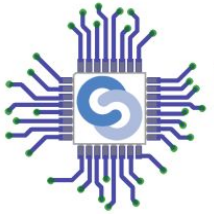


Embedded Systems Laboratory

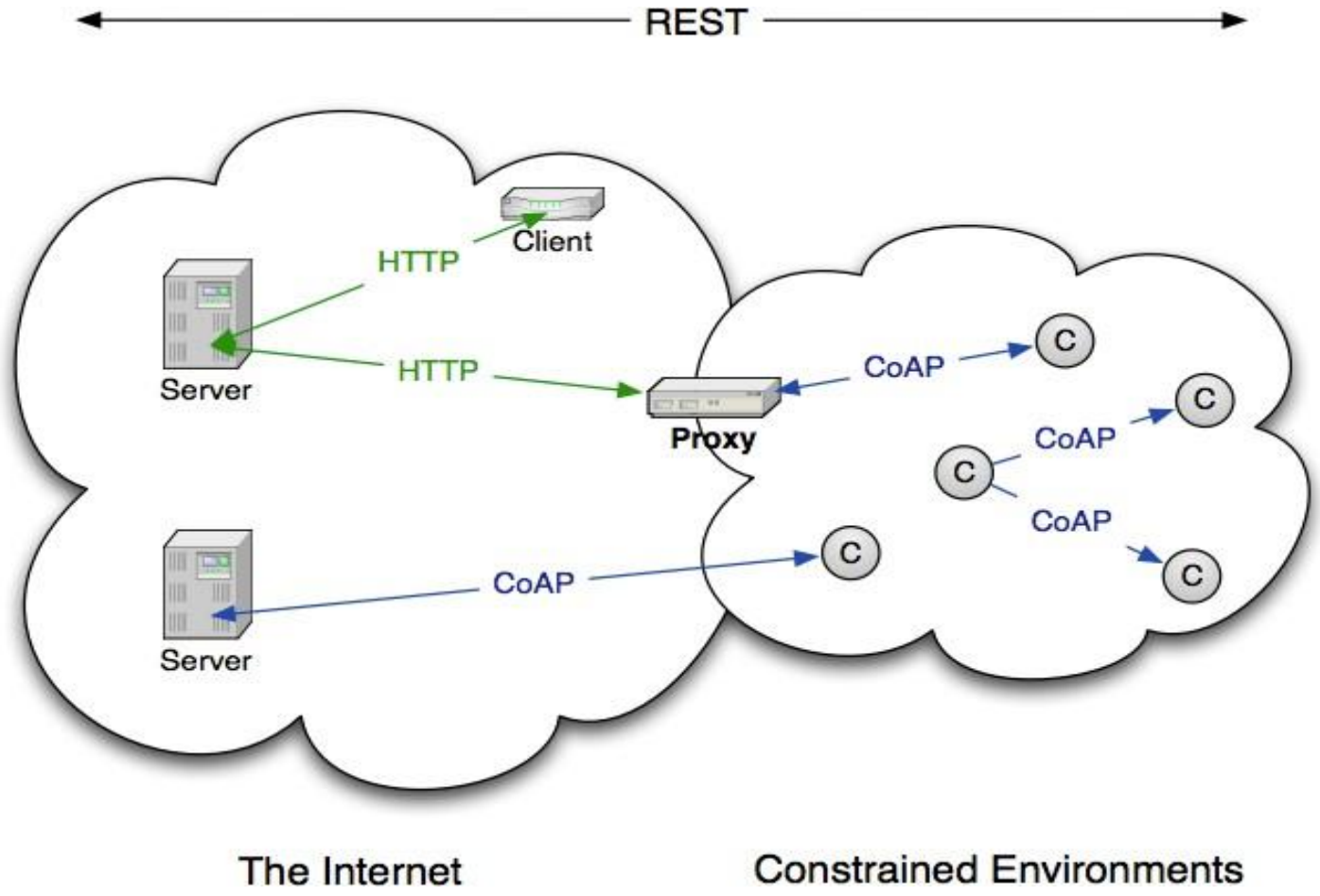
Internet of Things

Lecture 5 - CoAP & MQTT

CoAP
Constrained Application Protocol

CoAP:The Web of Things Protocol

- IETF RFC 7252
- Resource-constrained devices
- 4-byte Header
- Proxy translates CoAP <-> HTTP
- UDP, SMS, TCP
- DTLS Security
- Asynchronous Subscription
- Built-in Discovery

