Internet of Things

Lecture 6 - CoAP & MQTT

M2M vs. IoT

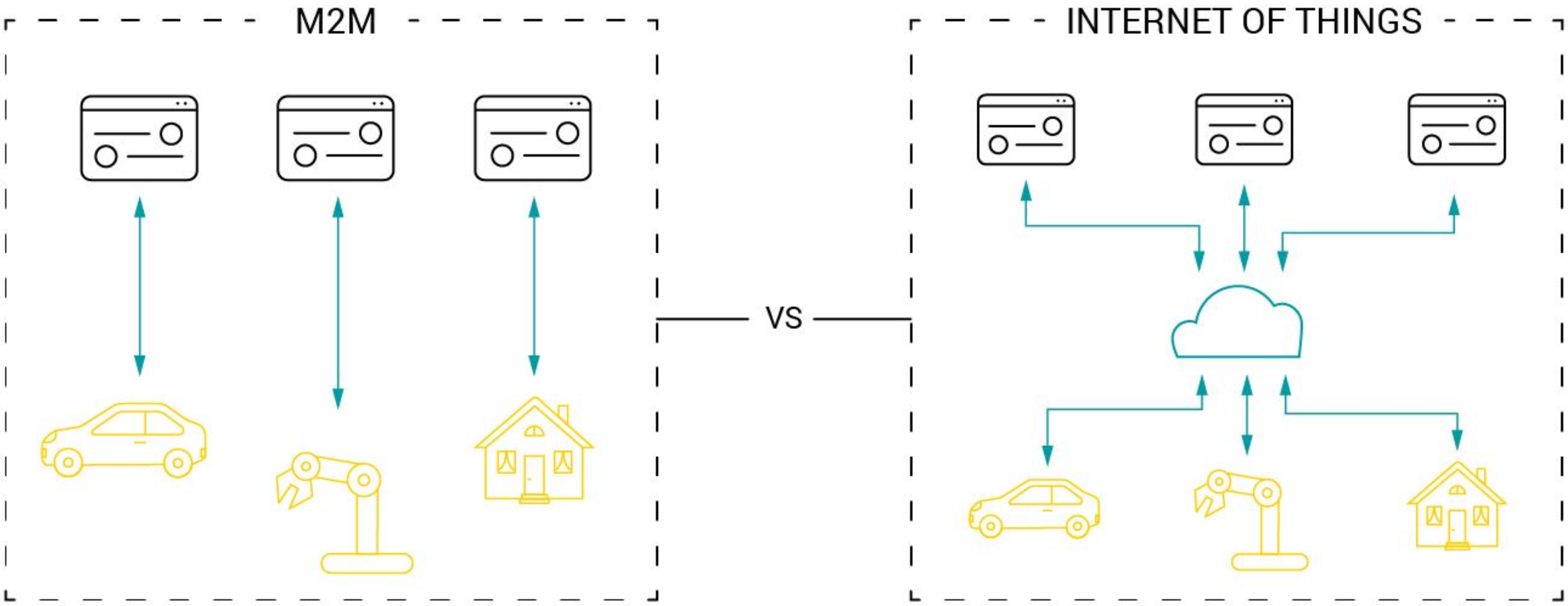


Image source:
https://www.avsystem.com/blog/iot-and-m2m-what-is-the-difference/

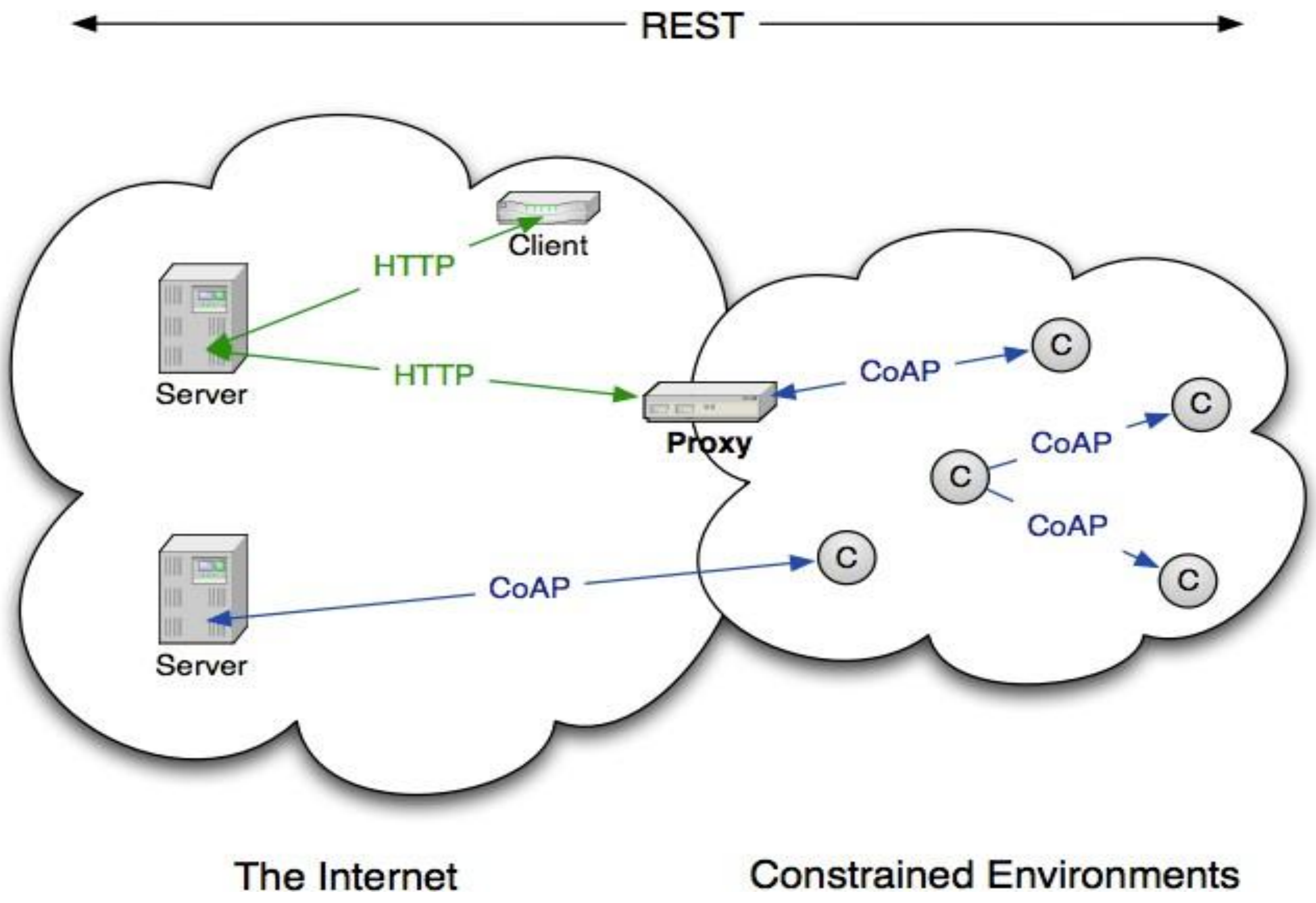
CoAP
Constrained Application Protocol

CoAP:The Web of Things Protocol

- Open IETF Standard
- Compact 4-byte Header
- UDP, SMS, (TCP) Support
- Strong DTLS Security
- Asynchronous Subscription
- Built-in Discovery

