MQTT vs. CoAP

challenging to rule IoT transmission?

Group D

Lakshika Perera, Mari Nikkarinen, Manohar Reddy Bayyapu Reddy, Martin Maritsch

Message Queue Telemetry Transport

- Client-Server publish/subscribe messaging transport protocol
- For M2M connections and the IoT
- Lightweight, open, simple and designed so as to be easy to implement
- On top of TCP (ensures reliability)

Features

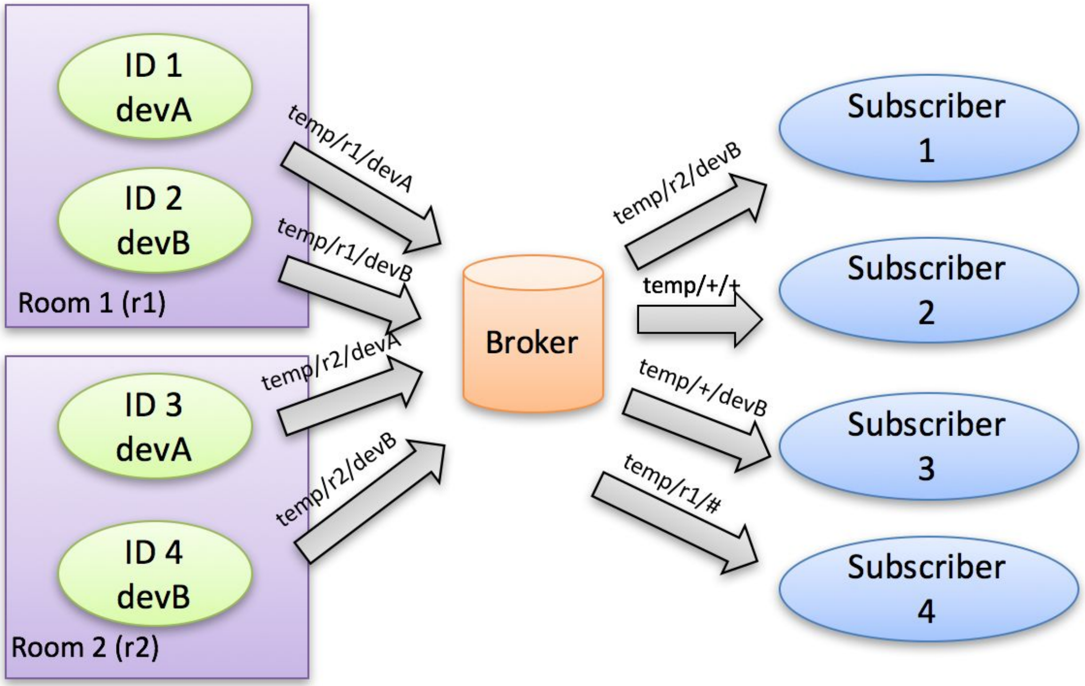
- Topic hierarchy to group messages
  - Multicasts easily possible
- Quality of Service (QoS) levels
- Persistent Sessions
- “Last Will”

Vocabulary and Encoding

- Packet type = {CONNECT, CONNACK, PUBLISH, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK, PINGREQ, PINGRESP, DISCONNECT}

- Binary encoding of MQTT protocol information
- Two byte fixed header (packet type, QoS, ...)
- variable length header (depending on topic name length)

Topic Example



Sub.	Topic	Messages from device ID
1	temp/r2/devB	4
2	temp/+/+	1, 2, 3, 4
3	temp/+/devB	2, 4
4	temp/r1/#	1, 2