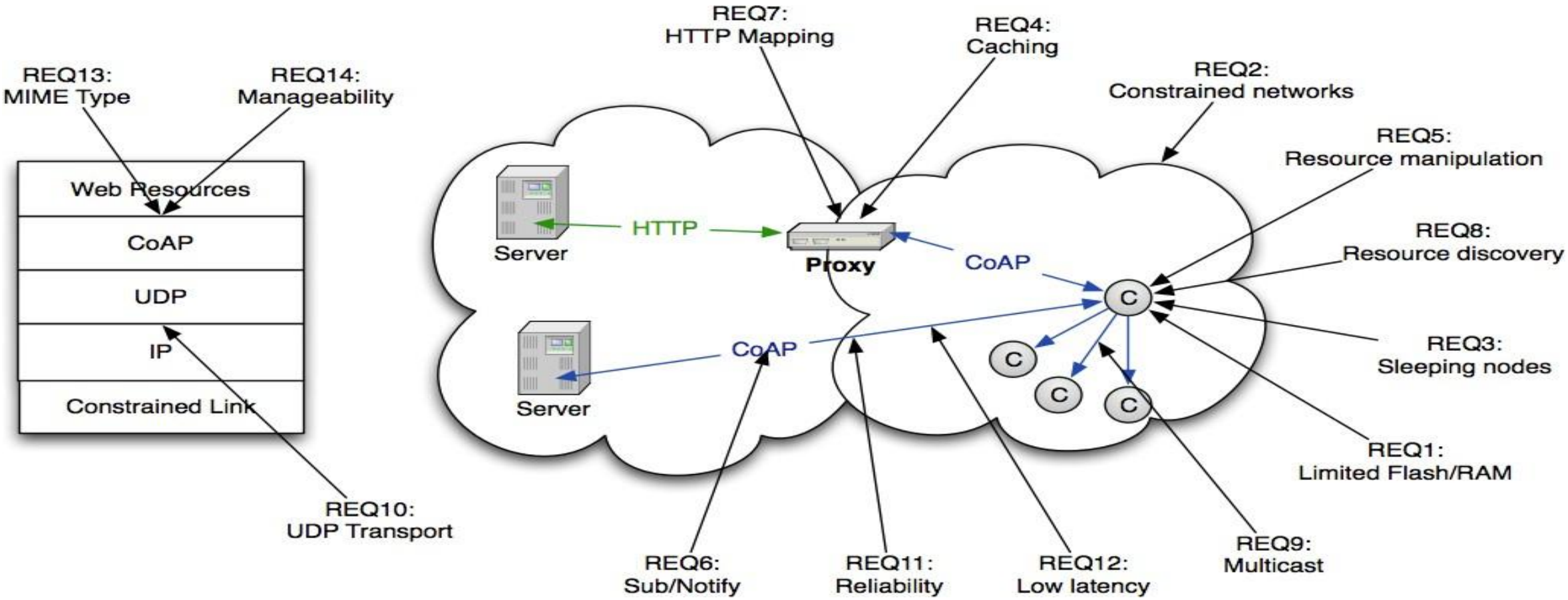


Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

When to use CoAP?

- Your device is constrained and cannot run HTTP or TLS
- Your device is powered by battery
  - CoAP is more energy efficient than HTTP
  - CoAP is stateless, has low overhead
  - DTLS to secure traffic (more lightweight than TLS)
  - UDP to reduce overhead & resource consumption

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
- IoT/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

- Obtain data from sensor nodes
- Device management
- Discover devices
- Discover settings & parameters
- Resource observation
- Block transfers
- Unicast, multicast

CoAP Features

- Web transfer protocol
  - Can be used in the Internet
  - Browser plugin (Copper)
- URI and content-type support
- Messages are similar to HTTP
  - GET, POST, PUT, DELETE methods
- MIME, response codes

