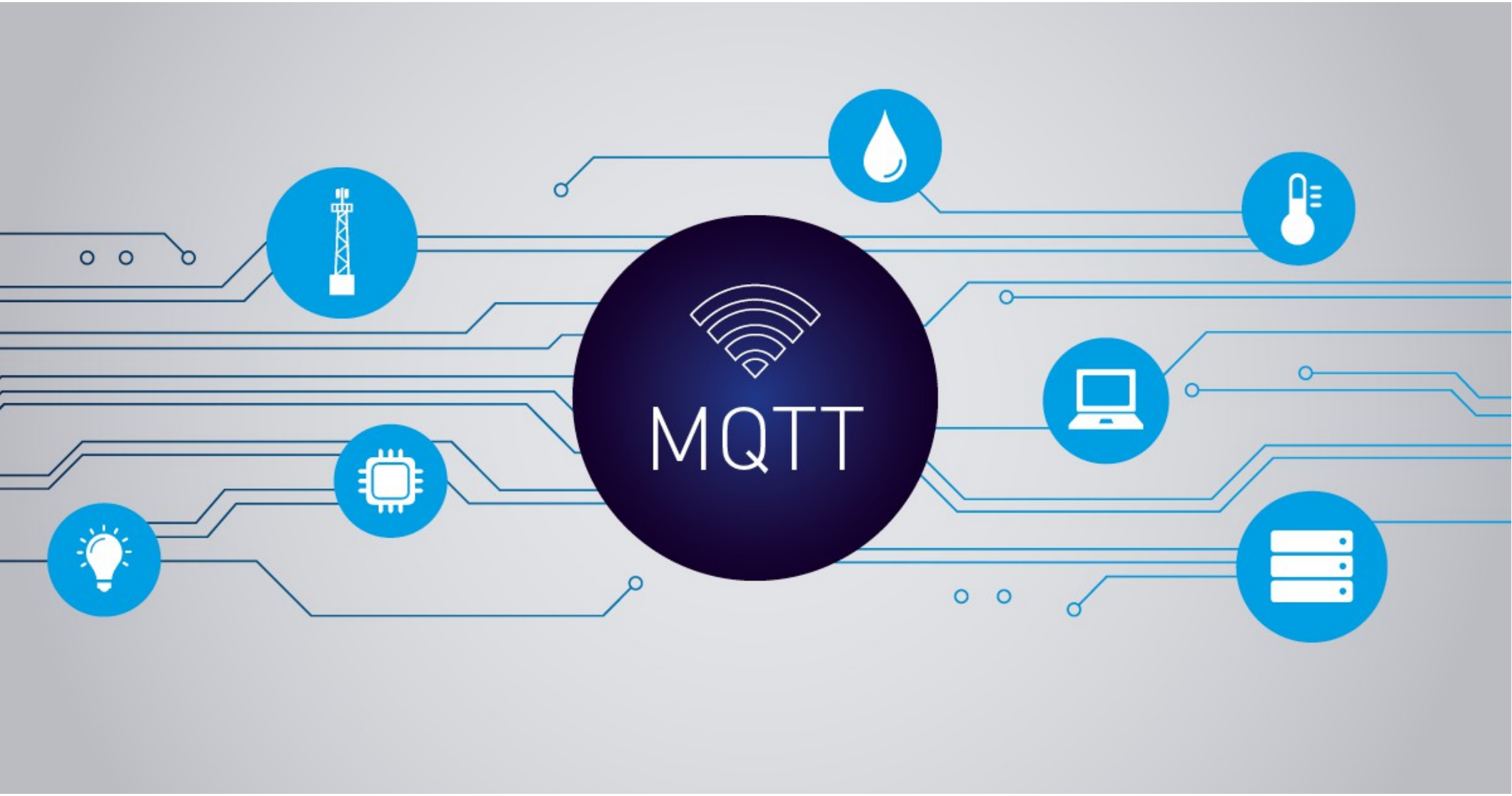

MQTT

MQTT

A practical protocol for the **Internet of Things**

TV Sets
Pacemakers
Ovens
Vehicles
Cows
Smartphones

The Internet is (in) everything

- **vehicles**
- **children**
- **cows**
- **smartphones**
- **ovens**
- **pacemakers**

By the year 2020…

57,000 /sec
new objects connecting

212 BILLION
Total number of available sensor enabled objects

30 BILLION
sensor enabled objects **connected to networks**

Data source: IDC

The world is getting smarter

Smarter Vehicles

- realtime telemetry
- predictive maintenance
- look-ahead alerting
- pay-as-you-drive

Smarter Homes

- energy tracking
- automation
- remote monitoring
- smart appliances

Smarter Logistics

- end-to-end tracking
- theft prevention
- real-time updates
- fleet monitoring

Smarter Healthcare

- smart scales
- in-home monitoring
- assisted living
- physician messaging

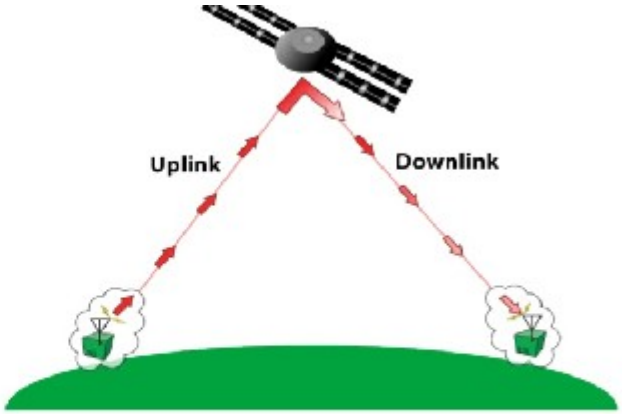
MQTT a lightweight protocol for IoT messaging

- **open** open spec, standard 40+ client implementations
- **lightweight** minimal overhead efficient format tiny clients (kb)
- **reliable** QoS for reliability on unreliable networks
- **simple** 43-page spec connect + publish + subscribe

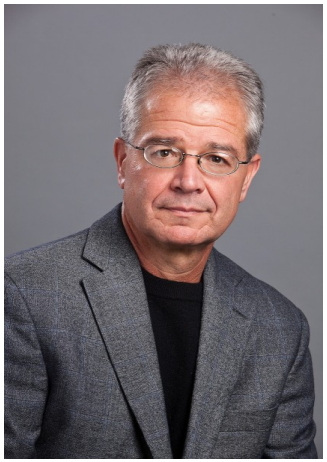
Invented	Published	Eclipse M2M	Standard
EUROTECH IBM	MQTT.ORG	paho	OASIS Advancing open standards for the information society
Late 1990s	Aug 2010	Nov 2011	Sep 2014

MQTT – message queuing telemetry transport

- 1999: Andy Stanford-Clark (IBM) and Arlen Nipper (Cirrus Link)



Pipelines in Desert. Low Bandwidth- Satellite Communication Links

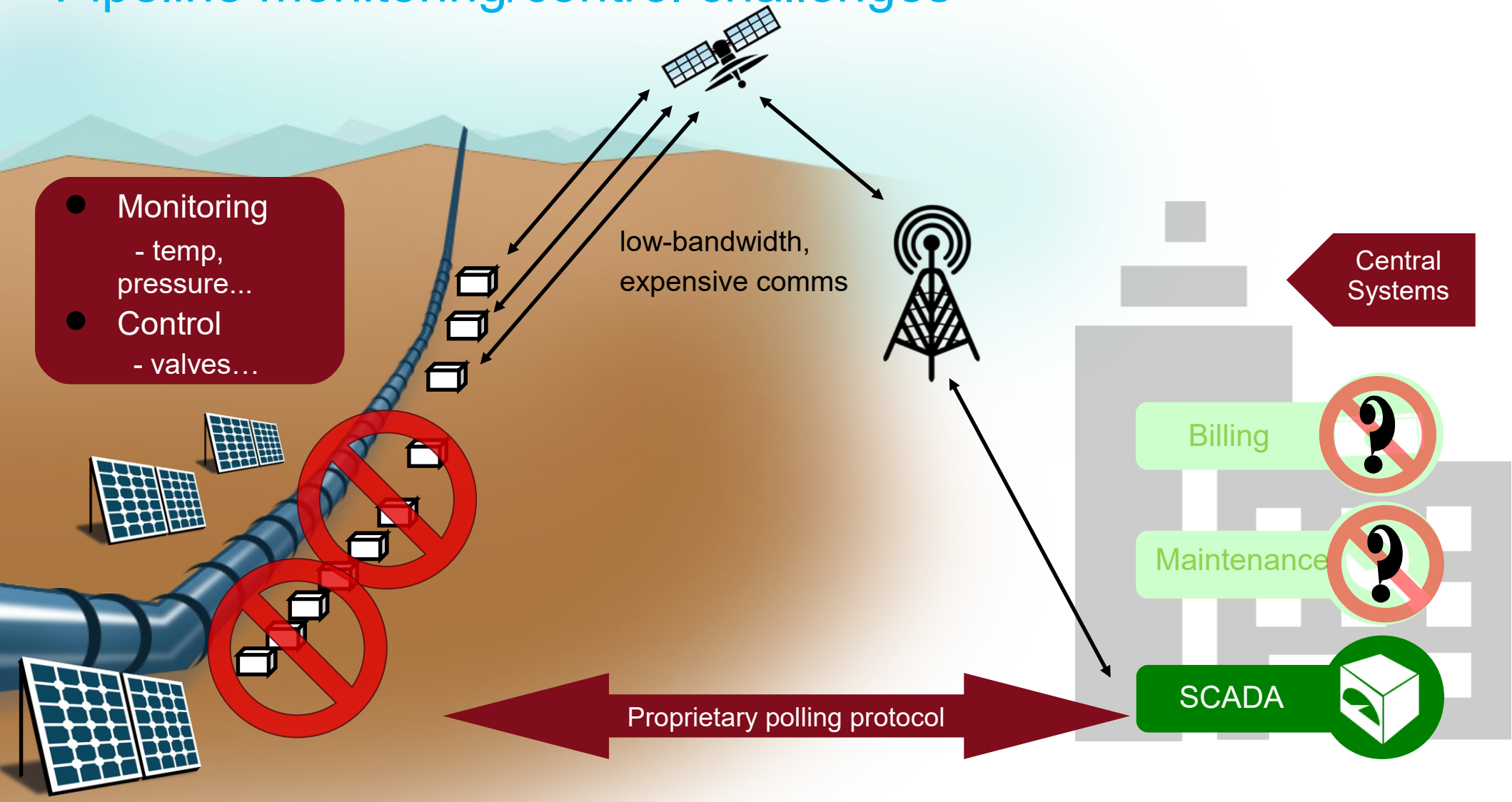


"MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium."