Architecture

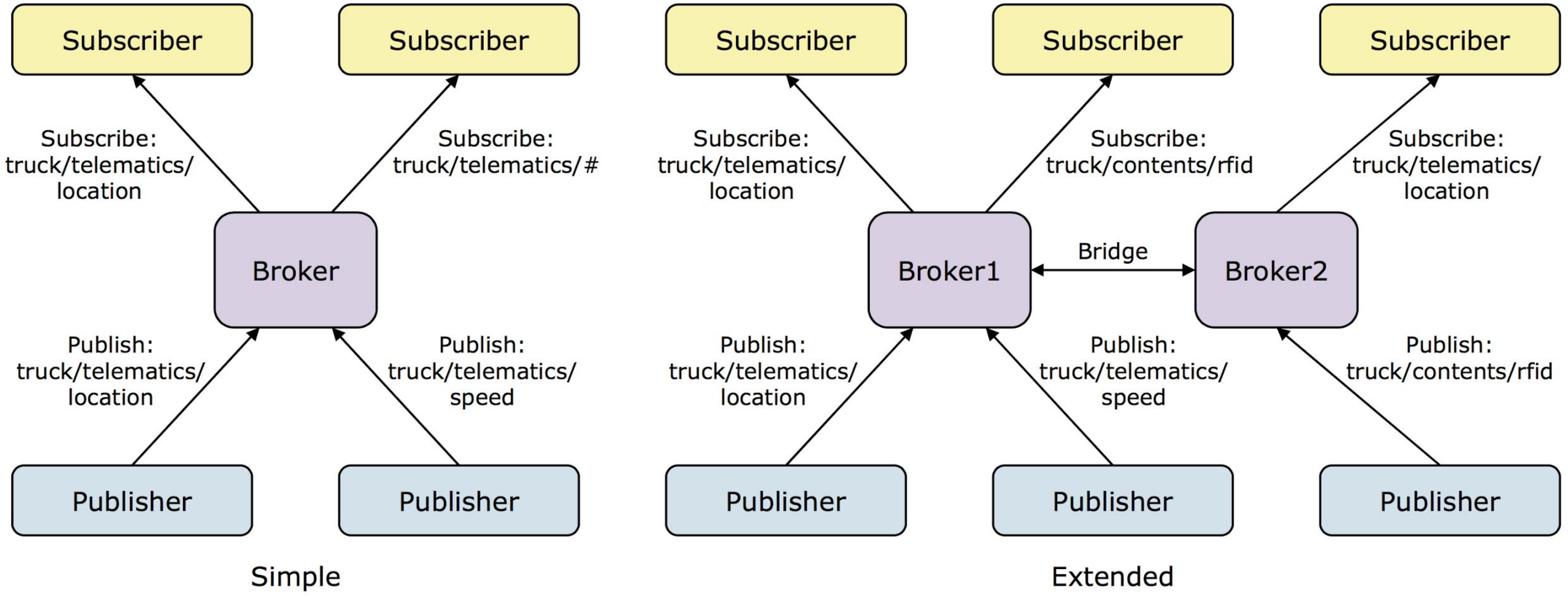