CoAP	
DTLS	SMS
UDP	
IP	

IETF



Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

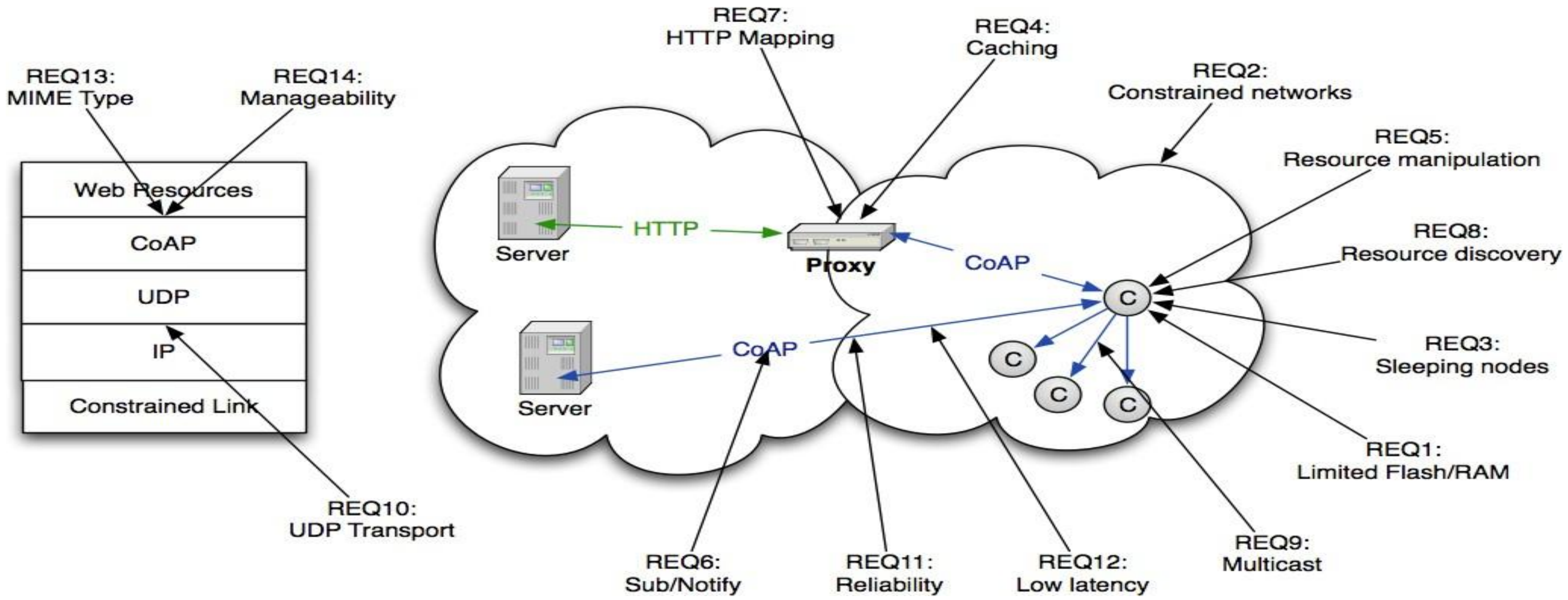
When to use CoAP?

- Your device is constrained and cannot run HTTP or TLS
  - Use CoAP & DTLS
- Your device is powered by battery
  - CoAP is more energy efficient than HTTP
  - UDP

CoAP Features

- Observe at new events happened on sensors or actuators.
- Device management and discoverability from external devices.
- Web protocol used in M2M with constrained requirements
- Asynchronous message exchange
- Low overhead and very simple to parse
- URI and content-type support
- Proxy and caching capabilities

CoAP Design Requirements



Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

What CoAP is (and is not)

CoAP is

- RESTful protocol
- Ideal for constrained devices
- M2M applications
- Easy to translate to/from HTTP

CoAP is not

- a replacement for HTTP
- a HTTP compression
- Restricted to isolated “automation” networks

CoAP Features

- Web transfer protocol
- Asynchronous transaction model
- UDP
- GET, POST, PUT, DELETE methods
- URI support
- Small, simple 4 byte header
- DTLS
- MIME, response codes
- Resource discovery
- Resource observation
- Block transfers

Transaction Model

Transport

- UDP, DTLS
- CoAP over SMS or TCP possible

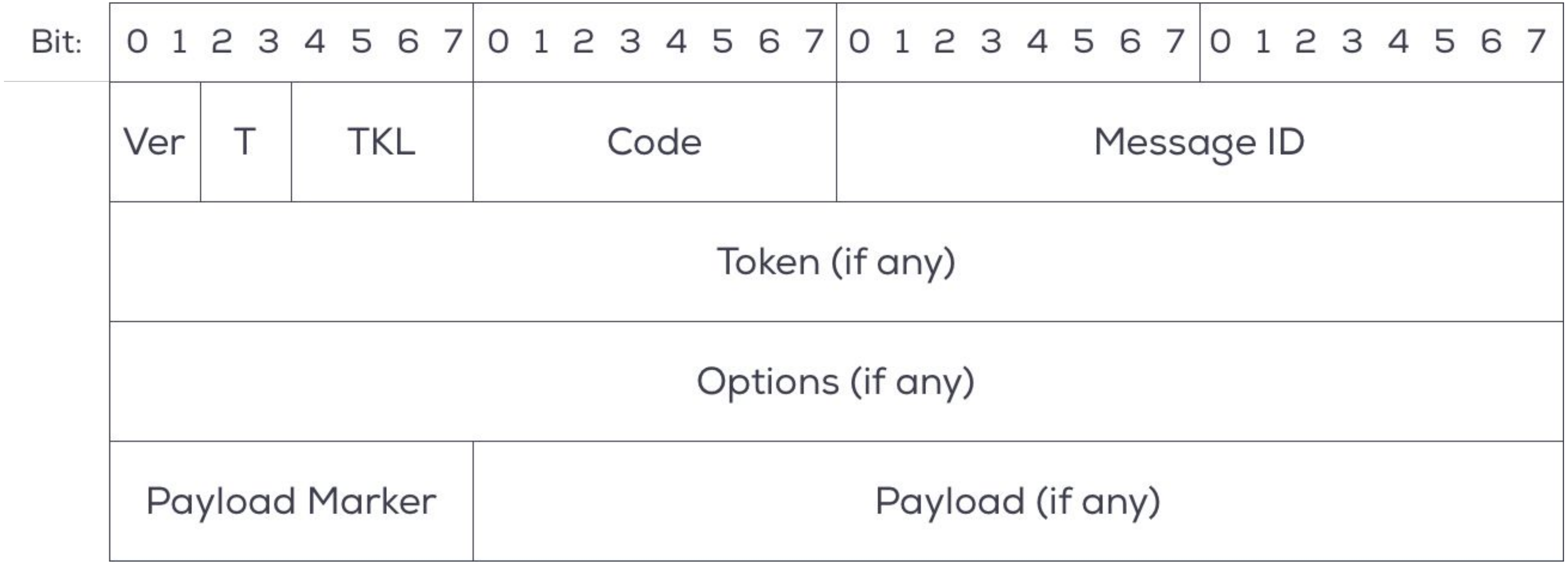
Base Messaging

- Simple message exchange
- Confirmable, Non-Confirmable messages
- Acknowledgement, Reset

REST Semantics

- REST Request/Response piggybacked on CoAP Messages
- Method, Response Code, Options