CoAP Features

- Low overhead
  - 4 bytes header
- More efficient than HTTP
- Asynchronous transaction model
- UDP
- DTLS
  - authentication & encryption
- Proxy and caching capabilities

Transaction Model

Transport

- UDP, DTLS
- CoAP over SMS or TCP possible

Base Messaging

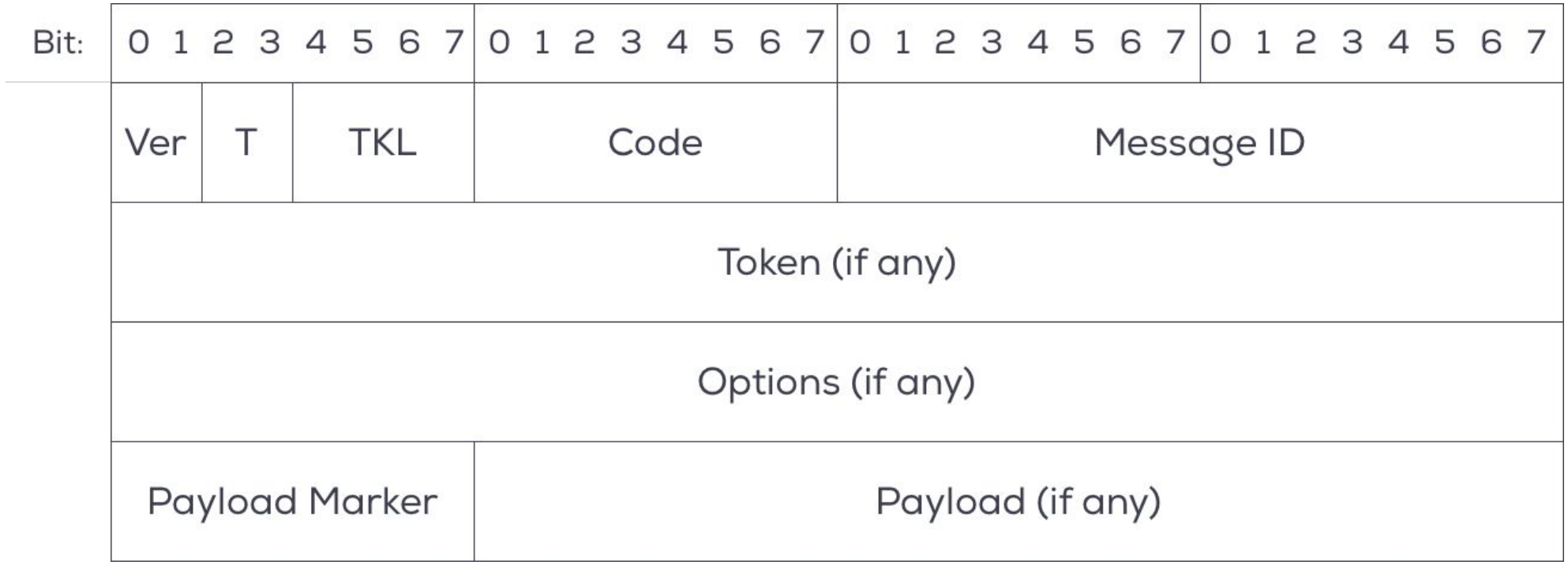
- Simple message exchange
- Confirmable, Non-Confirmable messages
- Acknowledgement, Reset

REST Semantics

- REST Request/Response
- Method, Response Code, Options

Method	CRUD	Action
GET	read	returns requested data
POST	create	creates a new record
PUT	update	updates an existing record
DELETE	delete	deletes an existing record

