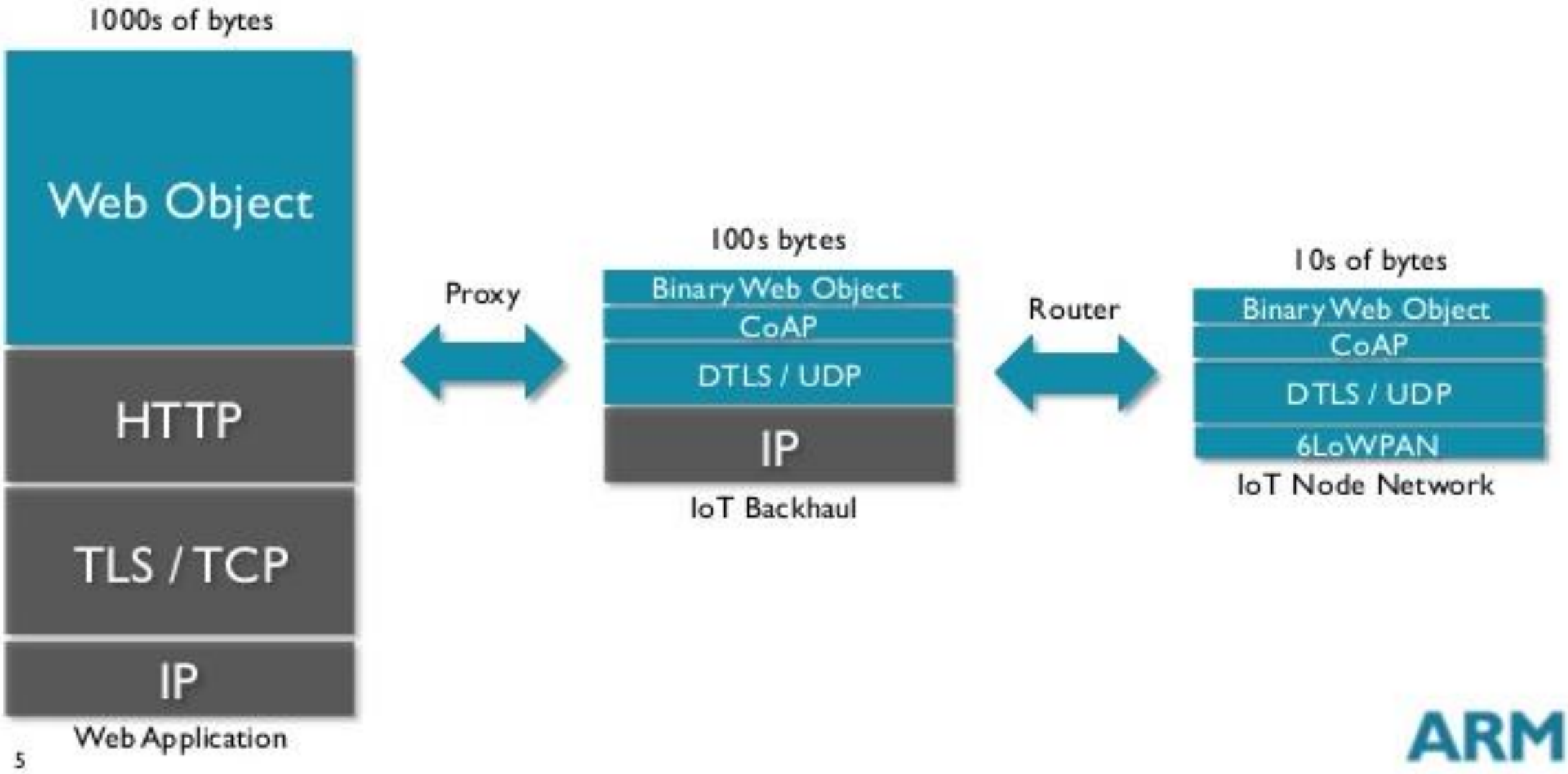
- Az alkalmazási rétegben tipikusan HTTP-t használunk web szolgáltatások eléréséhez.
- A HTTP...
  - komplex megoldás,
  - nagy energiafogyasztású,
  - kis adatsebességű.
- A CoAP...
  - HTTP-hez hasonlóan hálózat központú,
  - UDP-t használ (a komplex TCP helyett), optimalizált datagram mérettel.
  - REST architektúrán alapul,
  - lehetővé teszi az IP multicast-ot (IoT csoportos kommunikáció)