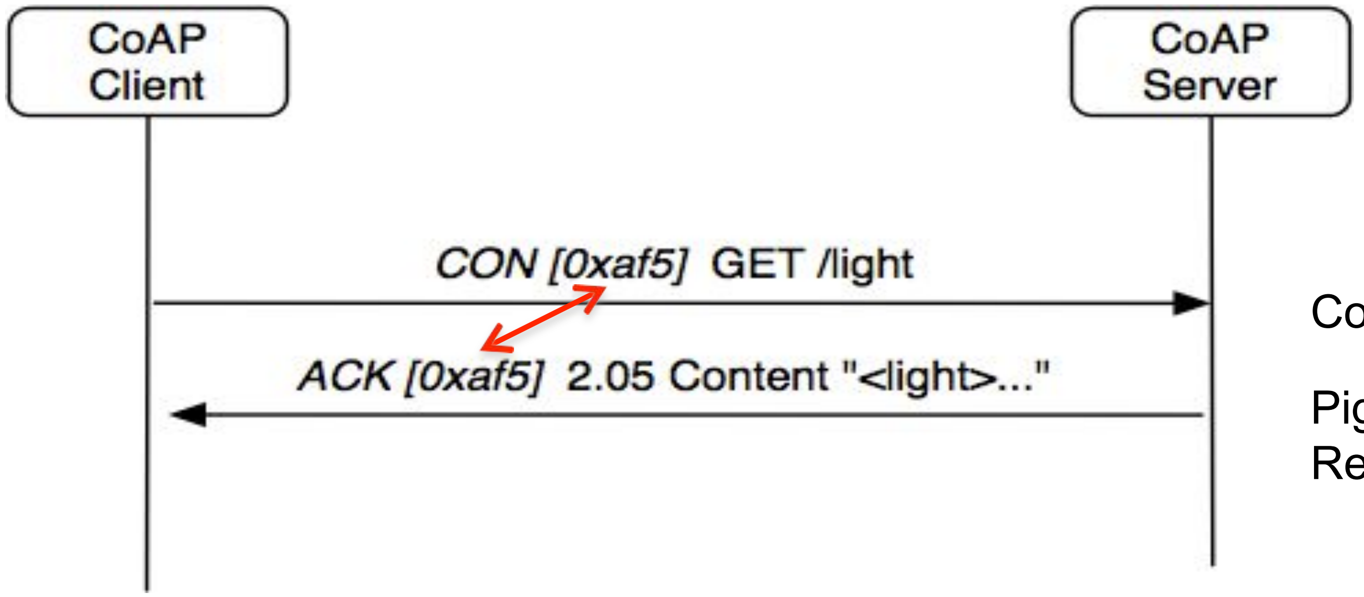
CoAP Message Header



Bit:	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
	Ver		T		TKL				Code								Message ID															
	Token (if any)																															
	Options (if any)																															
	Payload Marker								Payload (if any)																							

Ver: CoAP version - 2 bits
T: message type - 2 bits
TKL: Token Length - 4 bits
Code: response code - 8 bits
Message ID: 16 bits
Token: 0-8 bytes

Confirmable Request Example

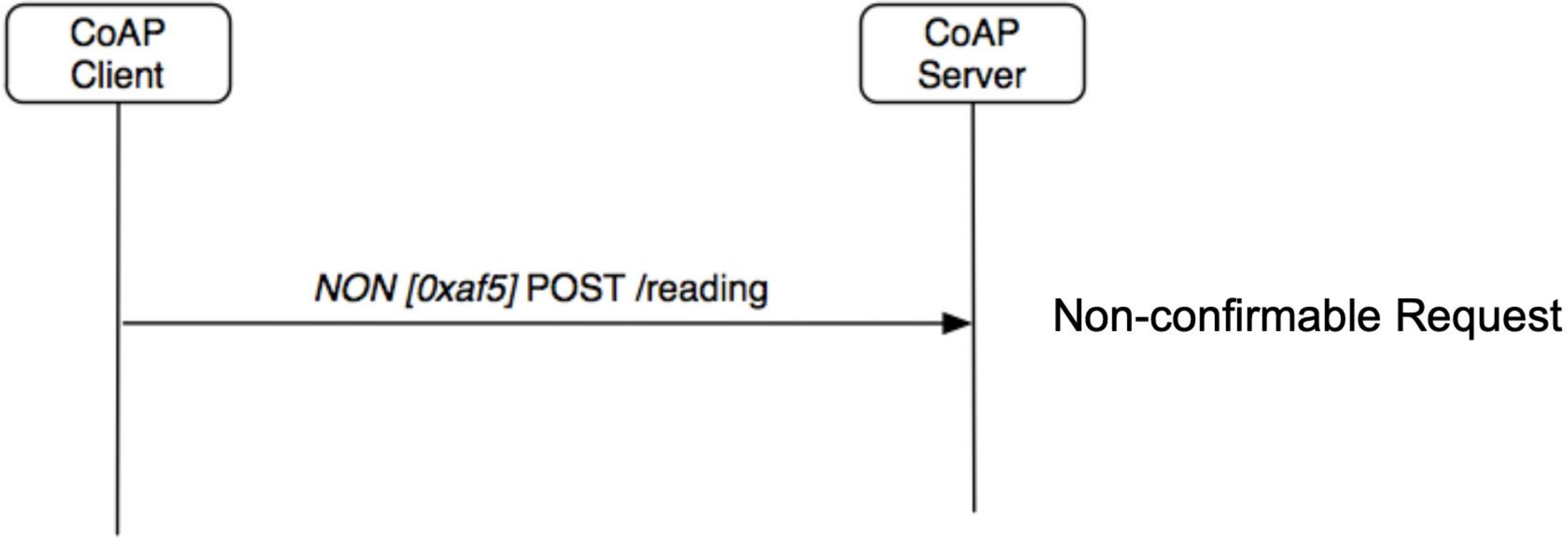


Confirmable Request

Piggy-backed Response

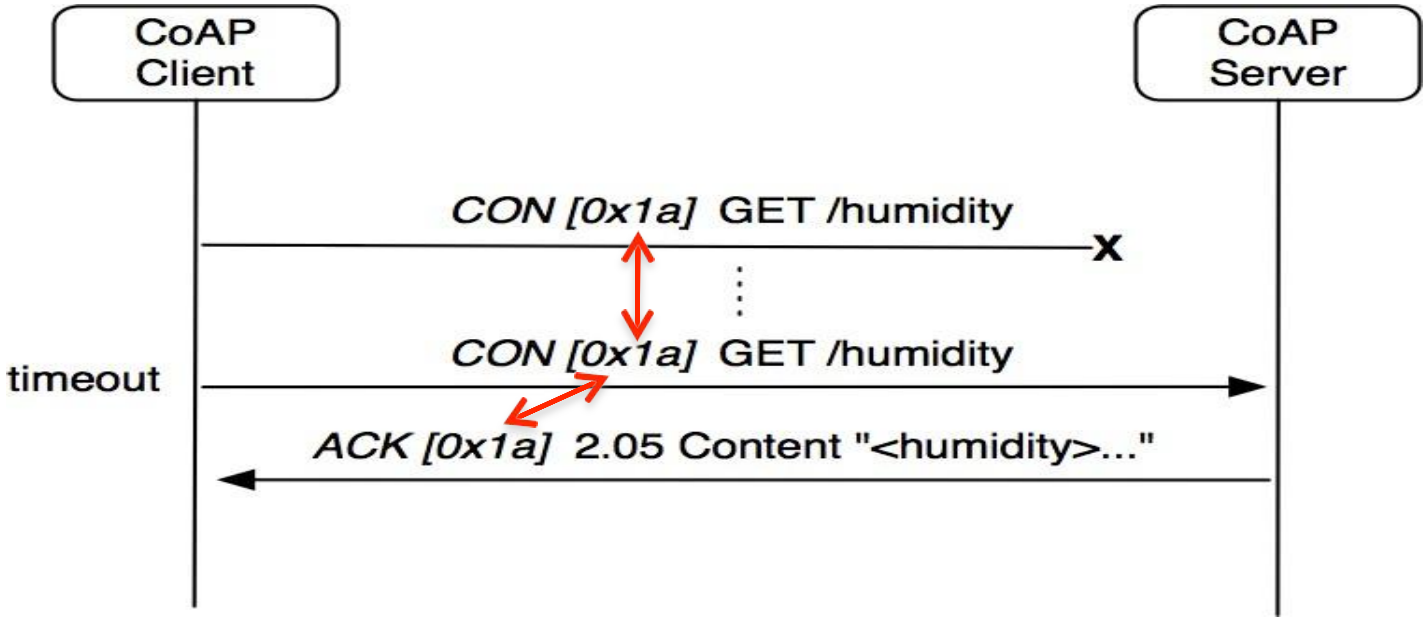
Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