Citation from the official MQTT 3.1.1 specification

Photo credit: Dave Houseknecht, USGS

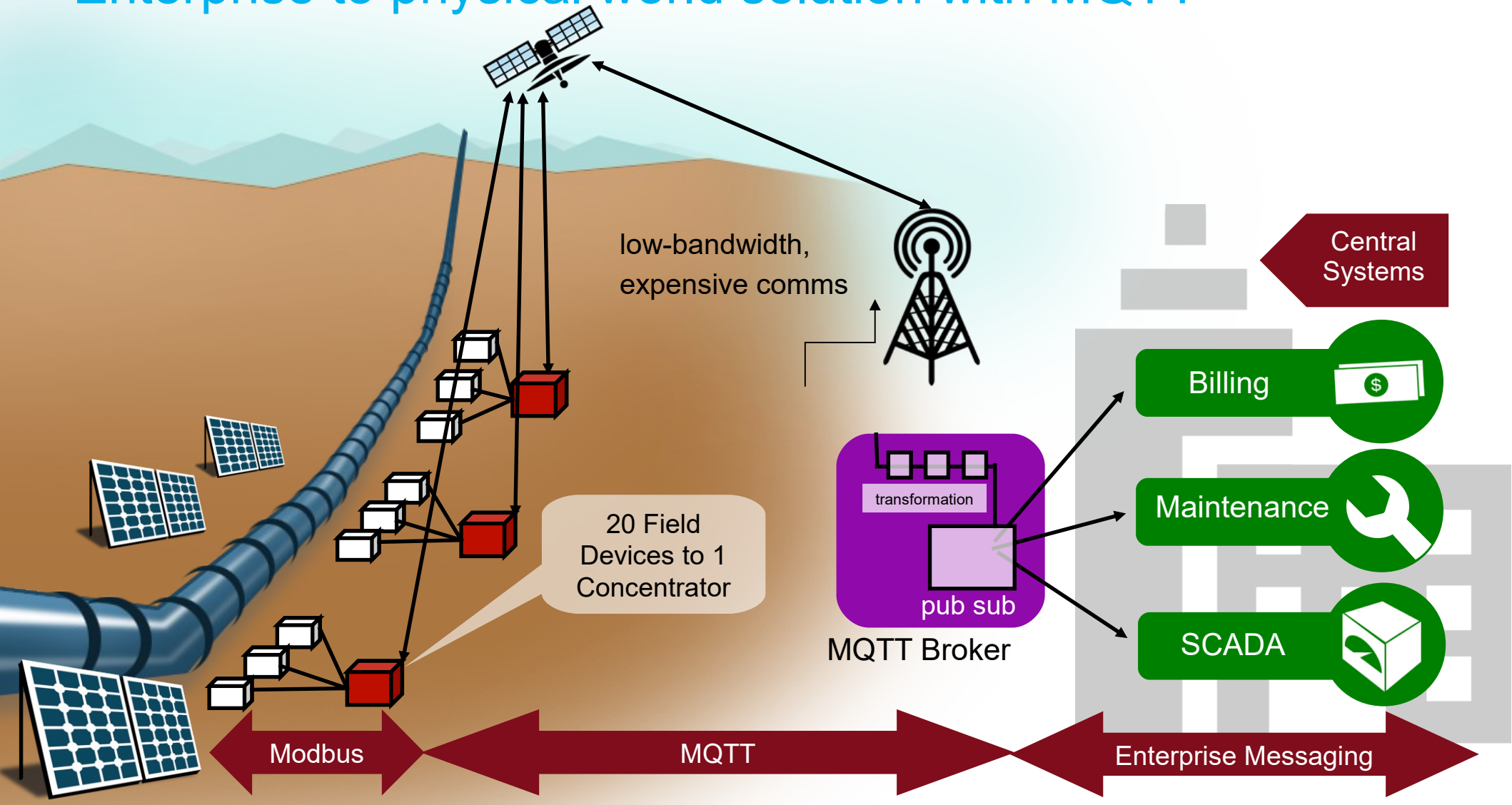
Pipeline monitoring/control challenges



4000 devices integrated, need to add 8000 more BUT:

Satellite network saturated due to polling of device
VALMET system CPU at 100%
Other applications needed access to data ("SCADA prison")

Enterprise to physical world solution with MQTT

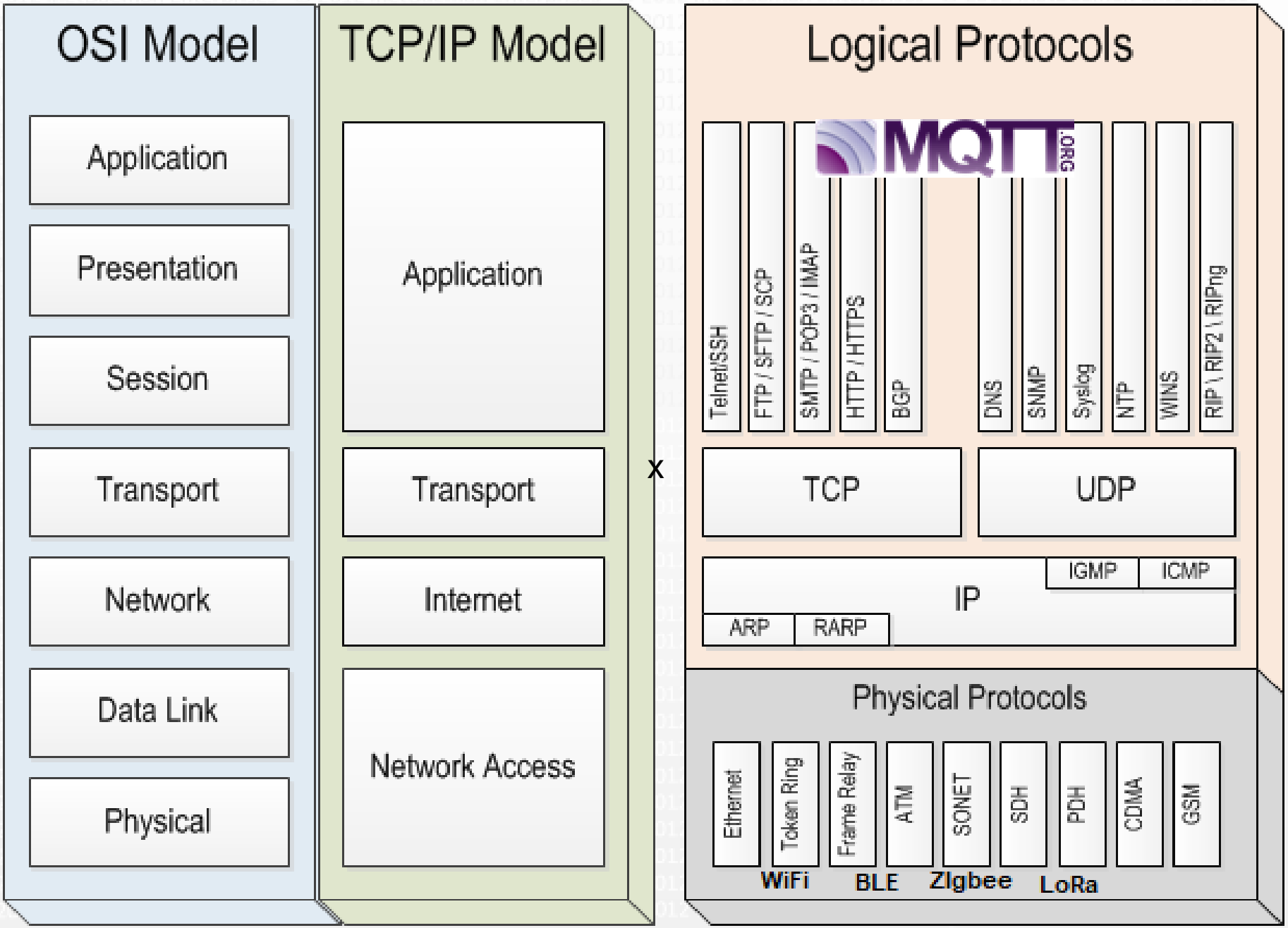


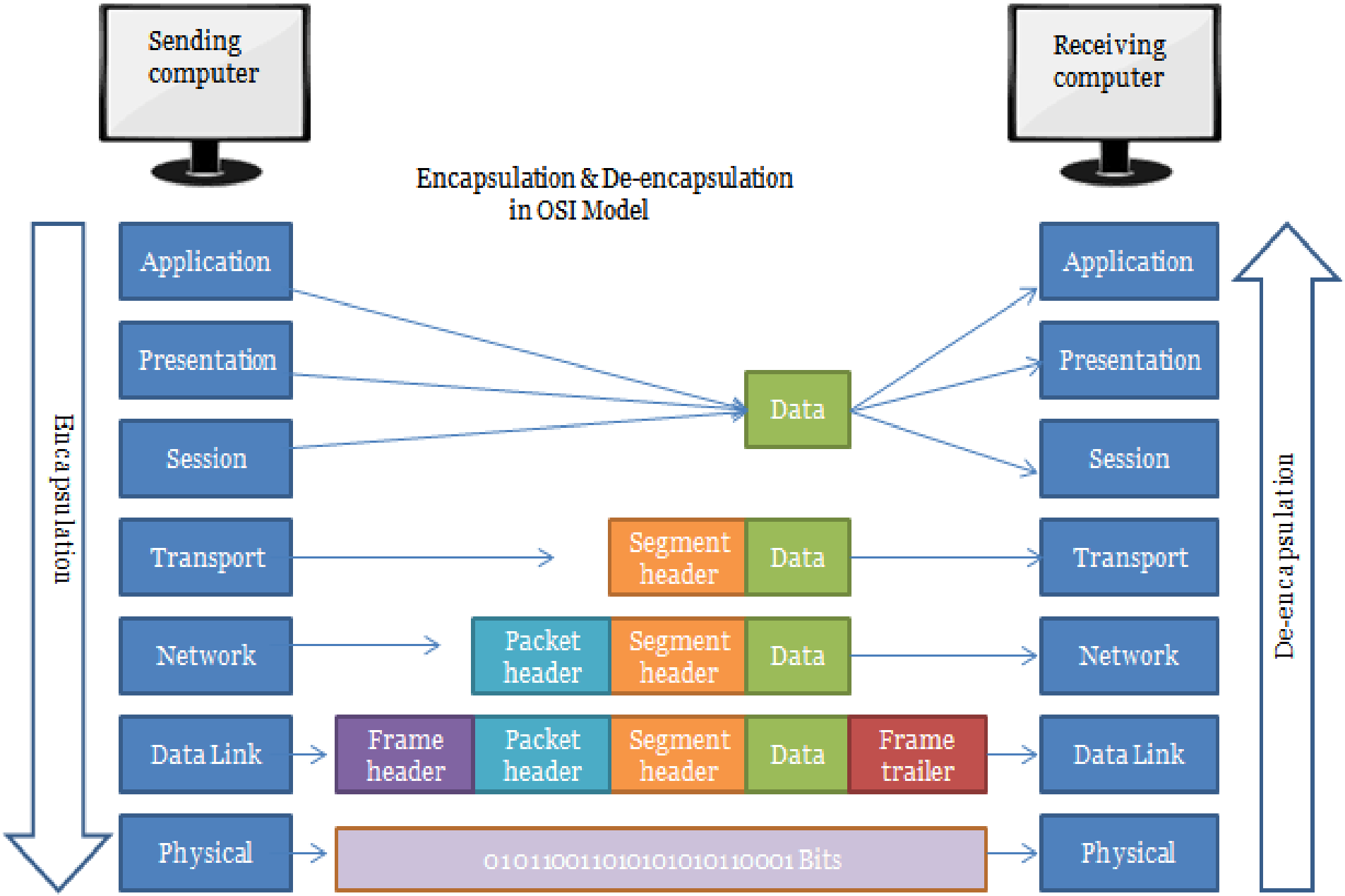
Scalability for whole pipeline
Network traffic much lower - events pushed to/from devices and report by exception
Network cost reduced
Lower CPU utilization
Broken out of the SCADA prison – data accessible to other applications

MQTT – message queuing telemetry transport

Requirements:

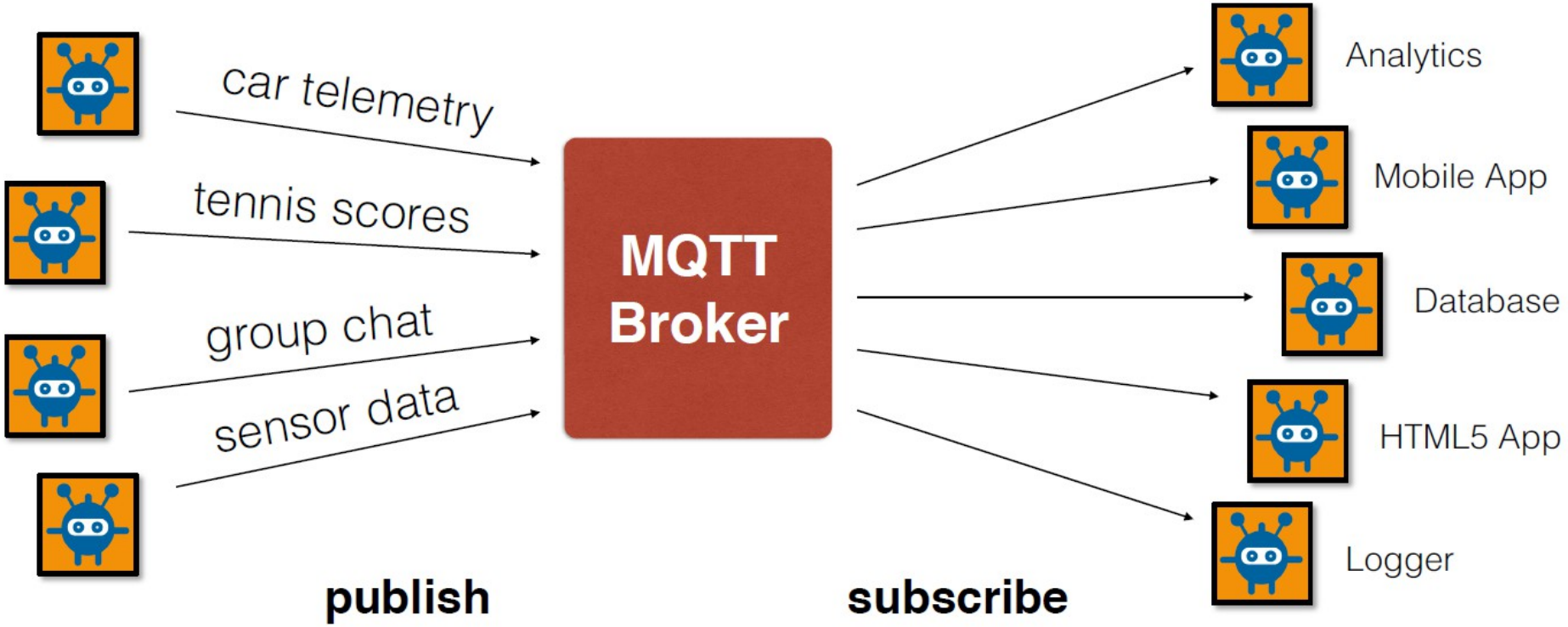
- Simple implementation
- Quality of Service data delivery
- Lightweight and bandwidth efficient
- Data agnostic
- Continuous session awareness