CoAP

- Beágyazott web transzport protokoll (coap://)
- Kompakt fejléc (4 byte)
- UDP, SMS támogatás (TCP is)
- Erős DTLS biztonság
- Aszinkron működés
- Multicast támogatás



CoAP


1000s of bytes
Web Object
HTTP
TLS / TCP
IP
Web Application
5
Proxy
100s bytes
Binary Web Object
CoAP
DTLS / UDP
IP
IoT Backhaul
Router
10s of bytes
Binary Web Object
CoAP
DTLS / UDP
6LoWPAN
IoT Node Network

CoAP felépítése

- A HTTP kliens-szerver modelljéhez hasonlóan a CoAP is interaktív.
- 2 rétegű modell: (1) Message layer + (2) Request/response layer

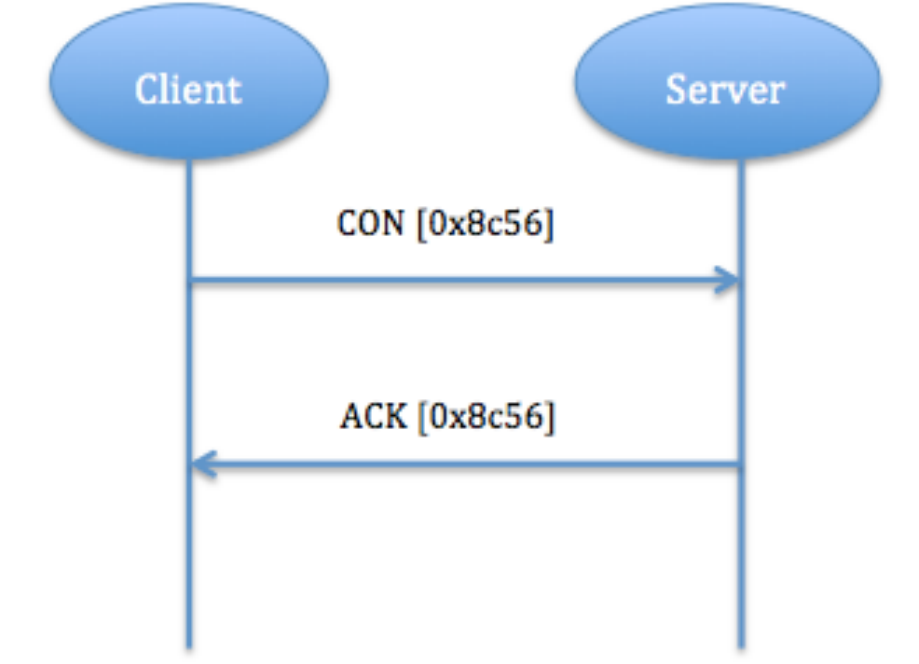
Message layer

- 4 féle üzenet: **CON** (confirmable), **NON** (non-confirmable), **ACK** (acknowledge), **RST** (reset)

CoAP Message réteg

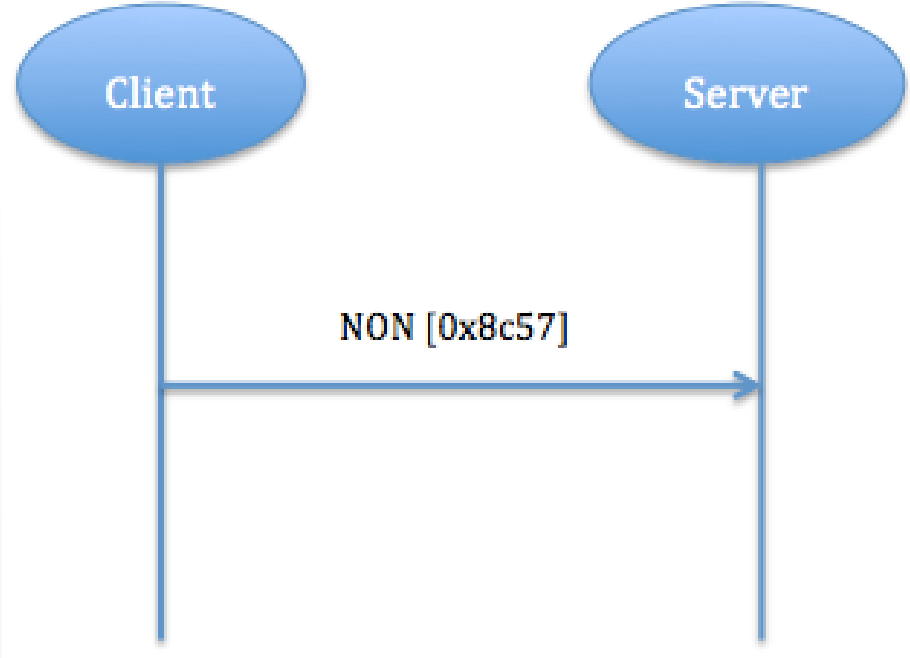
Megbízható átvitel

- Folyamatos újraküldés pozitív ACK-ig
- Ha a vevő nem tudja feldolgozni az üzenetet, RST-t küld vissza