Non-confirmable Request Example



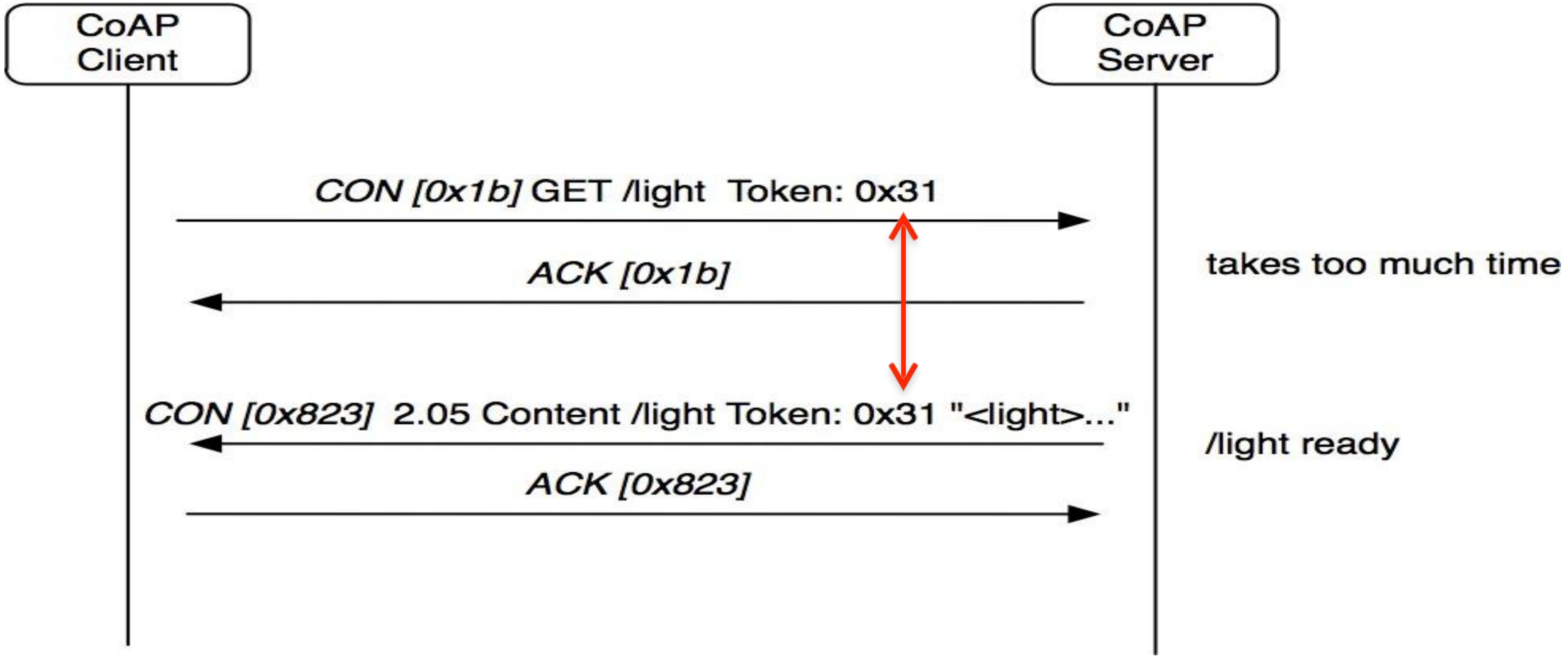
Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

Dealing with Packet Loss



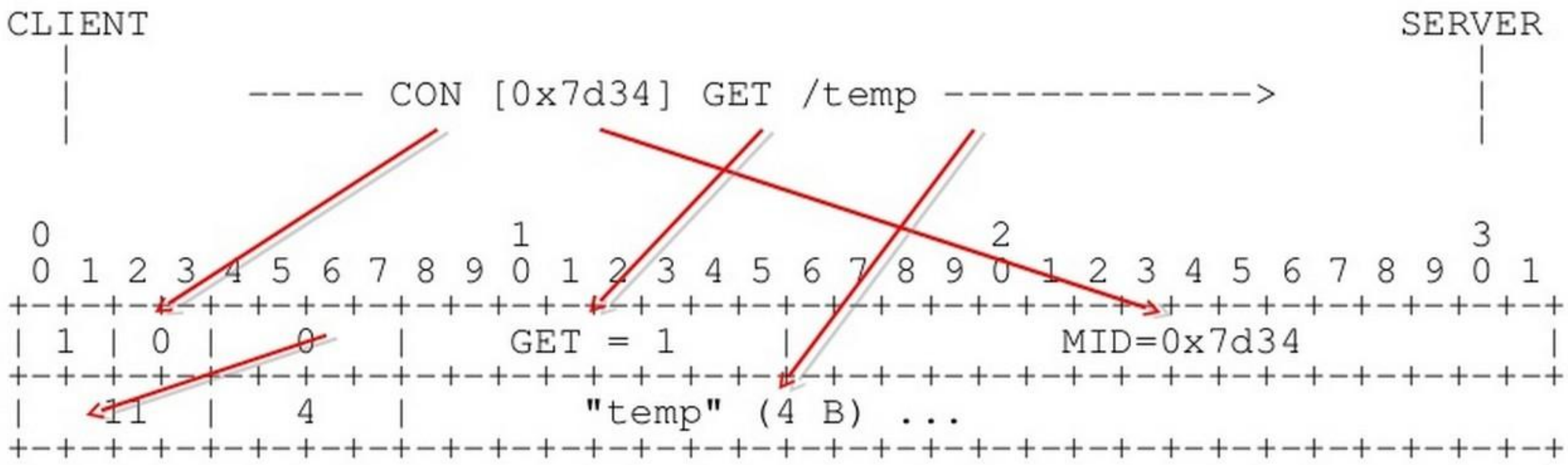
Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

Separate Response


CoAP Client
CoAP Server
CON [0x1b] GET /light Token: 0x31
ACK [0x1b]
takes too much time
CON [0x823] 2.05 Content /light Token: 0x31 "<light>..."
/light ready
ACK [0x823]