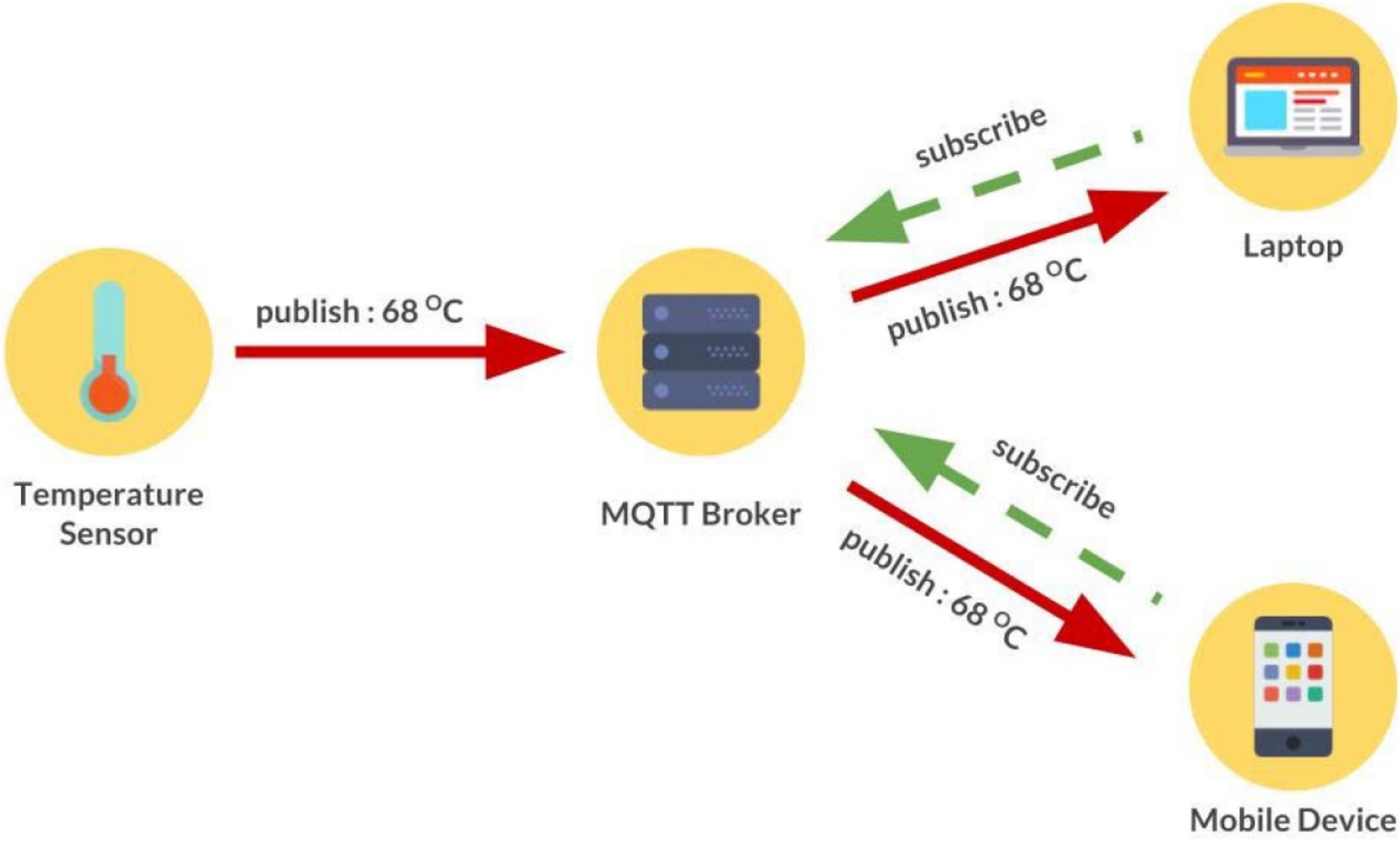



Sending computer
Receiving computer
Encapsulation & De-encapsulation in OSI Model
Application
Presentation
Session
Transport
Network
Data Link
Physical
Data
Segment header
Packet header
Frame header
Frame trailer
0101100110101010101101 0001 Bits
Encapsulation
De-encapsulation

MQTT

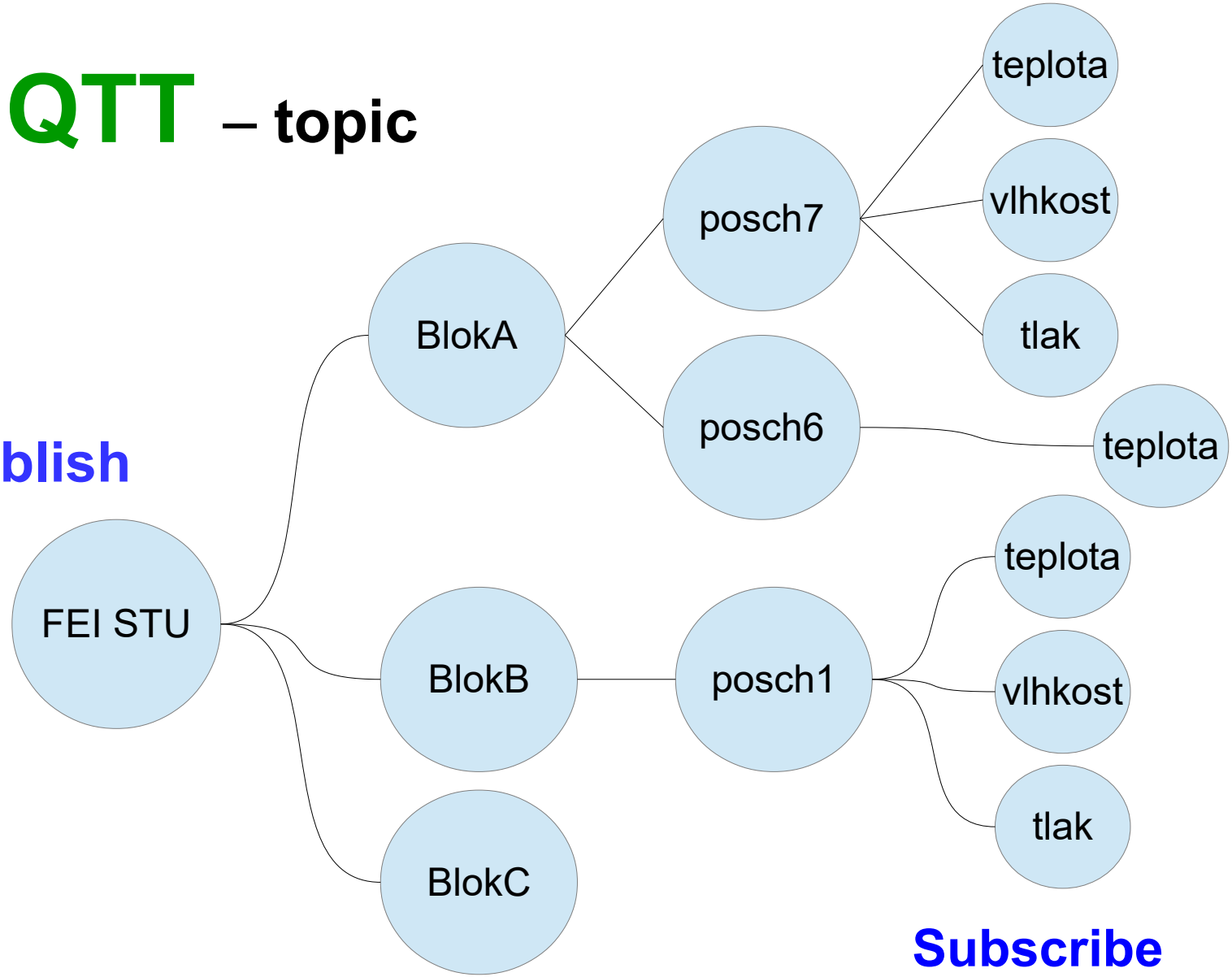
pub/sub decouples senders from receivers





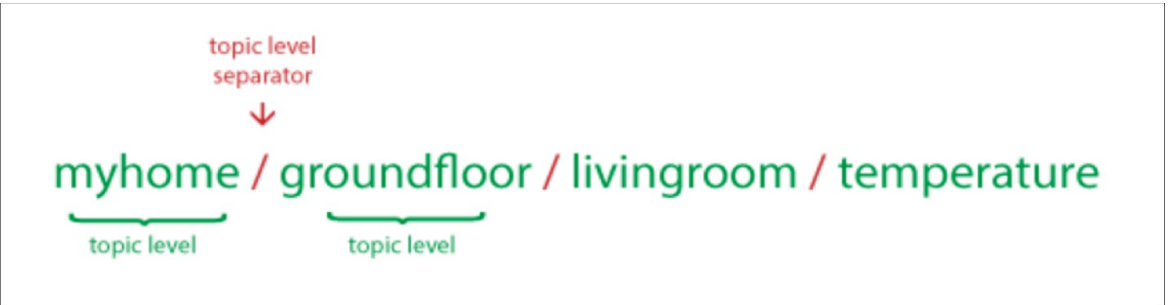
Hermanudin, Aldwin & Ekadiyanto, Fransiskus & Sari, Riri. (2019). Performance Evaluation of CoAP Broker and Access Gateway Implementation on Wireless Sensor Network. 10.1109/TENCONSpring.2018.8692050.

MQTT – topic



Publish

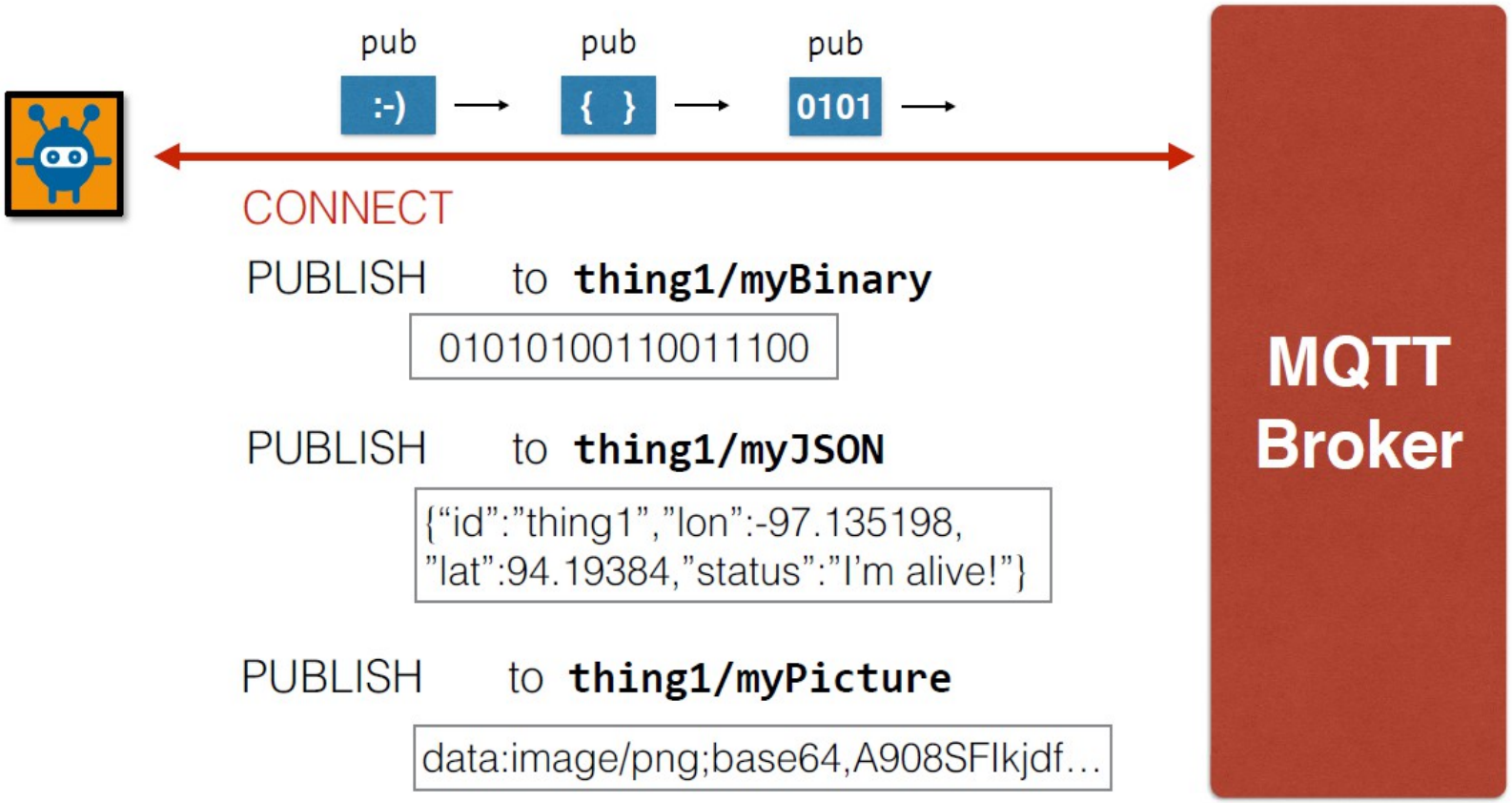
Subscribe



FEISTU / BlokA / posch7 / teplota
FEISTU / BlokA / posch7 / #
FEISTU / BlokA / + / teplota
FEISTU / #