CoAP Message Header

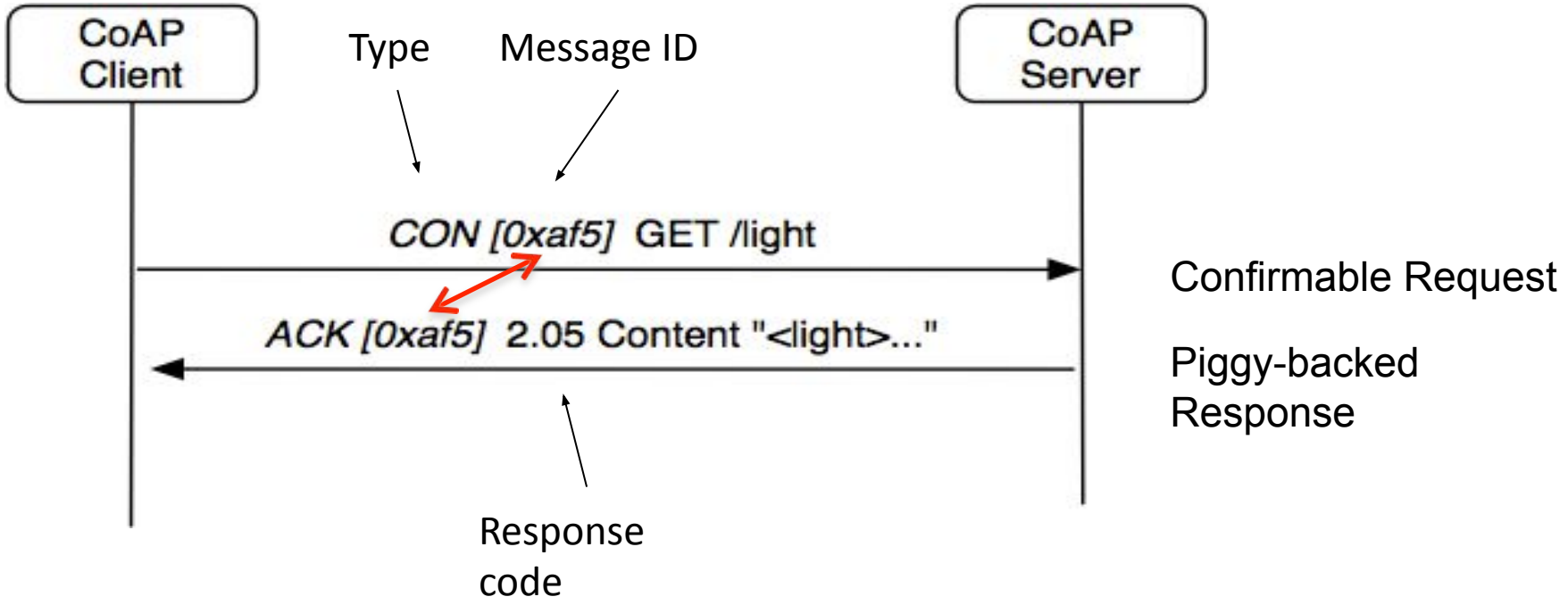


Ver: CoAP version - 2 bits
T: message type - 2 bits
TKL: Token Length - 4 bits
Code: response code - 8 bits
Message ID: 16 bits
Token: 0-8 bytes

Message type:

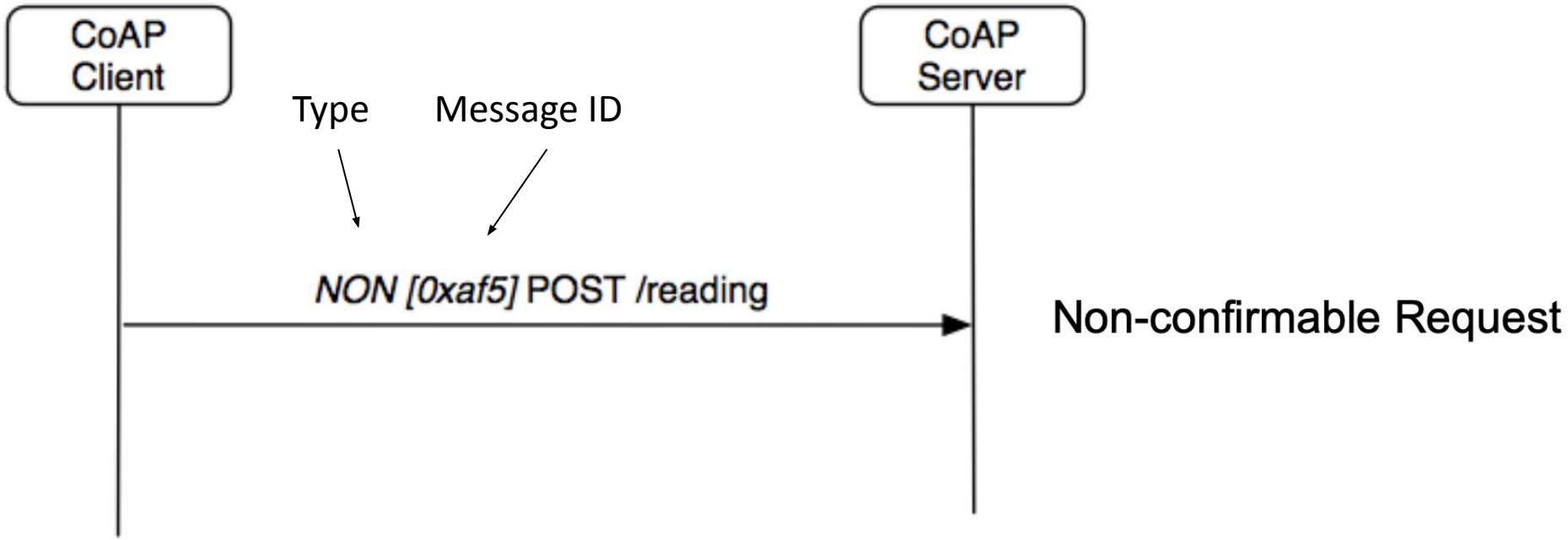
- confirmable (0)
- non-confirmable (1)
- ack (2)
- reset (3)

Confirmable Request Example



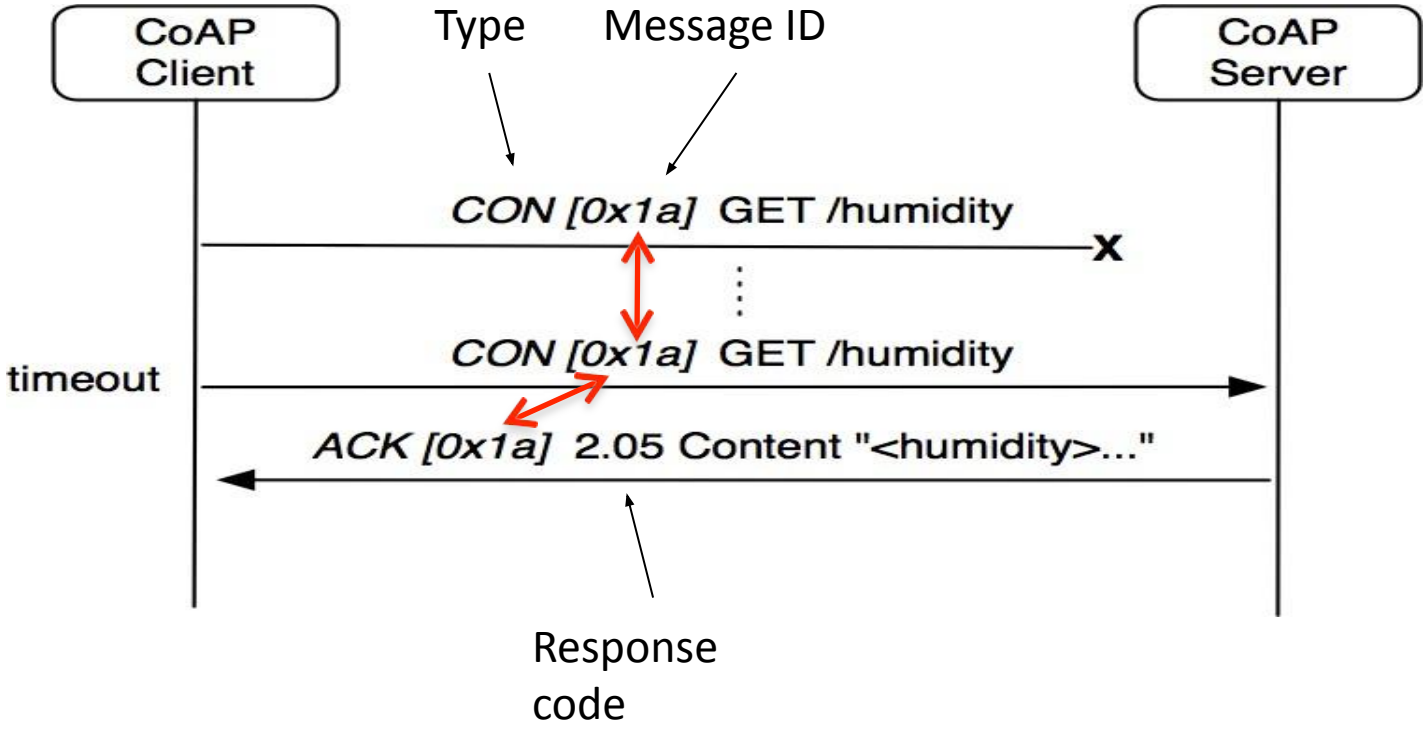
Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