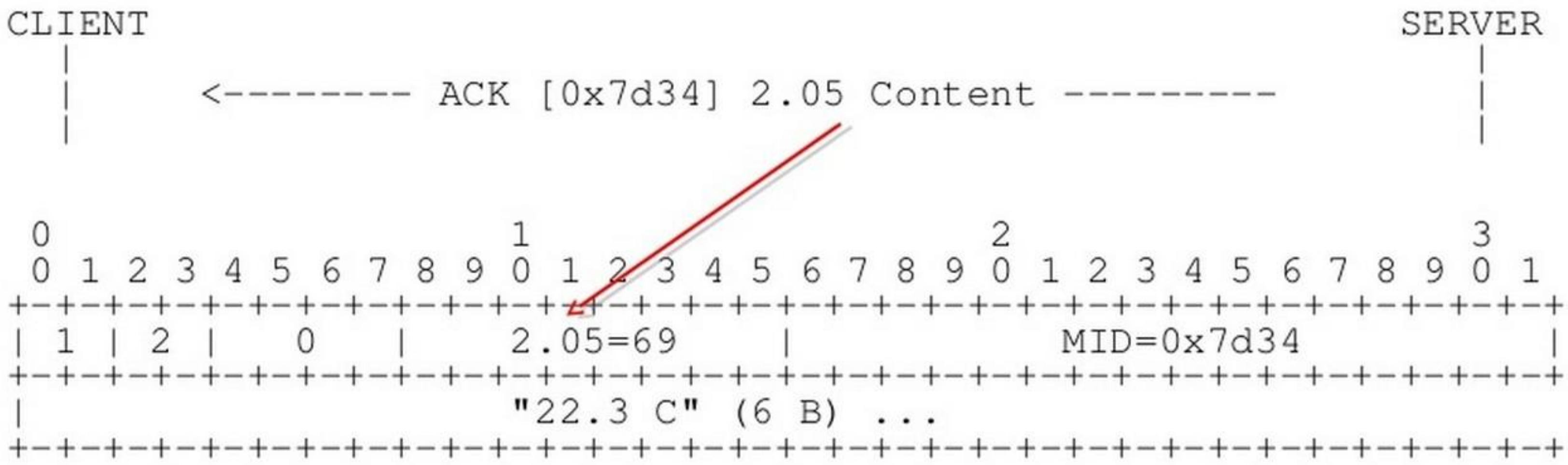
Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

Bits and bytes...



```
CLIENT                                                     SERVER
  |                                                          |
  |      ----- CON [0x7d34] GET /temp -------------->        |
  |                                                          |

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 | 0 |   0   |     GET = 1   |          MID=0x7d34           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  11   |   4   |      "temp" (4 B) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```



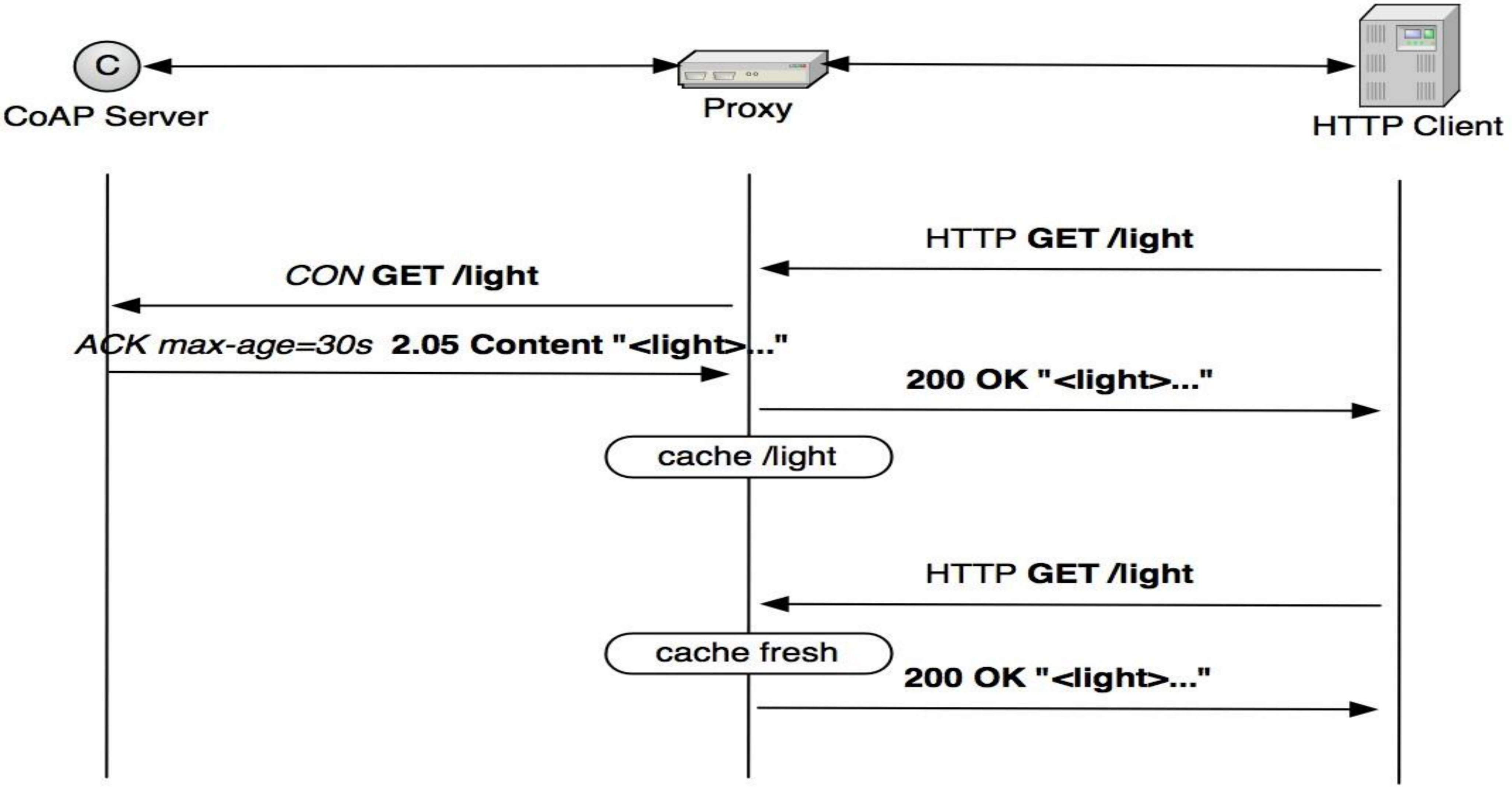
```
CLIENT                                                     SERVER
  |                                                          |
  |    <-------- ACK [0x7d34] 2.05 Content ---------         |
  |                                                          |

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 | 2 |   0   |    2.05=69    |          MID=0x7d34           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   "22.3 C" (6 B) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Caching

- CoAP includes a simple caching model
- Freshness model
  - Max-Age option indicates cache lifetime
- Validation model
- A proxy performs caching
  - On behalf of a constrained node
  - To reduce network load