Nem megbízható átvitel

- Nincs nyugtázás
- Ha a vevő nem tudja feldolgozni az üzenetet, RST-t küld vissza



- „The most secure messaging standard”

  „Battle-tested. Independent. Privacy-focused.”

„XMPP is the open standard for messaging and presence”

https://xmpp.org/

XMPP

Extensible Messaging and Presence Protocol

XMPP (aka Jabber)

- XMPP = Extensible Messaging and Presence Protocol
- Eredetileg a zárt IM (Instant Messaging) szolgáltatások alternatívájaként jött létre, egy nyílt, decentralizált megoldásként (~2002-ben). Probléma volt, hogy a számos IM szolgáltatás nem tudott együttműködni egymással.
- Lassan átalakult publish/subscribe megoldássá
- IETF ajánlás (RFC3920, RFC3921)
- XML alapú
- Gyors (közel valósidejű) üzenetküldés több eszköz között
- Decentralizált
  - Az email-hez hasonlóan bárki üzemeltethet XMPP szervert
- Pl a Google Talk is ezt használja

XMPP

- Kommunikáció
  - Request/response
  - Aszinkron üzenetküldés
  - Publish/Subscribe
- Skálázhatóság, címzés
  - **Global Identity** – egyedi, globális XMPP címek
  - **Kliens-szerver** modell: Egy kliens a szerver segítségével kommunikálhat a többi klienssel.
  - **Federation** = peer-szerverek összekapcsolódhatnak (doménen belüli kommunikáció elősegítésére).

XMPP kontra MQTT

XMPP

- Üzenetformátum definiálható, strukturált adatok nyerhetőek ki az eszközökből.
- Nincsenek QoS szintek.
- Jól skálázódik az eszközök számával.
- Egyedi eszköz azonosítók.

MQTT

- Nem definiál üzenetformátumot.
- 3 QoS szint.
- Nem skálázódik olyan jól.
- Egyedi azonosítók elkülönülten, a bróker implementációkban.

XMPP kontra MQTT

- XMPP a jó megoldás, ha...
  - strukturált adatokat kezelünk,
  - az eszközeink nem memória limitáltak.
- MQTT a jó megoldás, ha..
  - M2M kommunikációban.