MQTT

agnostic payload for flexible delivery



MQTT

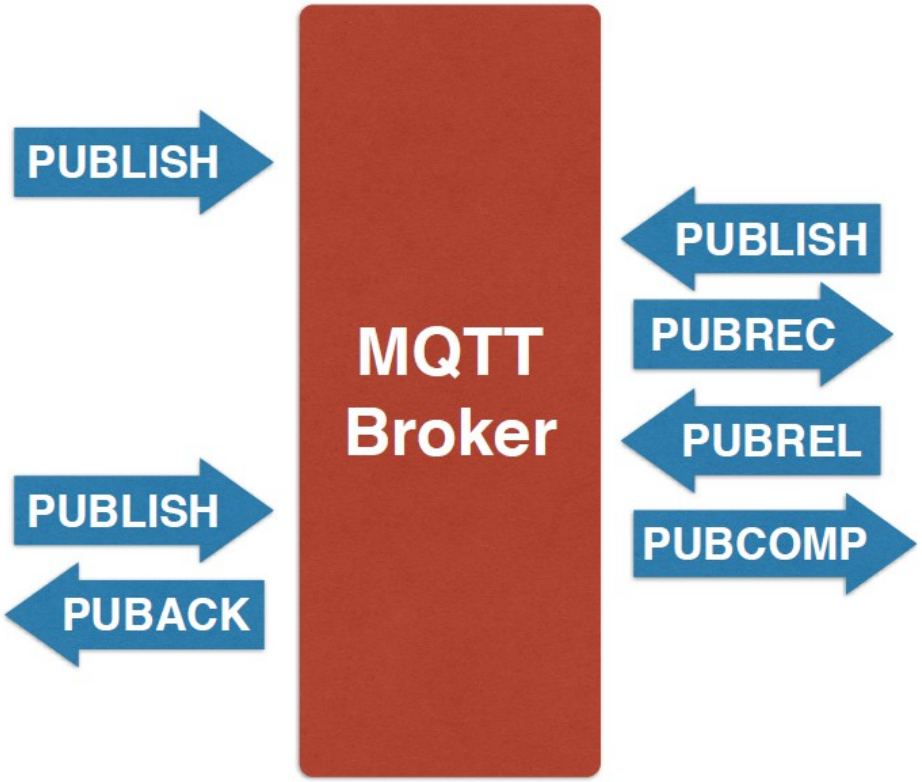
Quality of Service for reliable messaging