Proxying and caching



C

CoAP Server

Proxy

HTTP Client

HTTP **GET /light**

CON **GET /light**

ACK max-age=30s **2.05 Content "<light>..."**

200 OK "<light>..."

cache /light

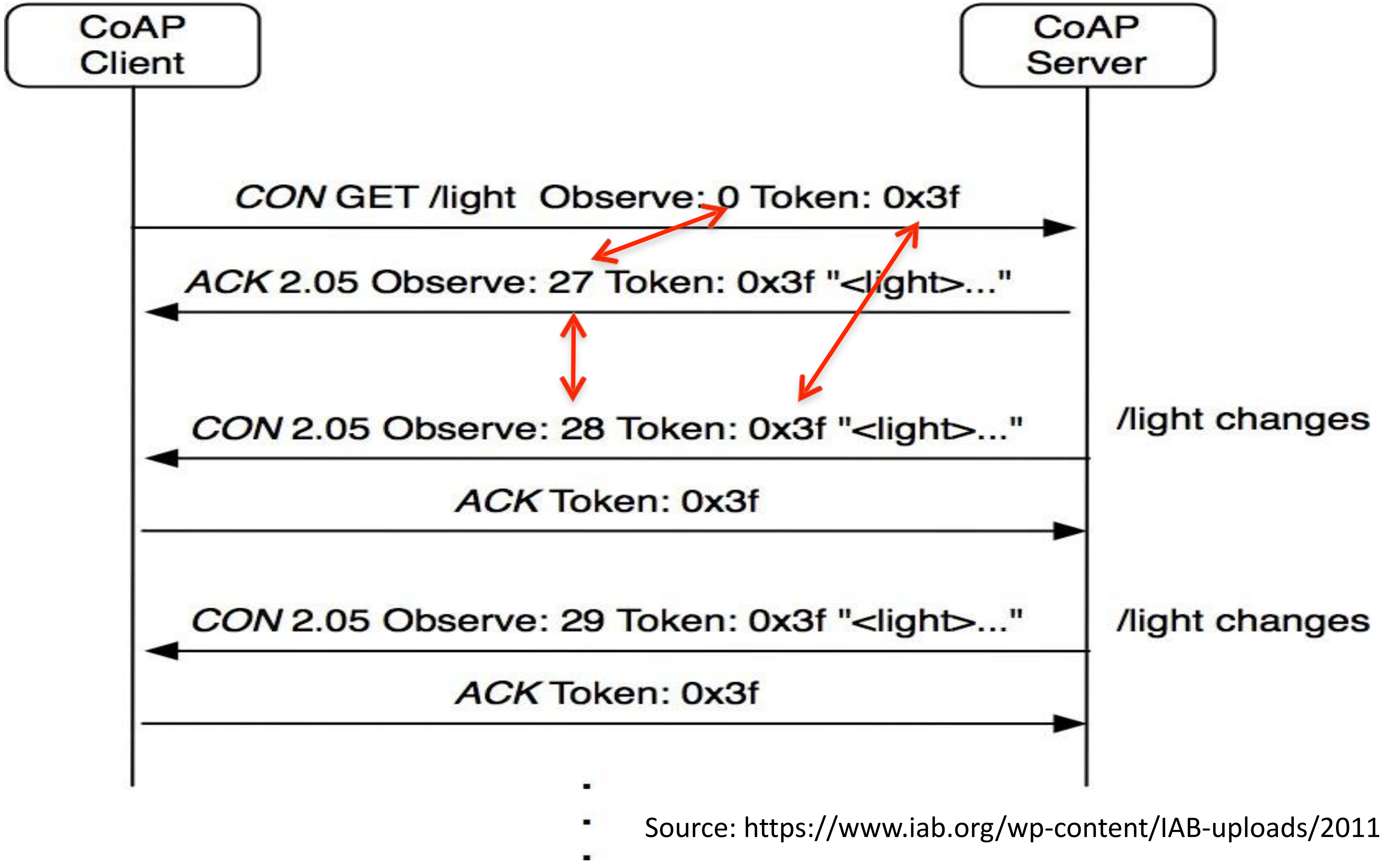
HTTP **GET /light**

cache fresh

200 OK "<light>..."

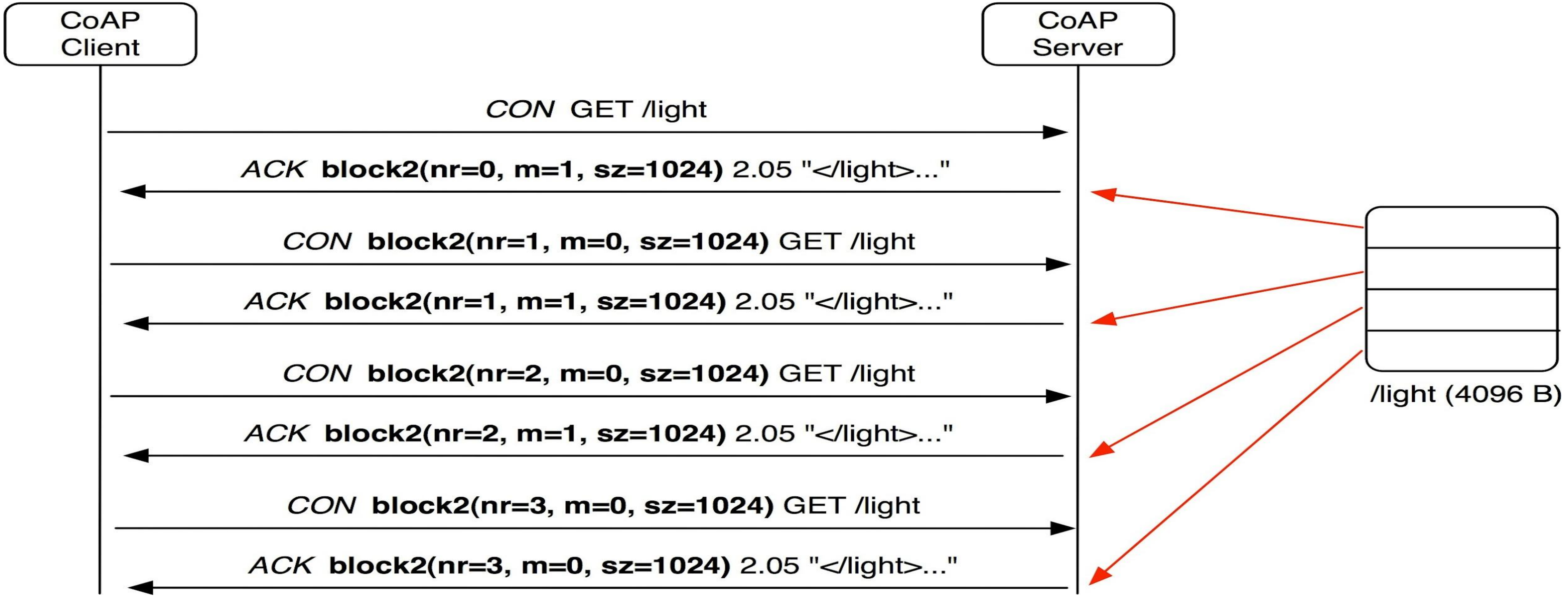
Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

Observing Resources


CoAP Client
CoAP Server
CON GET /light Observe: 0 Token: 0x3f
ACK 2.05 Observe: 27 Token: 0x3f "<light>..."
CON 2.05 Observe: 28 Token: 0x3f "<light>..."
/light changes
ACK Token: 0x3f
CON 2.05 Observe: 29 Token: 0x3f "<light>..."
/light changes
ACK Token: 0x3f

Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

Block transfer


CoAP Client
CoAP Server
CON GET /light
ACK **block2(nr=0, m=1, sz=1024)** 2.05 "</light>..."
CON **block2(nr=1, m=0, sz=1024)** GET /light
ACK **block2(nr=1, m=1, sz=1024)** 2.05 "</light>..."
CON **block2(nr=2, m=0, sz=1024)** GET /light
ACK **block2(nr=2, m=1, sz=1024)** 2.05 "</light>..."
CON **block2(nr=3, m=0, sz=1024)** GET /light
ACK **block2(nr=3, m=0, sz=1024)** 2.05 "</light>..."
/light (4096 B)

Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

Getting Started with CoAP

There are many open source implementations available

- mbed includes CoAP support
- Java CoAP Library Californium, jCoAP Java Library
- C CoAP Library Erbium, libCoAP C Library, OpenCoAP C Library
- TinyOS and Contiki include CoAP support

Firefox has a CoAP plug-in called Copper, Chrome plug-in

Wireshark has CoAP dissector support

CoAP is already part of many commercial products/systems

- ARM Sensinode NanoService
- RTX 4100 WiFi Module

MQTT
Message Queueing Telemetry Transport

MQTT

- Message Queueing Telemetry Transport
- Machine-to-machine (M2M)/"Internet of Things" connectivity protocol
- Invented by Dr. Andy Stanford-Clark of IBM and Arlen Nipper of Arcom (now Eurotech) in 1999
- OASIS standard in 2013
- ISO recommendation (ISO/IEC 20922)
- Public and royalty-free license
- Used by Amazon Web Services, IBM WebSphere MQ, Microsoft Azure IoT, Adafruit, Facebook Messenger etc.