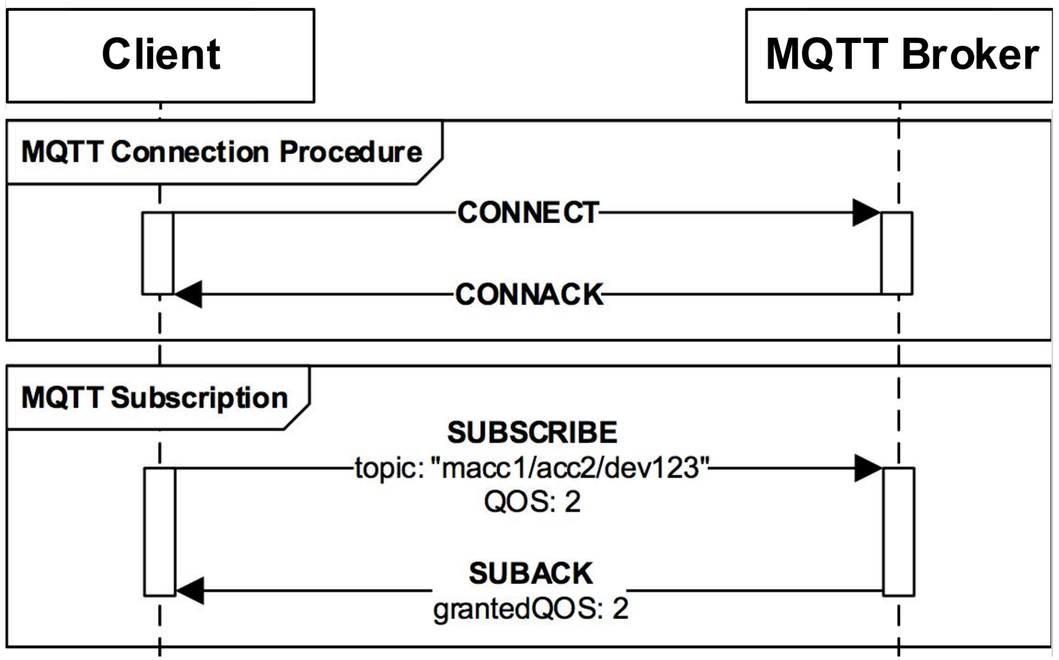
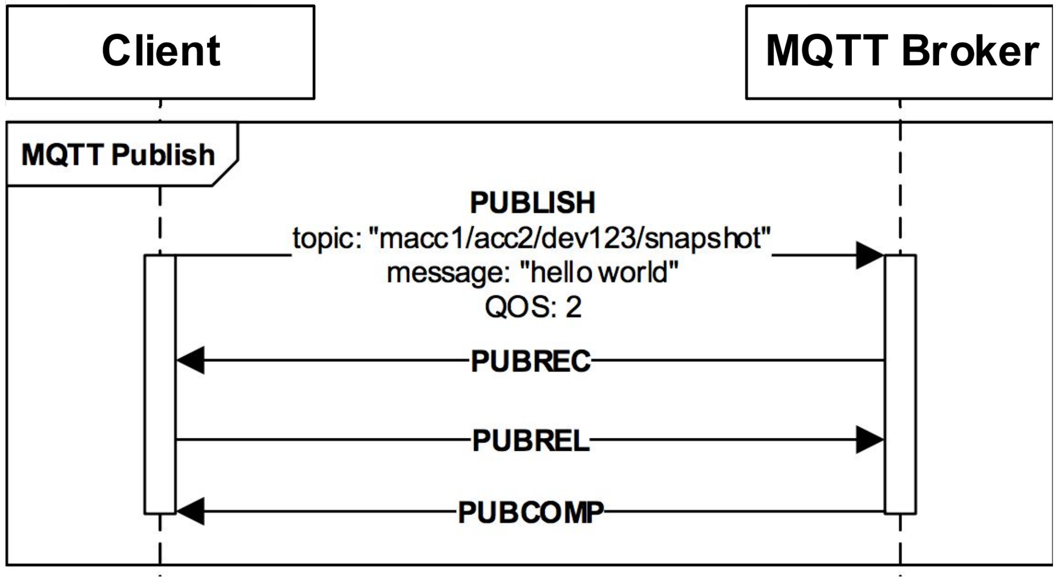