Non-confirmable Request Example



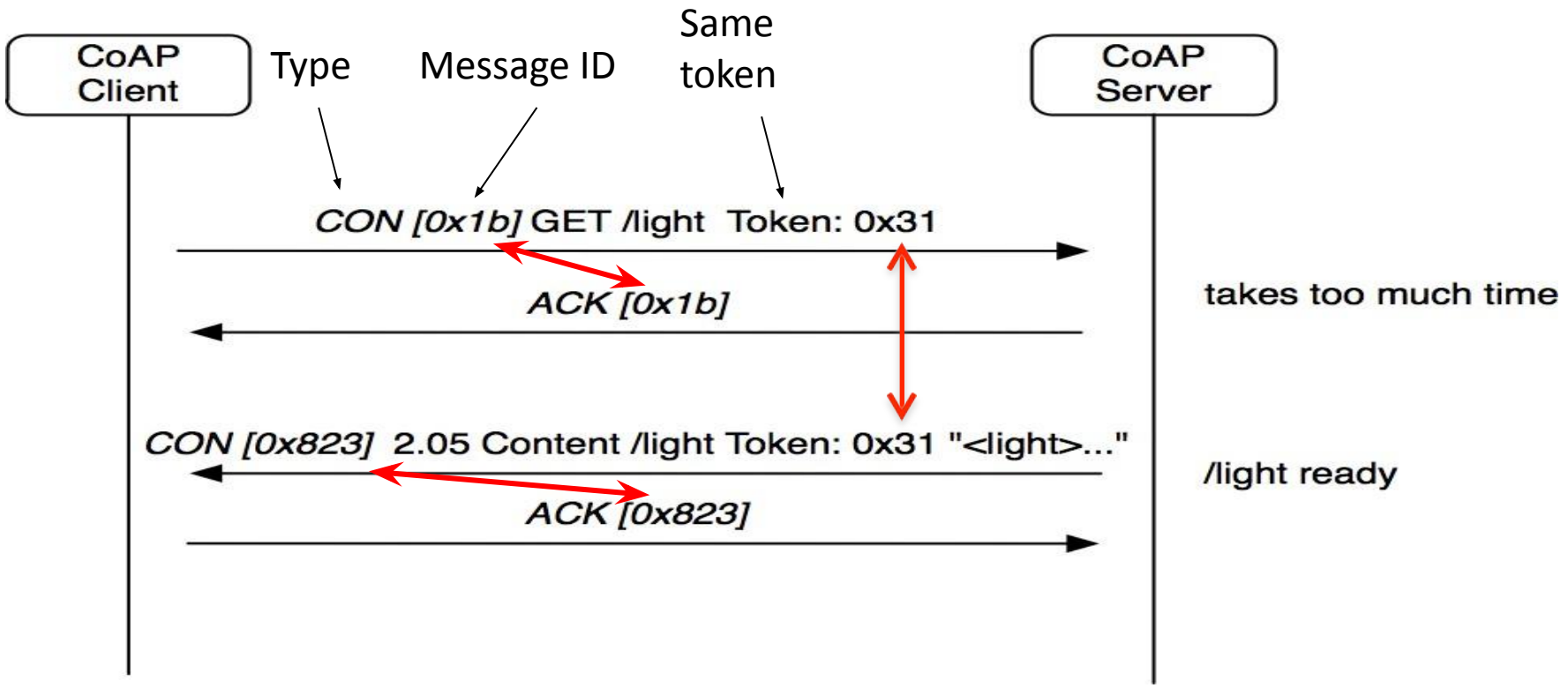
Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