QoS 0
at most once

- doesn't survive failures
- never duplicated

QoS 1
at least once

- survives connection loss
- can be duplicated

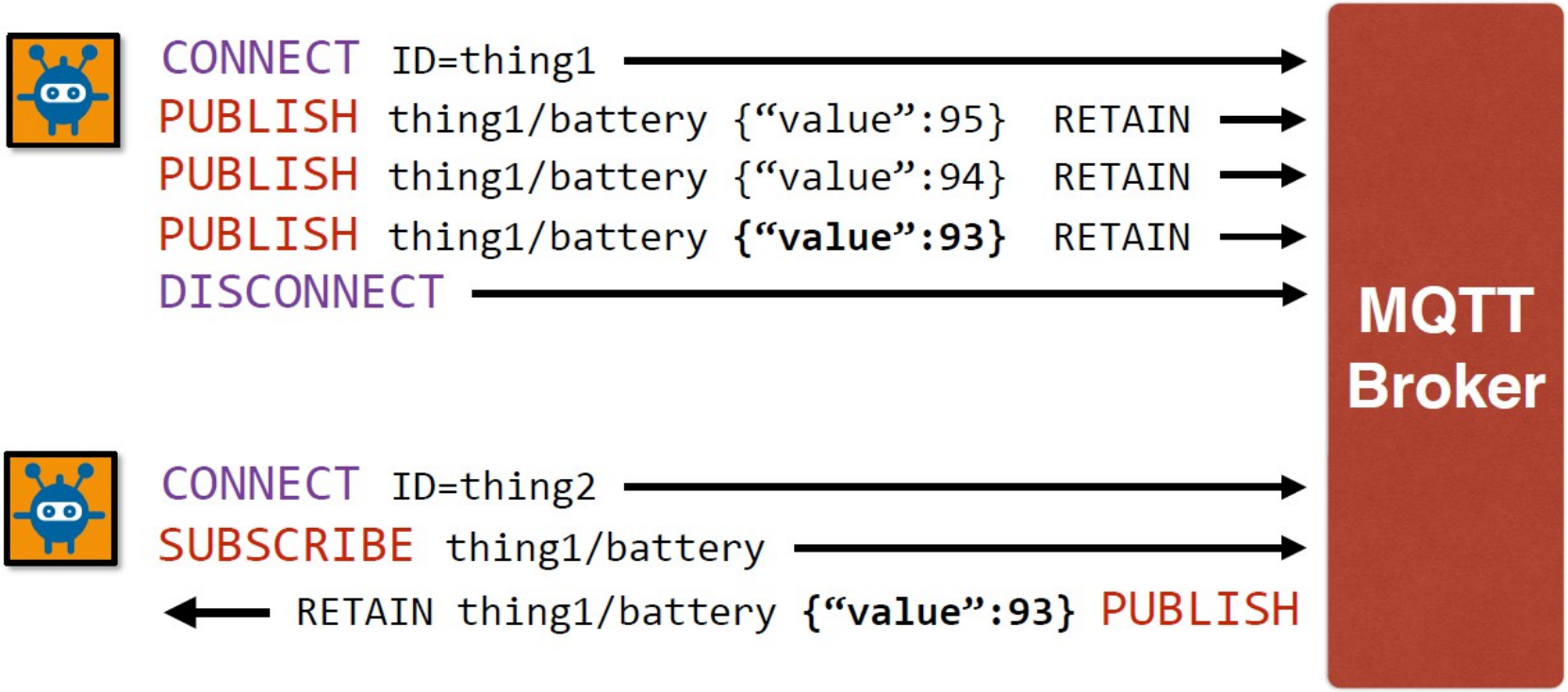


QoS 2
exactly once

- survives connection loss
- never duplicated

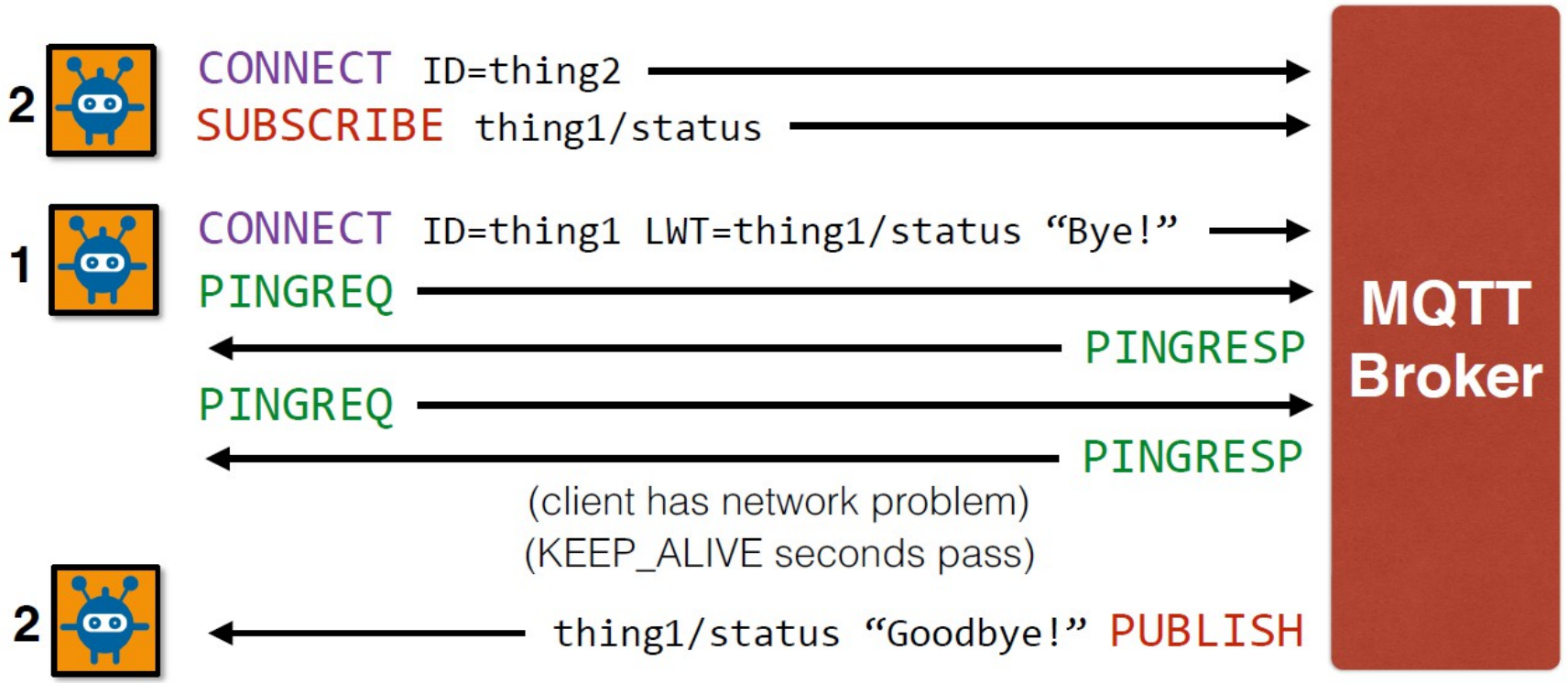
MQTT

retained messages for last value caching

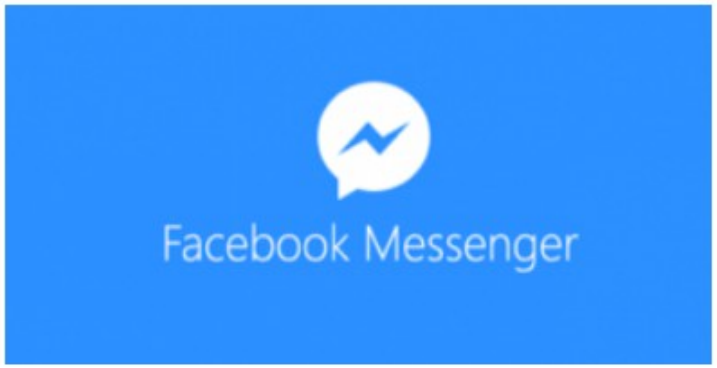


MQTT

last will and testament for presence

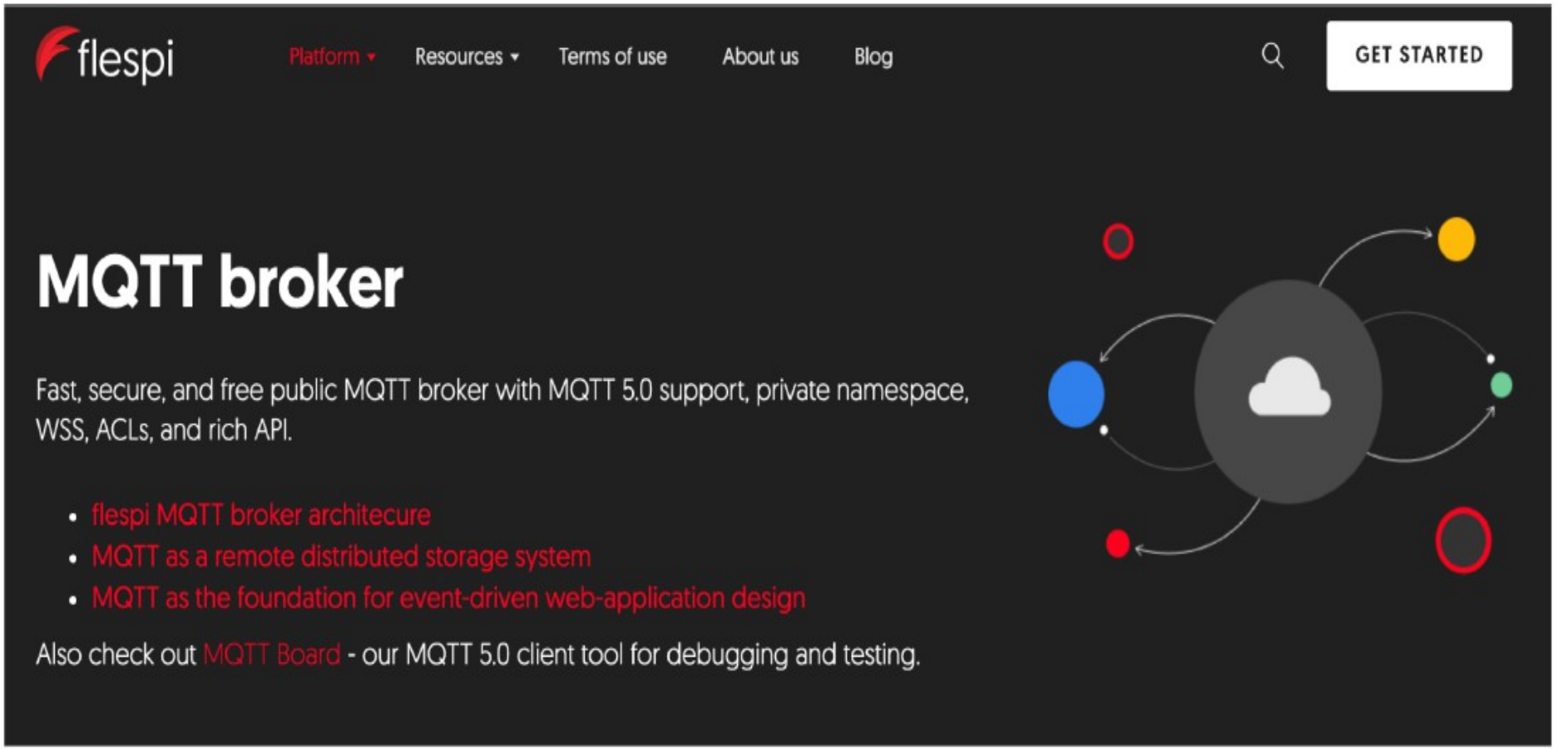


Popular Users

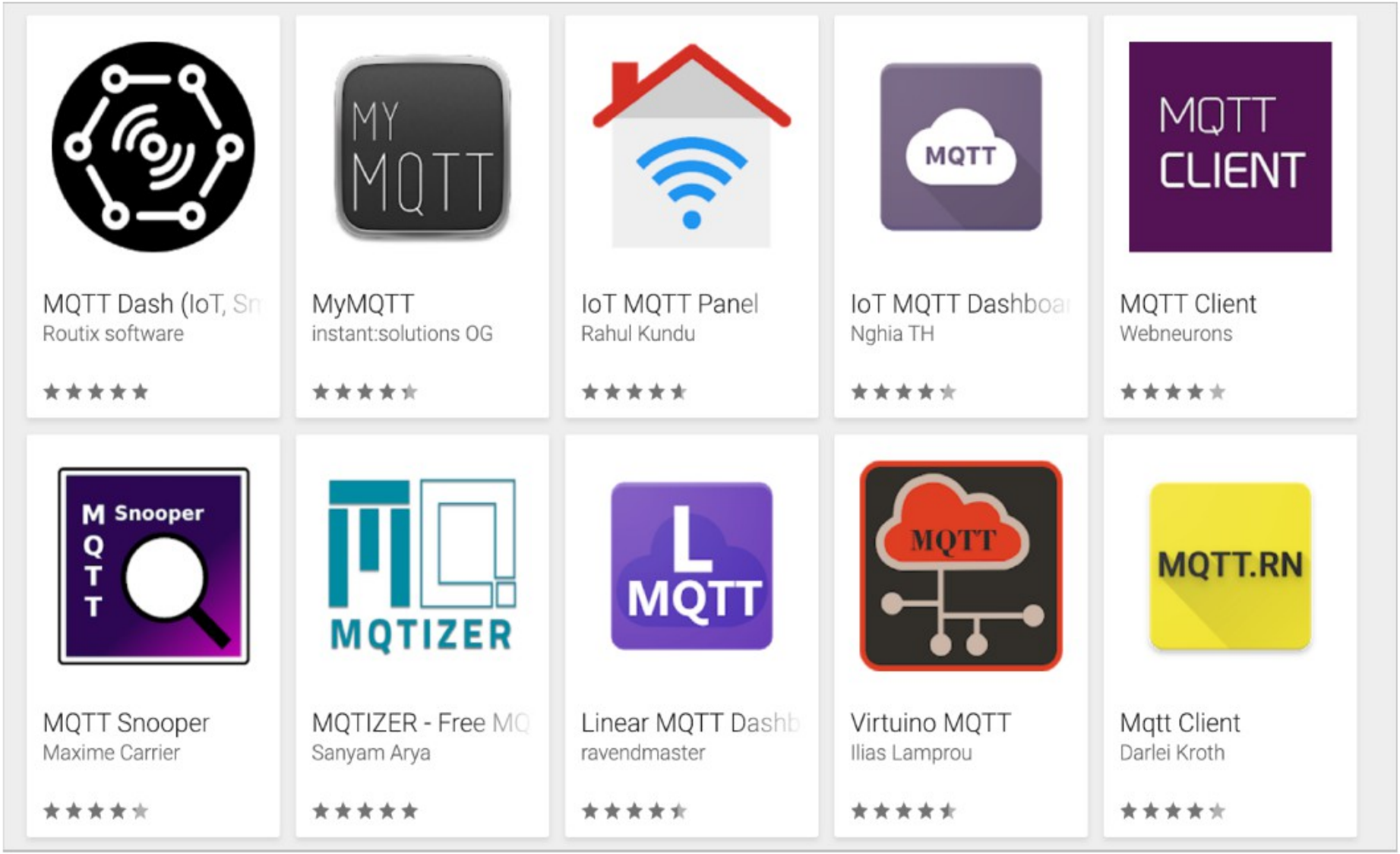


Cloud based brokers: flespi

https://flespi.com/mqtt-broker



MQTT clients: Android



MQTT – disadvantages

- **If the broker fails…**
- Does not define a standard client API, so application developers have to select the best fit.
- Does not include many features that are common in Enterprise Messaging Systems like:
  - expiration, timestamp, priority, custom message headers, …
- Does not have a point-to-point (aka queues) messaging pattern
  - Point to Point or One to One means that there can be more than one consumer listening on a queue but only one of them will be get the message
- Maximum message size 256MB

MQTT – príklad v Processingu

```
import mqtt.*;
MQTTClient client;

void setup() {
  client = new MQTTClient(this);
  client.connect("mqtt://try:try@broker.shiftr.io", "userName");
}

void draw() { /* draw nothing */}

void keyPressed() {
  client.publish("/FEISTU", "myMessage");
}
```

MQTT – príklad v Processingu

```
void clientConnected() {
  println("client connected");
  client.subscribe("/hello");
}

void messageReceived(String topic, byte[] payload) {
  println("new message: " + topic + " - " + new String(payload));
}

void connectionLost() {
  println("connection lost");
}
```

JSON – JavaScript Object Notation

Douglas Crockford

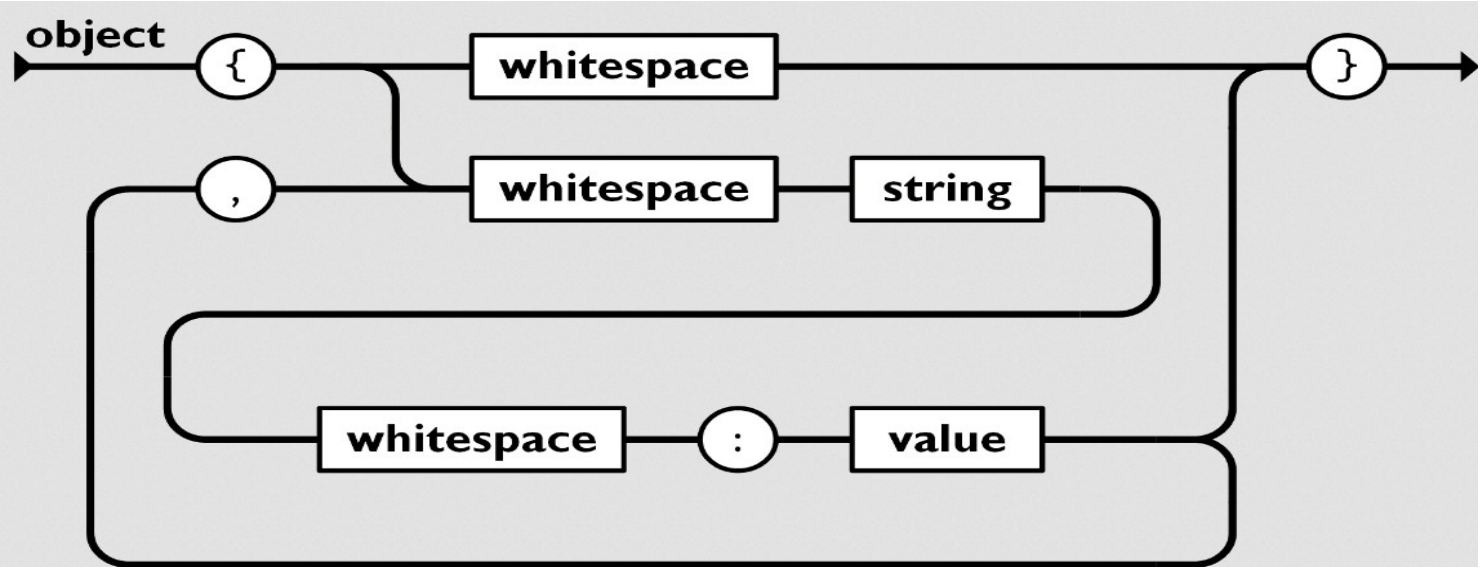
```
Developed by Douglas Crockford

Standard ISO/IEC 21778:2017

The simplest supported data formats are:

    {"key1":"value1", "key2":"value2"}
```

```
{"stringKey":"value1", "booleanKey":true, "doubleKey":42.0, "longKey":73}
```



XML vs. JSON

```xml
<person>
  <name>John Smith</name>
  <age>25</age>
  <address>
    <street>21 2nd Street</street>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <sex>
    <type>male</type>
  </sex>
</person>
```

```json
{
  "name": "John Smith",
  "age": 25,
  "address": {
    "street": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postal code": "10021"
  },
  "sex": {"type": "male"}
}
```

JSON – príklad v Processingu

```
JSONObject message;
void setup()
{  message = new JSONObject();
   message.setFloat("temperature", 10.0);
   message.setInt("state",2);
   message.setString("name", "Lion");

   saveJSONObject(message, "data/new.json");
    int aktualnyStav = message.getInt("state");
   float aktualnaTeplota = message.getFloat("temperature");
   String realName = message.getString("name");
   println("Stav: " + aktualnyStav
  + ", Teplota: " + aktualnaTeplota + ",  Meno: " + realName);
}
```

JSON – príklad v Processingu - pokračovanie

```
void draw() { /* nic nekreslime */ }

void keyPressed() {

  temperature = random(-10, 32.5);
  message.setFloat("temperature", temperature);
  println(message.toString());
}
```