Subscriber
Subscriber
Subscribe: truck/telematics/location
Subscribe: truck/telematics/#
Broker
Publish: truck/telematics/location
Publish: truck/telematics/speed
Publisher
Publisher
Simple
Subscriber
Subscriber
Subscriber
Subscribe: truck/telematics/location
Subscribe: truck/contents/rfid
Subscribe: truck/telematics/location
Broker1
Bridge
Broker2
Publish: truck/telematics/location
Publish: truck/telematics/speed
Publish: truck/contents/rfid
Publisher
Publisher
Publisher
Extended

Image: Lampkin, V. et al. (2012) *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*. IBM Redbooks.

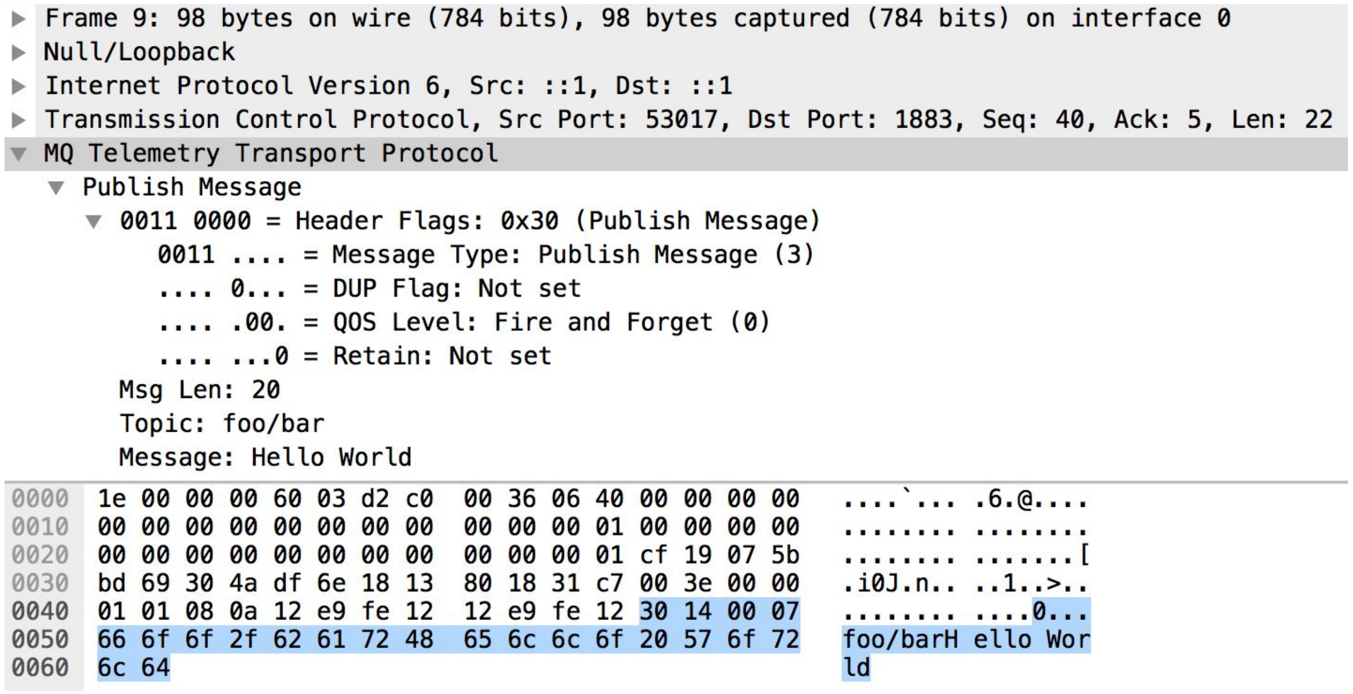
Subscription



Publish QoS 2



MQTT Publish on Wire



```
▶ Frame 9: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 53017, Dst Port: 1883, Seq: 40, Ack: 5, Len: 22
▼ MQ Telemetry Transport Protocol
  ▼ Publish Message
    ▼ 0011 0000 = Header Flags: 0x30 (Publish Message)
        0011 .... = Message Type: Publish Message (3)
        .... 0... = DUP Flag: Not set
        .... .00. = QOS Level: Fire and Forget (0)
        .... ...0 = Retain: Not set
      Msg Len: 20
      Topic: foo/bar
      Message: Hello World

0000  1e 00 00 00 60 03 d2 c0  00 36 06 40 00 00 00 00   ....`... .6.@....
0010  00 00 00 00 00 00 00 00  00 00 00 01 00 00 00 00   ........ ........
0020  00 00 00 00 00 00 00 00  00 00 00 01 cf 19 07 5b   ........ .......[
0030  bd 69 30 4a df 6e 18 13  80 18 31 c7 00 3e 00 00   .i0J.n.. ..1..>..
0040  01 01 08 0a 12 e9 fe 12  12 e9 fe 12 30 14 00 07   ........ ....0...
0050  66 6f 6f 2f 62 61 72 48  65 6c 6c 6f 20 57 6f 72   foo/barH ello Wor
0060  6c 64                                              ld
```

MQTT-SN

- MQTT for Sensor Networks
- (More) lightweight substandard
- Transport: TCP ➔ UDP
- Topic indexing: `europe/finland/factoryA/deviceC/temp` ➔ `123`
- Discovery procedure