Dealing with Packet Loss



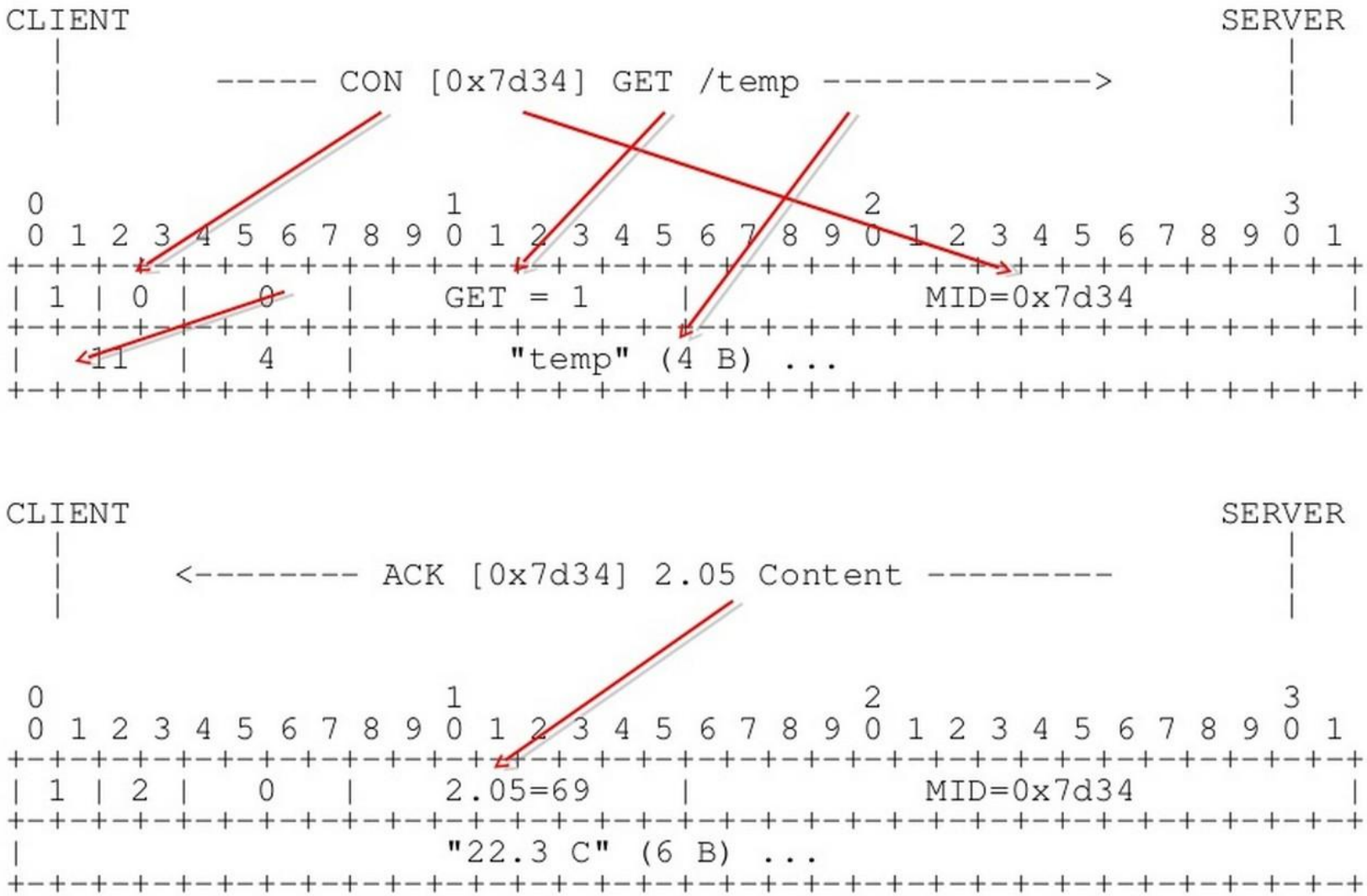
Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