Úloha – zadanie

Vyskúšajte si posielanie protokolom MQTT. Pošlite jednoduchú správu

MQTT server `mqtt://try:try@broker.shiftr.io`

topic **`/feistu/misa/2020/XXX`**

a potom na

MQTT server `mqtt://9RYd7rPhakMm9CCwPBJG@demo.thingsboard.io`

topic **`v1/devices/me/telemetry`**

Správa vo formáte JSON má vyzerať takto:

```
{"XXX-Lat": 49.1634, "XXX-Lon": 20.1349, "XXX-Temp": 18.2}
```

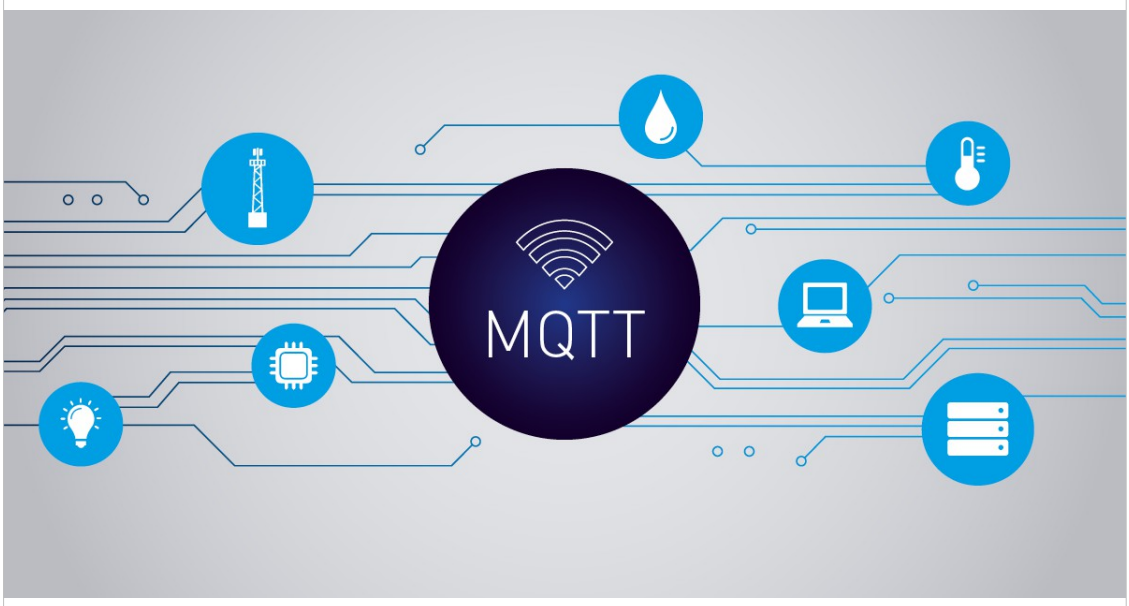
kde

`XXX` sú prvé tri písmená vášho priezviska

`Lat` je zemepisná šírka na štyri desatinné miesta

`Lon` je zemepisná dĺžka na štyri desatinné miesta

`Temp` aktuálna vonkajšia teplota


MQTT

MQTT

A practical protocol for the **Internet of Things**

TV Sets
Pacemakers
Ovens
Vehicles
Cows
Smartphones

The Internet is (in) everything

- **vehicles**
- **children**
- **cows**
- **smartphones**
- **ovens**
- **pacemakers**

By the year 2020…

57,000 /sec
new objects connecting

212 BILLION
Total number of available sensor enabled objects

30 BILLION
sensor enabled objects **connected to networks**

Data source: IDC

The world is getting smarter

Smarter Vehicles

- realtime telemetry
- predictive maintenance
- look-ahead alerting
- pay-as-you-drive

Smarter Homes

- energy tracking
- automation
- remote monitoring
- smart appliances

Smarter Logistics

- end-to-end tracking
- theft prevention
- real-time updates
- fleet monitoring

Smarter Healthcare

- smart scales
- in-home monitoring
- assisted living
- physician messaging

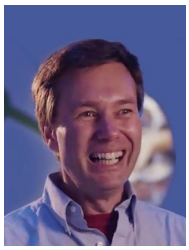
MQTT a lightweight protocol for IoT messaging

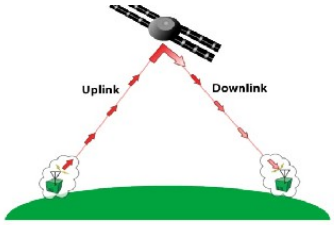
- **open** open spec, standard 40+ client implementations
- **lightweight** minimal overhead efficient format tiny clients (kb)
- **reliable** QoS for reliability on unreliable networks
- **simple** 43-page spec connect + publish + subscribe

Invented	Published	Eclipse M2M	Standard
EUROTECH IBM	MQTT.org	paho	OASIS
Late 1990s	Aug 2010	Nov 2011	Sep 2014

MQTT – message queuing telemetry transport

- 1999: Andy Stanford-Clark (IBM) and Arlen Nipper (Cirrus Link)

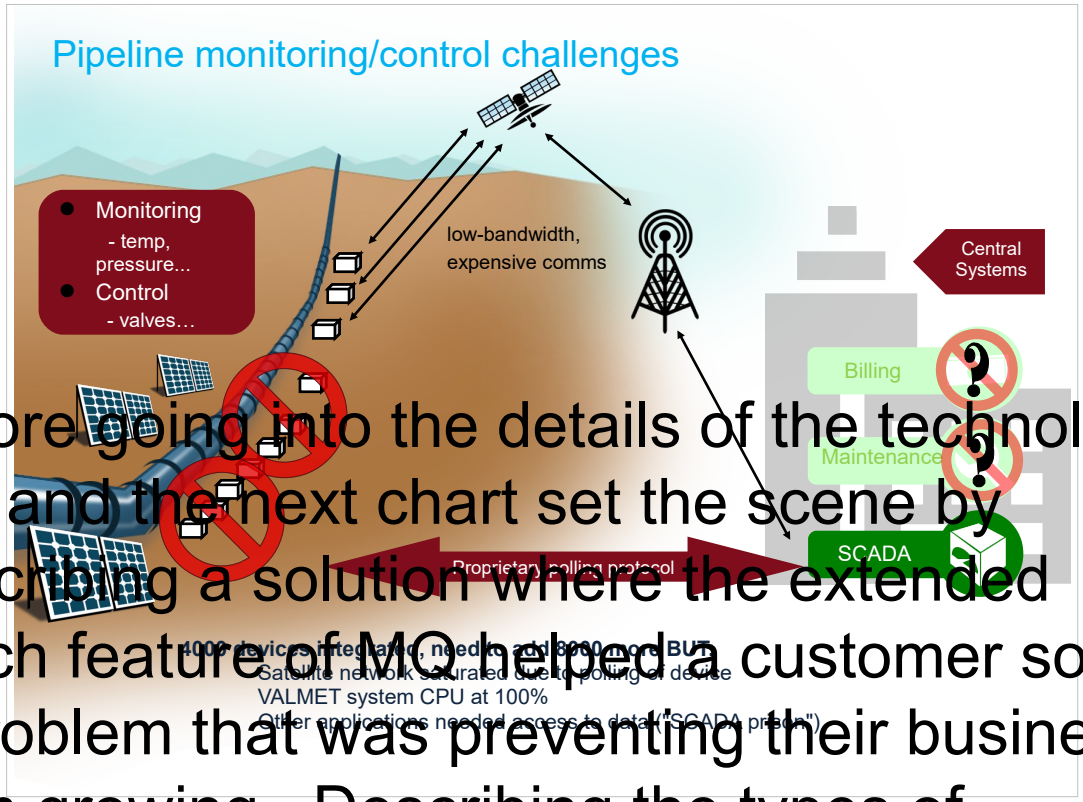




Pipelines in Desert. Low Bandwidth- Satellite Communication Links

> "MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium."
>
> Citation from the official MQTT 3.1.1 specification

Photo credit: Dave Houseknecht, USGS



Before going into the details of the technology this and the next chart set the scene by describing a solution where the extended reach feature of MQ helped a customer solve a problem that was preventing their business from growing. Describing the types of problems that it is designed to help with.

The problem:

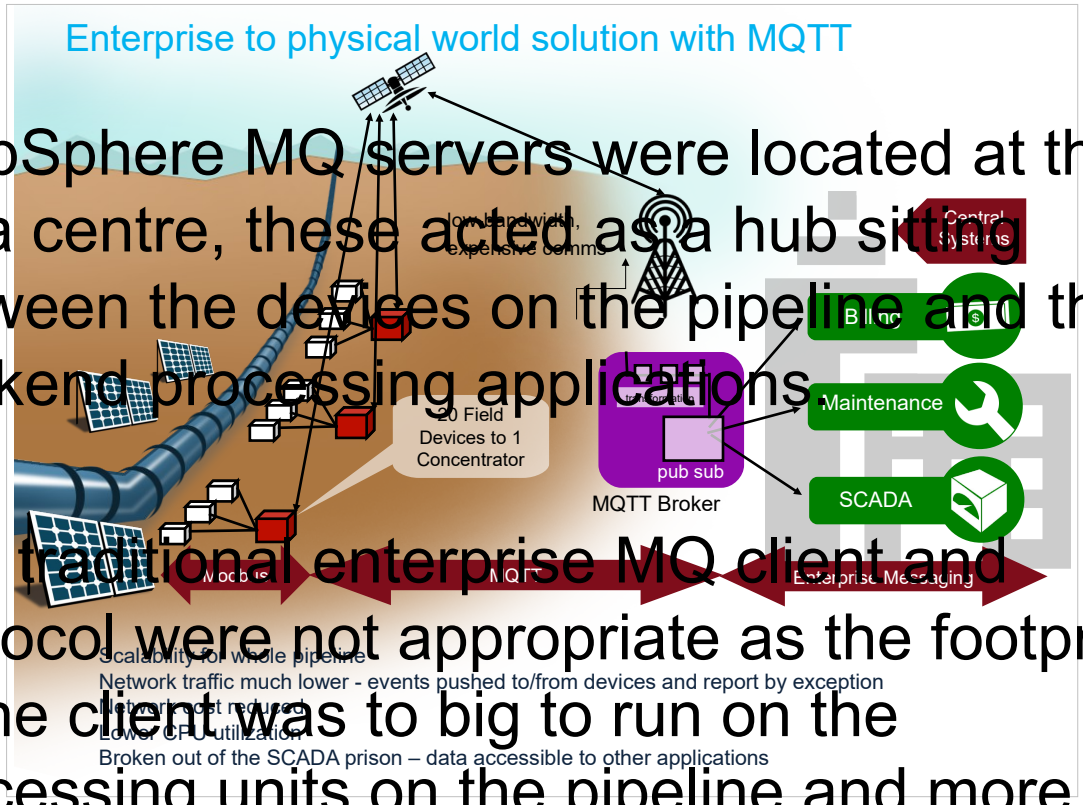
A pipeline running thousands of kilometres through inhospitable regions. The existing solution monitored and controlled 4000 devices along the pipeline BUT the system needed to scale to a total of 12,000 devices. This type of system is referred to as a SCADA system or supervisory control and data acquisition.

The incumbent system could not scale for a couple of reasons

1)the network was saturated

2)The monitoring and control application running on the central system was running at a

system to use an asynchronous messaging approach.



WebSphere MQ servers were located at the data centre, these acted as a hub sitting between the devices on the pipeline and the backend processing applications.

The traditional enterprise MQ client and protocol were not appropriate as the footprint of the client was to big to run on the processing units on the pipeline and more importantly the protocol was far too heavy for the network that was charged by the volume of data sent.

Instead the MQ Telemetry Transport or MQTT for short was chosen to provide the comms between the pipeline and the MQ server. The use of MQTT together with MQ solved the customers problems because:

1)MQTT is a bidirectional/duplex messaging protocol enabling messages and events to be pushed from the client to the server and the server to the client. Pushing messages is inherently more efficient than polling. Rather than the server continuously polling the device to find out when a state has changed now a state change event is generated and pushed to the server only when the state change occurs. This change alone dramatically reduced the network usage enabling the 8000 additional evices to be added to the system.