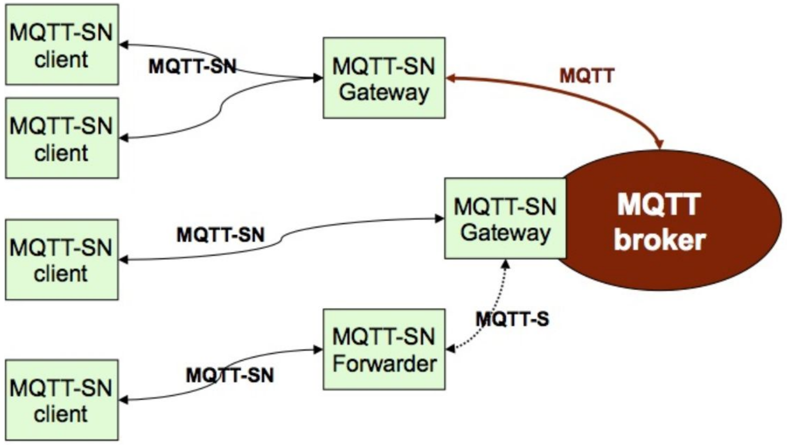


Image: http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf

Constrained Application Protocol

- RESTful, designed for constrained M2M
- usually over UDP
- Request-respond based

Features

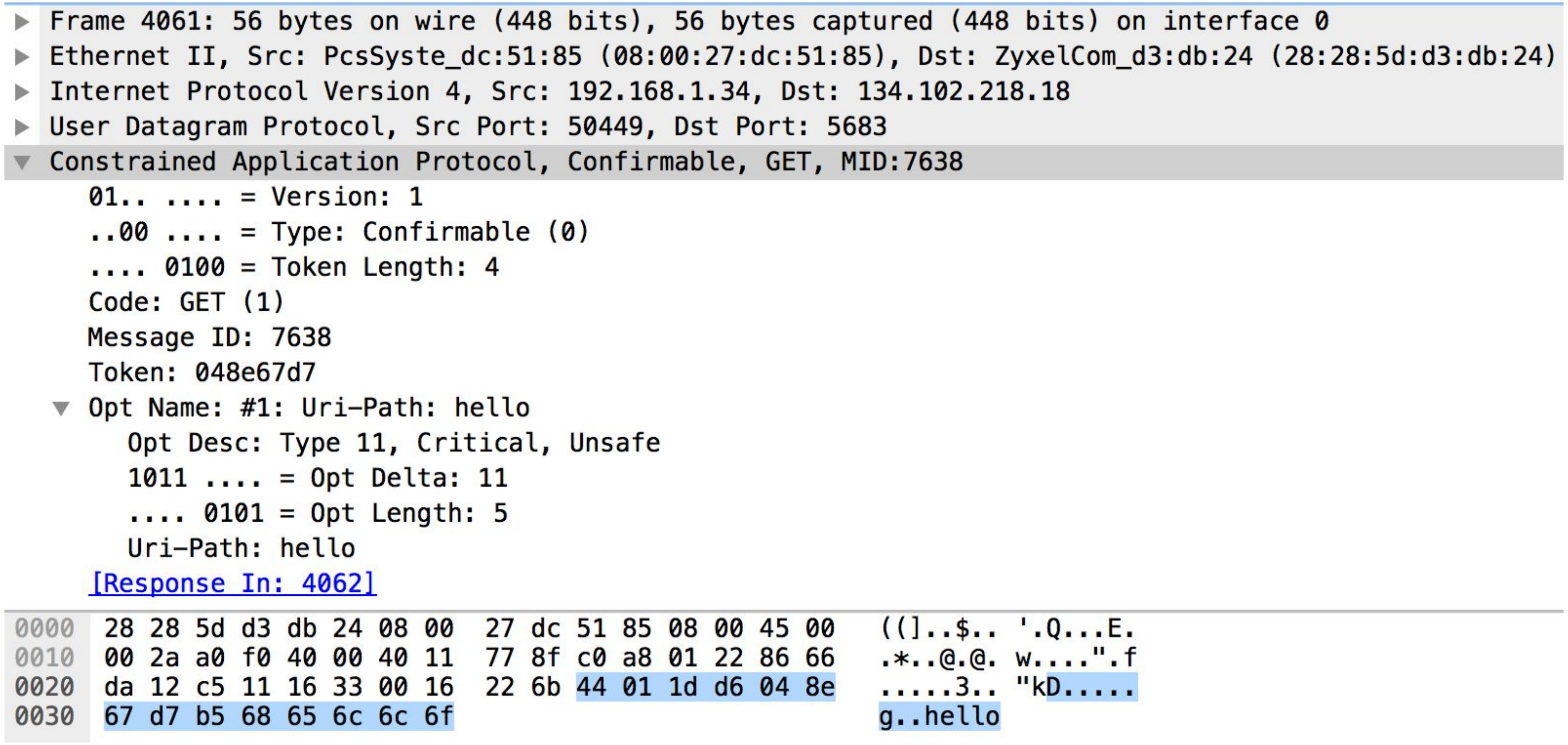
- Built-in discovery
- UDP binding for both unicast and multicast
  - In UDP binding a socket is reserved for sending and receiving for an application
- Low header overhead and parsing complexity
- Support for URIs and Content-type headers