MQTT Features

- Small code footprint
- Ideal if processor or memory resources are limited
- Ideal if bandwidth is low or network is unreliable
- Publish/subscribe message exchange pattern
- Works on top of TCP/IP
- Quality of service levels: at most once, at least once, exactly once
- Client libraries for Android, Arduino, C, C++, C#, Java, JavaScript, .NET etc.
- Security: authentication using username and password, encryption using SSL/TLS
- Support for persistent messages stored on the broker

Applications

- Home automation (e.g. smart lightning, smart metering)
- Healthcare
- Mobile phone apps (e.g. messaging, monitoring)
- Industrial automation
- Automotive
- General IoT applications

Publish/Subscribe

- Multiple clients connect to a broker and subscribe to topics
- Clients connect to the broker and publish messages to topics.

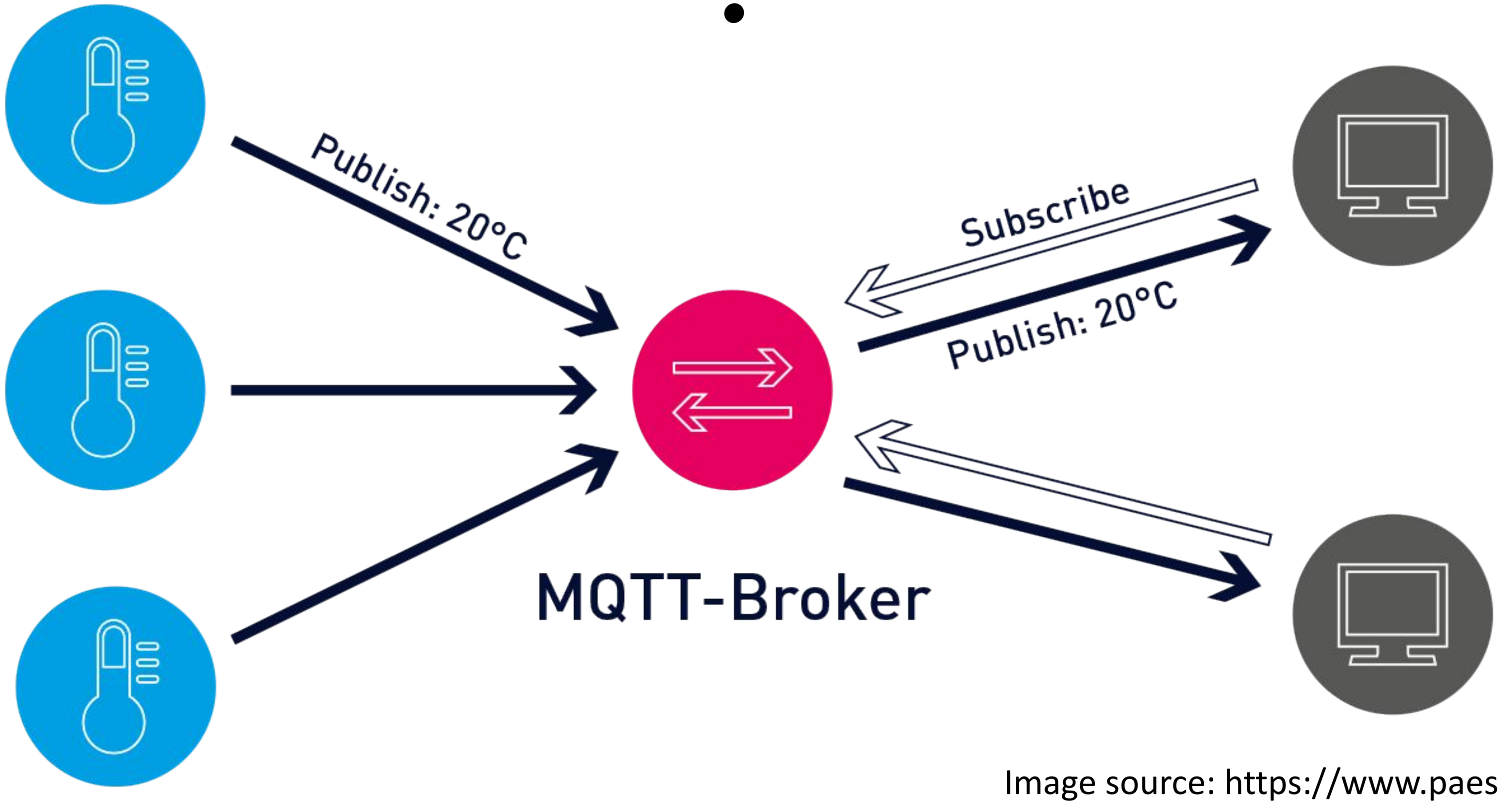


Image source: https://www.paessler.com/it-explained/mqtt

Publish/Subscribe

- Topics are treated as a hierarchy, using a slash (/) as a separator.
  - Example: multiple sensor devices may publish temperature readings on the topic:
  - *sensors/DEVICE_NAME/temperature/NODE_ID*
- A subscription may be to an explicit topic or it may include wildcards.
  - Two wildcards are available: + or #
- Clients can register a custom ‘last will testament’ message
  - This message can be used to signal to subscribers when a device disconnects