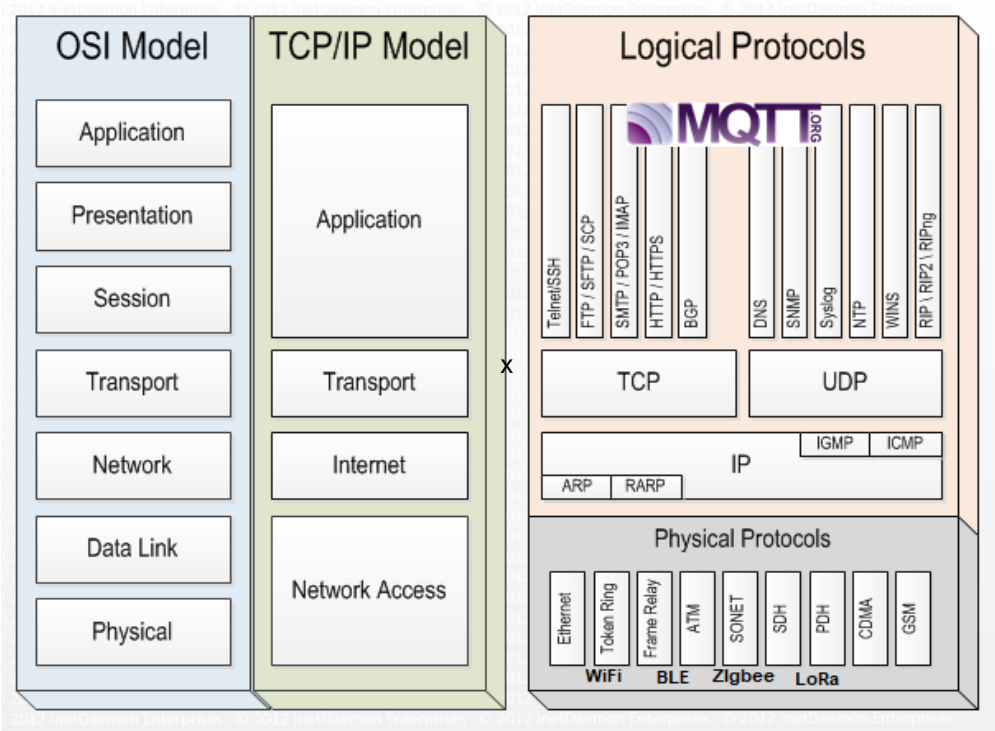
2)The MQTT protocol has a tiny footprint on the wire, this combined with only sending data when needed meant that network costs were reduced (for the equivalent no of devices)

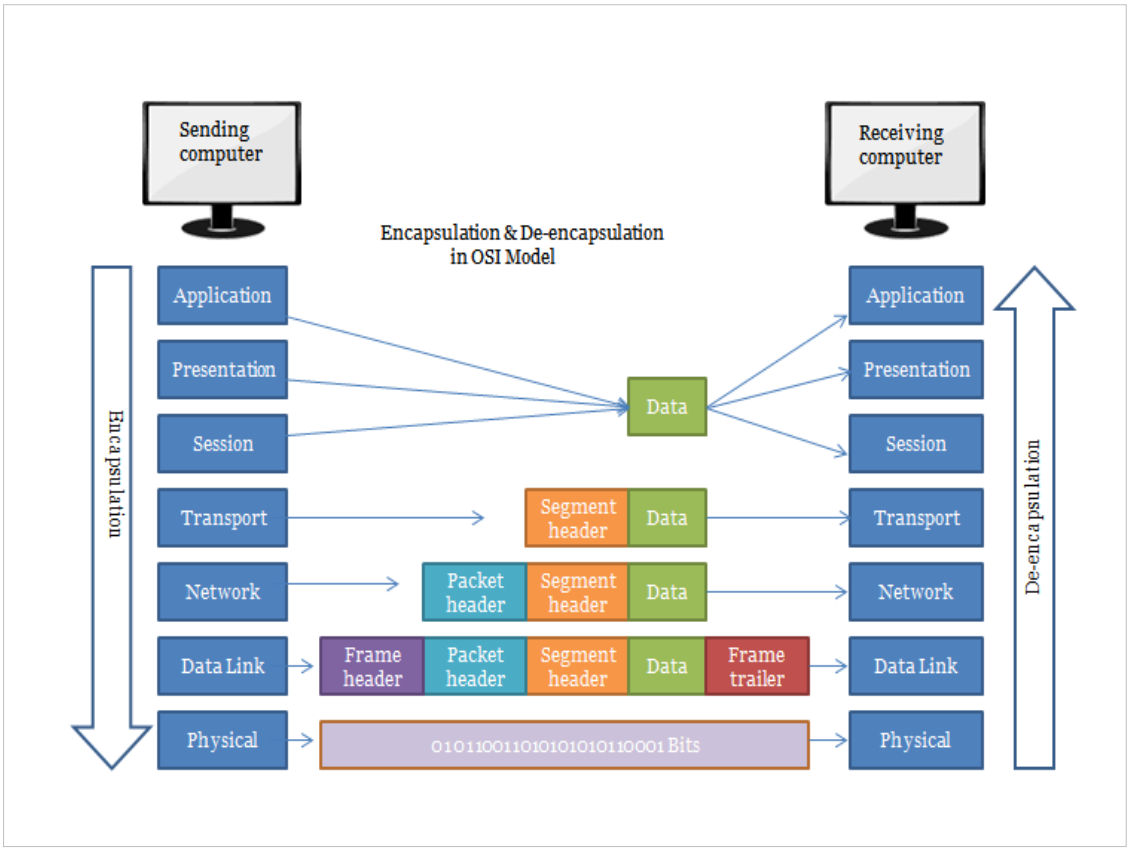
3)MQTT is a publish subscribe protocol. When a message is published (sent) to the MQ server multiple applications can access the message by subscribing to it. The Valmet monitoring and control application subscribed for all messages as before but

MQTT – message queuing telemetry transport

Requirements:

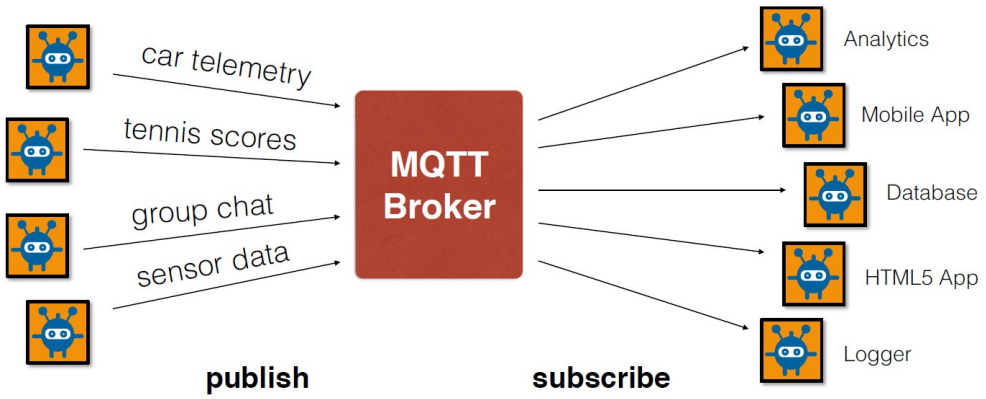
- Simple implementation
- Quality of Service data delivery
- Lightweight and bandwidth efficient
- Data agnostic
- Continuous session awareness

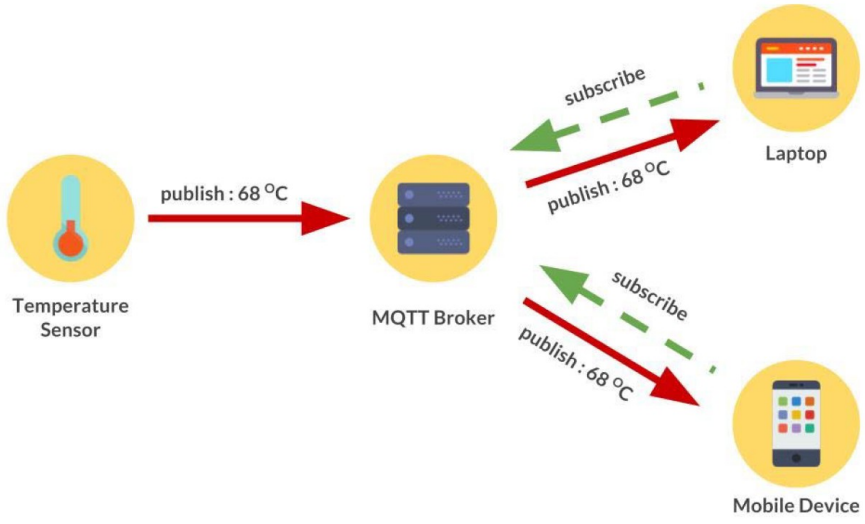

OSI Model
Application
Presentation
Session
Transport
Network
Data Link
Physical
TCP/IP Model
Application
Transport
Internet
Network Access
x
Logical Protocols
MQTT.ORG
Telnet/SSH
FTP / SFTP / SCP
SMTP / POP3 / IMAP
HTTP / HTTPS
BGP
DNS
SNMP
Syslog
NTP
WINS
RIP \ RIP2 \ RIPng
TCP
UDP
IP
IGMP
ICMP
ARP
RARP
Physical Protocols
Ethernet
Token Ring
Frame Relay
ATM
SONET
SDH
PDH
CDMA
GSM
WiFi
BLE
Zigbee
LoRa


Sending computer
Receiving computer
Encapsulation & De-encapsulation in OSI Model
Encapsulation
De-encapsulation
Application
Presentation
Session
Transport
Network
Data Link
Physical
Data
Segment header
Packet header
Frame header
Frame trailer
0101100110101010101100001 Bits

MQTT

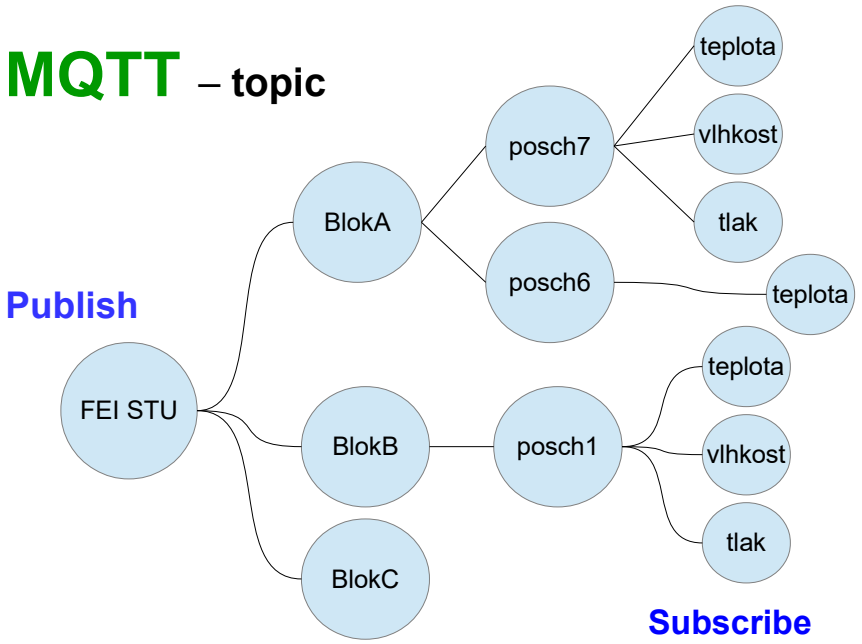
pub/sub decouples senders from receivers

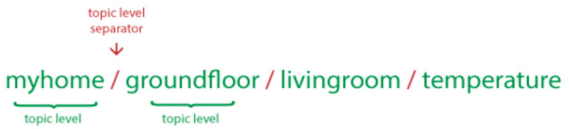




Hermanudin, Aldwin & Ekadiyanto, Fransiskus & Sari, Riri. (2019). Performance Evaluation of CoAP Broker and Access Gateway Implementation on Wireless Sensor Network. 10.1109/TENCONSpring.2018.8692050.