Vocabulary and encoding

- Packet type = {CON, NON, ACK, RST}

- fixed-length binary header (4 bytes)
    - version: 2-bits
    - type: 2-bits
    - token-length: 4-bits
    - code: 8-bits
    - Message ID: 16-bits
- variable-length token value (0-8 bytes)
- sequence of zero or more CoAP Options in Type-Length-Value (TLV) format
- payload (optional)

CoAP confirmable GET



```
▶ Frame 4061: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
▶ Ethernet II, Src: PcsSyste_dc:51:85 (08:00:27:dc:51:85), Dst: ZyxelCom_d3:db:24 (28:28:5d:d3:db:24)
▶ Internet Protocol Version 4, Src: 192.168.1.34, Dst: 134.102.218.18
▶ User Datagram Protocol, Src Port: 50449, Dst Port: 5683
▼ Constrained Application Protocol, Confirmable, GET, MID:7638
    01.. .... = Version: 1
    ..00 .... = Type: Confirmable (0)
    .... 0100 = Token Length: 4
    Code: GET (1)
    Message ID: 7638
    Token: 048e67d7
  ▼ Opt Name: #1: Uri-Path: hello
      Opt Desc: Type 11, Critical, Unsafe
      1011 .... = Opt Delta: 11
      .... 0101 = Opt Length: 5
      Uri-Path: hello
    [Response In: 4062]

0000  28 28 5d d3 db 24 08 00  27 dc 51 85 08 00 45 00   ((]..$.. '.Q...E.
0010  00 2a a0 f0 40 00 40 11  77 8f c0 a8 01 22 86 66   .*..@.@. w....".f
0020  da 12 c5 11 16 33 00 16  22 6b 44 01 1d d6 04 8e   .....3.. "kD.....
0030  67 d7 b5 68 65 6c 6c 6f                            g..hello
```

CoAP response types

```
Client              Server
   |                  |
   |   CON [0xbc90]   |
   | GET /temperature |
   |   (Token 0x71)   |
   +----------------->|
   |                  |
   |   ACK [0xbc90]   |
   |   2.05 Content   |
   |   (Token 0x71)   |
   |     "22.5 C"     |
   |<-----------------+
   |                  |
```

piggybacked

```
Client              Server
   |                  |
   |   CON [0x7a10]   |
   | GET /temperature |
   |   (Token 0x73)   |
   +----------------->|
   |                  |
   |   ACK [0x7a10]   |
   |<-----------------+
   |                  |
   ... Time Passes  ...
   |                  |
   |   CON [0x23bb]   |
   |   2.05 Content   |
   |   (Token 0x73)   |
   |     "22.5 C"     |
   |<-----------------+
   |                  |
   |   ACK [0x23bb]   |
   +----------------->|
   |                  |
```

separate

```
Client              Server
   |                  |
   |   NON [0x7a11]   |
   | GET /temperature |
   |   (Token 0x74)   |
   +----------------->|
   |                  |
   |   NON [0x23bc]   |
   |   2.05 Content   |
   |   (Token 0x74)   |
   |     "22.5 C"     |
   |<-----------------+
   |                  |
```

non-confirmable

MQTT vs. CoAP

Common features

- Aim for low data overhead and little computing efforts
- Promise to work even in restricted network environments

Differences

- MQTT is publish-subscribe oriented, CoAP is request-response oriented
- MQTT on top of TCP, CoAP on top of UDP ➔ reliability effects