Publish/Subscribe

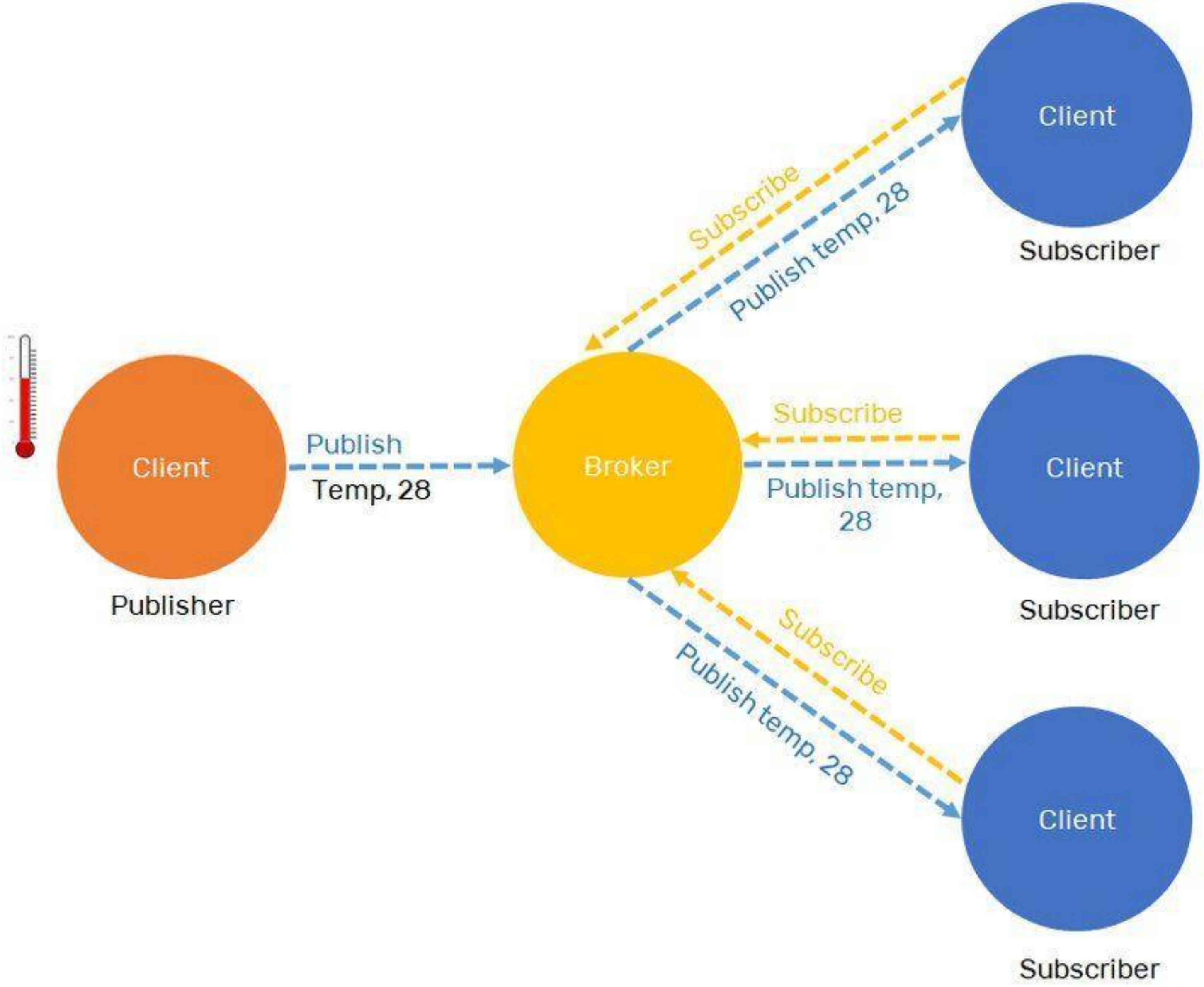


Image source: https://openlabpro.com/guide/introduction-to-mqtt-protocol/

Actions in MQTT

- 4 possible actions:
- Publish:
  - Sends data to broker on a certain topic
- Subscribe:
  - Client subscribes to a certain topic
  - Broker sends SUBACK response & maybe data
- Ping:
  - PINGREQ & PINGRESP messages
  - Ensure that the connection is still working
- Disconnect
  - Publishers & subscribers may disconnect from broker

QoS Levels

- QoS 0 -> At most once
  - Best effort, No Ack
- QoS 1 -> At least once
  - Acked, retransmitted if Ack not received
- QoS 2 -> Exactly once
  - Request to send (Publish), Clear-to-send (Pubrec), message (Pubrel), ack (Pubcomp)

- Retained Messages:
  - Server keeps messages even after sending them to all subscribers
  - New subscribers get the retained messages

MQTT Features

- **Clean Sessions** and **Durable Connections**
  - Clean session flag -> all subscriptions are removed on disconnect
  - Otherwise subscriptions remain in effect after disconnection
  - Subsequent messages with high QoS are stored for delivery after reconnection
- **Last Will Testament**
  - A will or a message that should be published if unexpected disconnection
  - Alarm if the client loses connection

MQTT Features

- Periodic **keep alive** messages -> If a client is still alive
- **Topic Trees -** topics are organized as trees using the / character
  - /# matches all sublevels
  - /+ matches only one sublevel

MQTT vs. CoAP

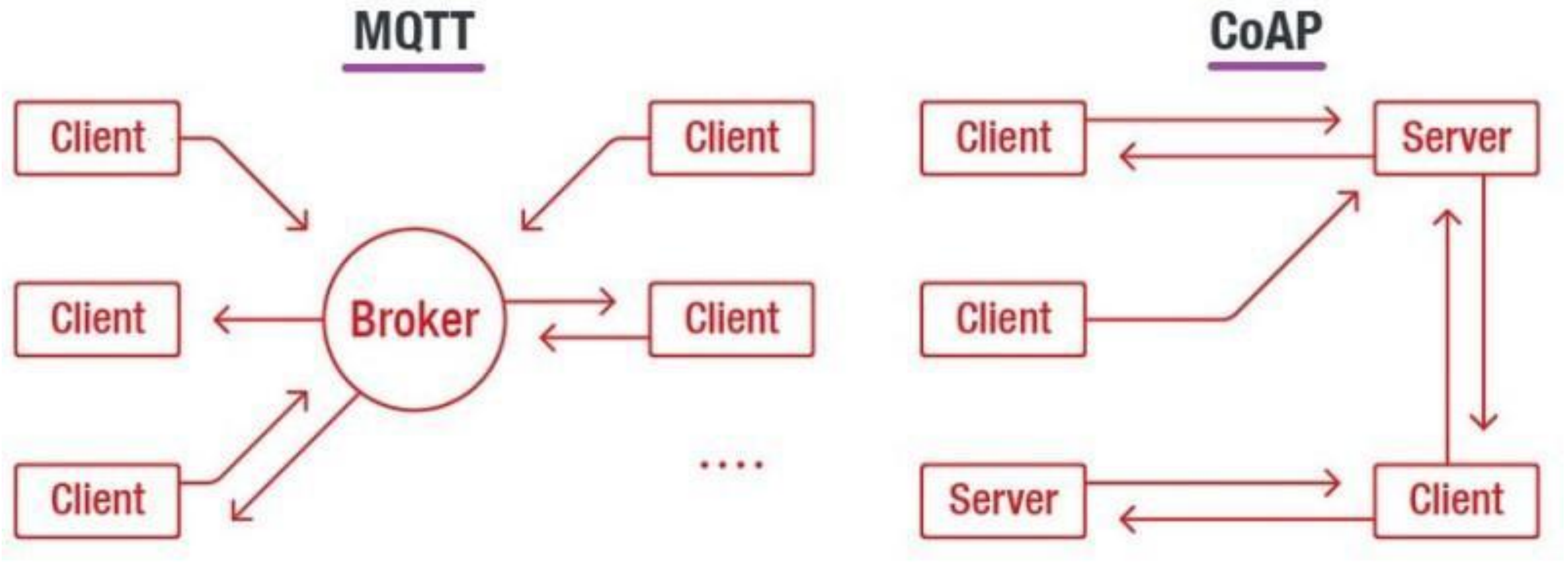


Image Source: https://iotbyhvm.ooo/coap-vs-mqtt/

MQTT vs. CoAP

Features	MQTT	CoAP
Base protocol	TCP	UDP
Model used for communication	Publish-Subscribe	Request-Response Publish-Subscribe
Communication node	M:N	1:1
Power consumption	Higher than CoAP	Lower than MQTT
RESTful	No	Yes
Number of messages type used	16	4
Header size	2 Bytes	4 Bytes
Messaging	Asynchronous	Asynchronous & Synchronous
Reliability	3 Quality of service levels	Confirmable messages
Implementation	Easy to implement Hard to add extensions	Few existing libraries and support
Security	Not defined Can use TLS/SSL	DTLS or IPSec

Source: https://www.pickdata.net/news/mqtt-vs-coap-best-iot-protocol

Bibliography

- CoAP - IETF RFC 7252: https://datatracker.ietf.org/doc/html/rfc7252
- Observing resources in CoAP - RFC 7641: https://datatracker.ietf.org/doc/html/rfc7641
- Block transfers in CoAP - RFC 7959: https://datatracker.ietf.org/doc/html/rfc7959
- https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf
- MQTT standard https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html
- https://www.paessler.com/it-explained/mqtt
- https://iotbyhvm.ooo/coap-vs-mqtt/
- https://www.pickdata.net/news/mqtt-vs-coap-best-iot-protocol