MQTT – topic

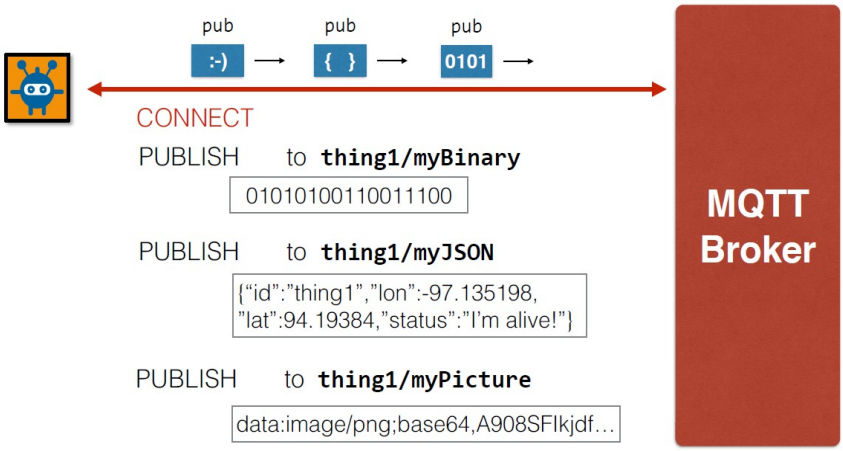




FEISTU / BlokA / posch7 / teplota
FEISTU / BlokA / posch7 / #
FEISTU / BlokA / + / teplota
FEISTU / #

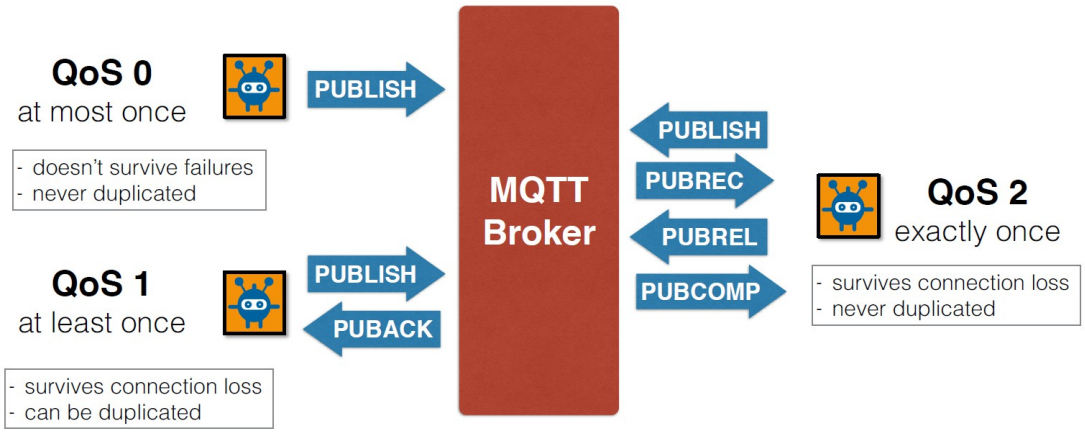
MQTT

agnostic payload for flexible delivery



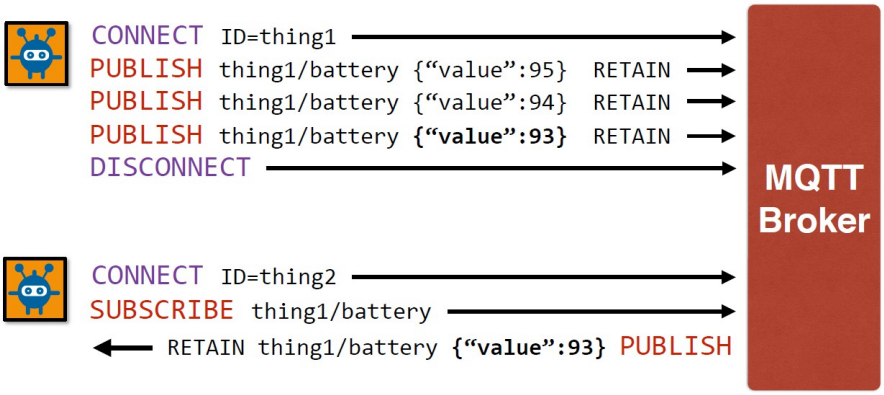
MQTT

Quality of Service for reliable messaging



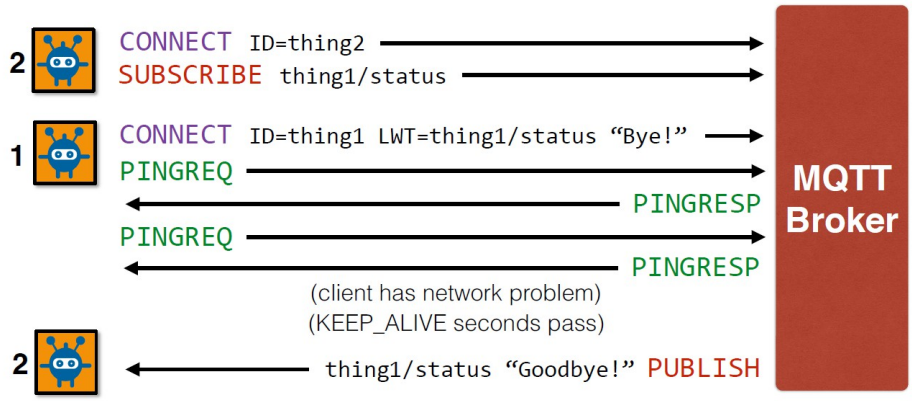
MQTT

retained messages for last value caching



MQTT

last will and testament for presence



Popular Users

Facebook Messenger

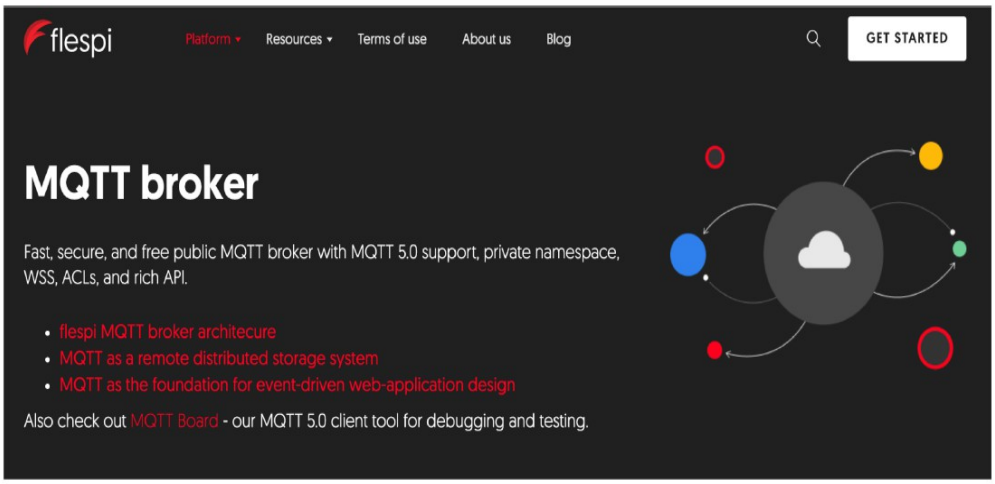
openHAB
empowering the smart home

EVRYTHNG™

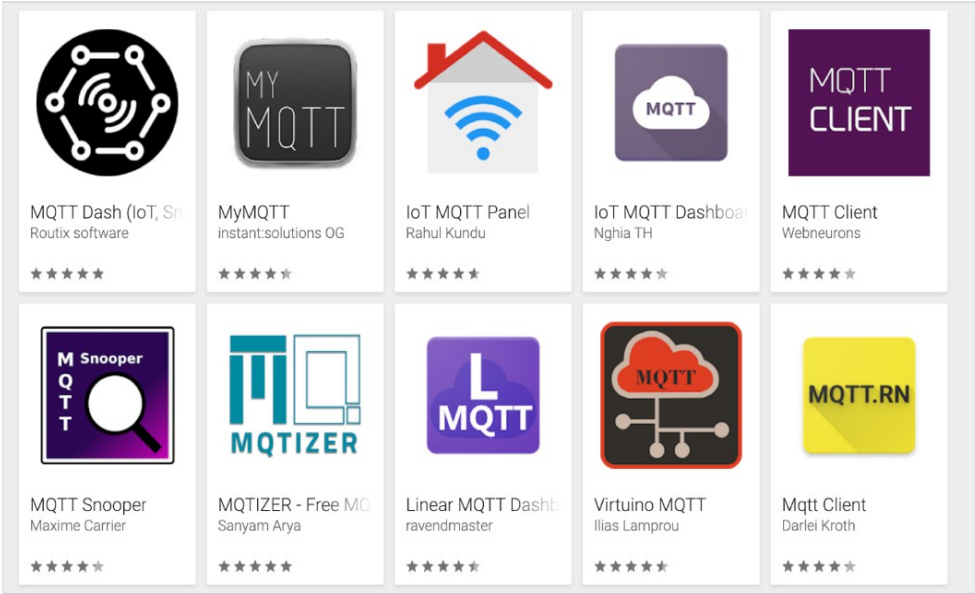
amazon
web services

Cloud based brokers: flespi

https://flespi.com/mqtt-broker



MQTT clients: Android



MQTT – disadvantages

- **If the broker fails…**
- Does not define a standard client API, so application developers have to select the best fit.
- Does not include many features that are common in Enterprise Messaging Systems like:
  - expiration, timestamp, priority, custom message headers, …
- Does not have a point-to-point (aka queues) messaging pattern
  - Point to Point or One to One means that there can be more than one consumer listening on a queue but only one of them will be get the message
- Maximum message size 256MB

MQTT – príklad v Processingu

```
import mqtt.*;
MQTTClient client;

void setup() {
  client = new MQTTClient(this);
  client.connect("mqtt://try:try@broker.shiftr.io", "userName");
}

void draw() { /* draw nothing */}

void keyPressed() {
  client.publish("/FEISTU", "myMessage");
}
```

MQTT – príklad v Processingu

```
void clientConnected() {
  println("client connected");
  client.subscribe("/hello");
}

void messageReceived(String topic, byte[] payload) {
  println("new message: " + topic + " - " + new String(payload));
}

void connectionLost() {
  println("connection lost");
}
```

JSON – JavaScript Object Notation

Douglas Crockford

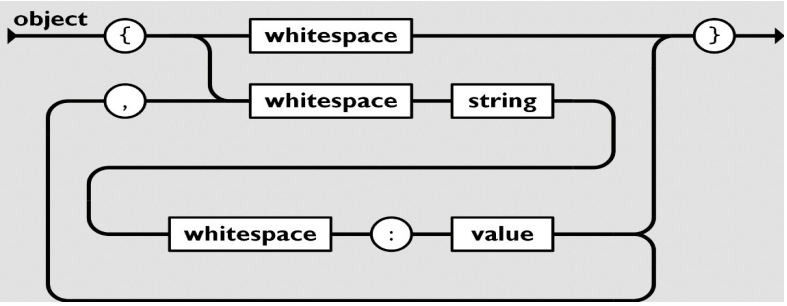
Developed by Douglas Crockford

Standard ISO/IEC 21778:2017

The simplest supported data formats are:

```
    {"key1":"value1", "key2":"value2"}
```

```
{"stringKey":"value1", "booleanKey":true, "doubleKey":42.0, "longKey":73}
```



XML vs. JSON

```xml
<person>
  <name>John Smith</name>
  <age>25</age>
  <address>
    <street>21 2nd Street</street>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <sex>
    <type>male</type>
  </sex>
</person>
```

```json
{
  "name": "John Smith",
  "age": 25,
  "address": {
    "street": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postal code": "10021"
  },
  "sex": {"type": "male"}
}
```

JSON – príklad v Processingu

```
JSONObject message;
void setup()
{  message = new JSONObject();
   message.setFloat("temperature", 10.0);
   message.setInt("state",2);
   message.setString("name", "Lion");

   saveJSONObject(message, "data/new.json");
    int aktualnyStav = message.getInt("state");
   float aktualnaTeplota = message.getFloat("temperature");
   String realName = message.getString("name");
   println("Stav: " + aktualnyStav
  + ", Teplota: " + aktualnaTeplota + ",  Meno: " + realName);
}
```

JSON – príklad v Processingu - pokračovanie

```
void draw() { /* nic nekreslime */ }

void keyPressed() {

  temperature = random(-10, 32.5);
  message.setFloat("temperature", temperature);
  println(message.toString());
}
```

Úloha – zadanie

Vyskúšajte si posielanie protokolom MQTT. Pošlite jednoduchú správu

MQTT server `mqtt://try:try@broker.shiftr.io`

topic **`/feistu/misa/2020/XXX`**

a potom na

MQTT server `mqtt://9RYd7rPhakMm9CCwPBJG@demo.thingsboard.io`

topic **`v1/devices/me/telemetry`**

Správa vo formáte JSON má vyzerať takto:

```
{"XXX-Lat": 49.1634, "XXX-Lon": 20.1349, "XXX-Temp": 18.2}
```

kde

- `XXX` sú prvé tri písmená vášho priezviska
- `Lat` je zemepisná šírka na štyri desatinné miesta
- `Lon` je zemepisná dĺžka na štyri desatinné miesta
- `Temp` aktuálna vonkajšia teplota