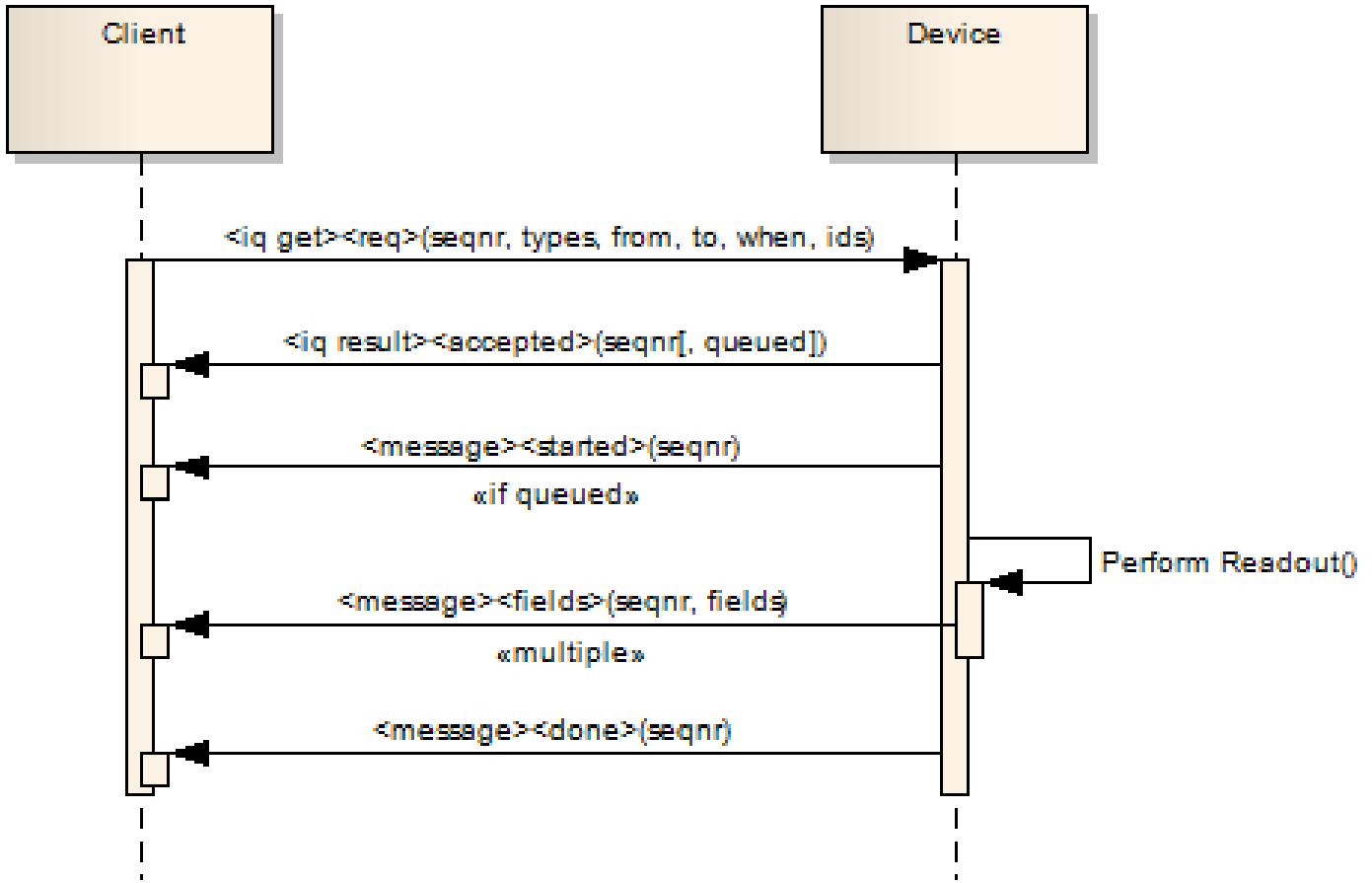
XMPP IoT

- XMPP-IoT = lightweight XMPP
- A probléma hasonló: Számtalan felhő-alapú IoT szolgáltatás ígéri, hogy az IoT eszközeink adatait kezeli. Az XMPP-IoT egy lehetőség ezen rendszerek együttműködésére. (~2014)

XEP-0323: Internet of Things - Sensor Data

- XEP = XMPP Extension Protocol (több mint 370 ilyen van!)
- „This specification provides the common framework for sensor data interchange over XMPP networks.”
- Use case: egy okos mérő (szenzor) adatának kiolvasása

  request/response



Request to another peer

```
<iq type='get'
  from='client@clayster.com/amr'
  to='device@clayster.com'
  id='S0001'>
  <req xmlns='urn:xmpp:iot:sensordata' seqnr='1' momentary='true'/>
</iq>
```

The response confirming the readout

```
<iq type='result'
   from='device@clayster.com'
   to='client@clayster.com/amr'
   id='S0001'>
   <accepted xmlns='urn:xmpp:iot:sensordata' seqnr='1'/>
</iq>
```

The next response with the data contained

```
<message from='device@clayster.com'
  to='client@clayster.com/amr'>
  <fields xmlns='urn:xmpp:iot:sensordata' seqnr='1' done='true'>
    <node nodeId='Device01'>
      <timestamp value='2013-03-07T16:24:30'>
        <numeric name='Temperature' momentary='true' automaticReadout='true'  value='23.4' unit='°C'/>
        <numeric name='load level' momentary='true' automaticReadout='true' value='75' unit='%'/>
      </timestamp>
    </node>
  </fields>
</message>
```

AMQP

Advanced Message Queueing Protocol

- „AMQP is the Internet Protocol for Busines Messaging”
- To become the standard protocol for interoperability between all messaging middleware”

https://www.amqp.org/

AMQP

- AMQP = Advanced Message Queueing Protocol
- Az egész megoldás sorkezelésre (*queueing*) épül.
- Szerverek közötti tranzakciós üzenetek küldése, nagyon figyelve arra, hogy **ne vesszenek el üzenetek**. (*strictly reliable p2p comm*)
  - Pl: banki alkalmazások esetében >>> ezért „business messaging”
- „*binary wire*” protokoll – alacsony szinten, különböző gyártók termékeinek összekapcsolására.
- 5 féle üzenetküldés: *direct*, *fanout*, *topic*, *headers*, *system*