Separate Response



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

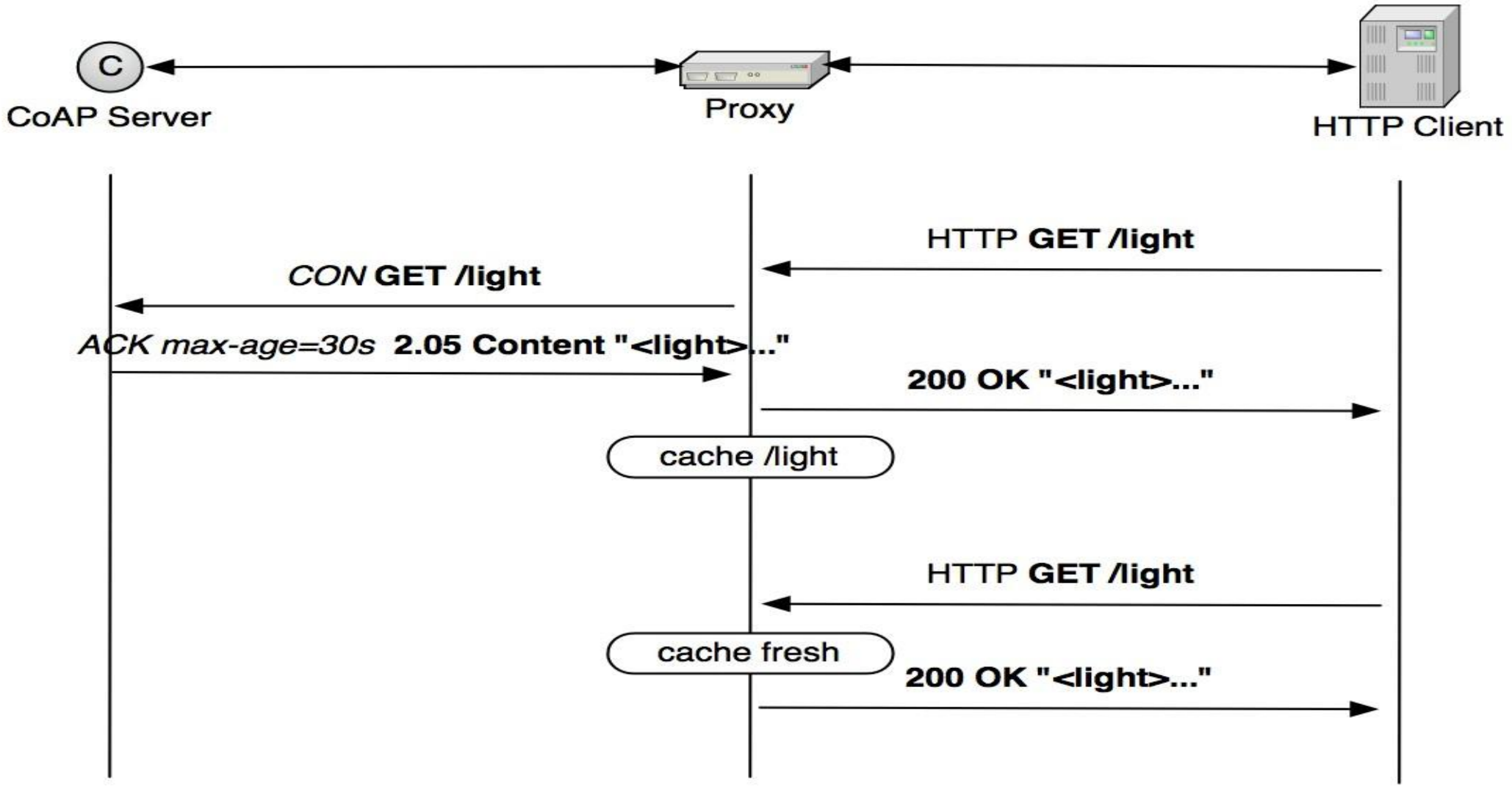
Bits and bytes...



Caching

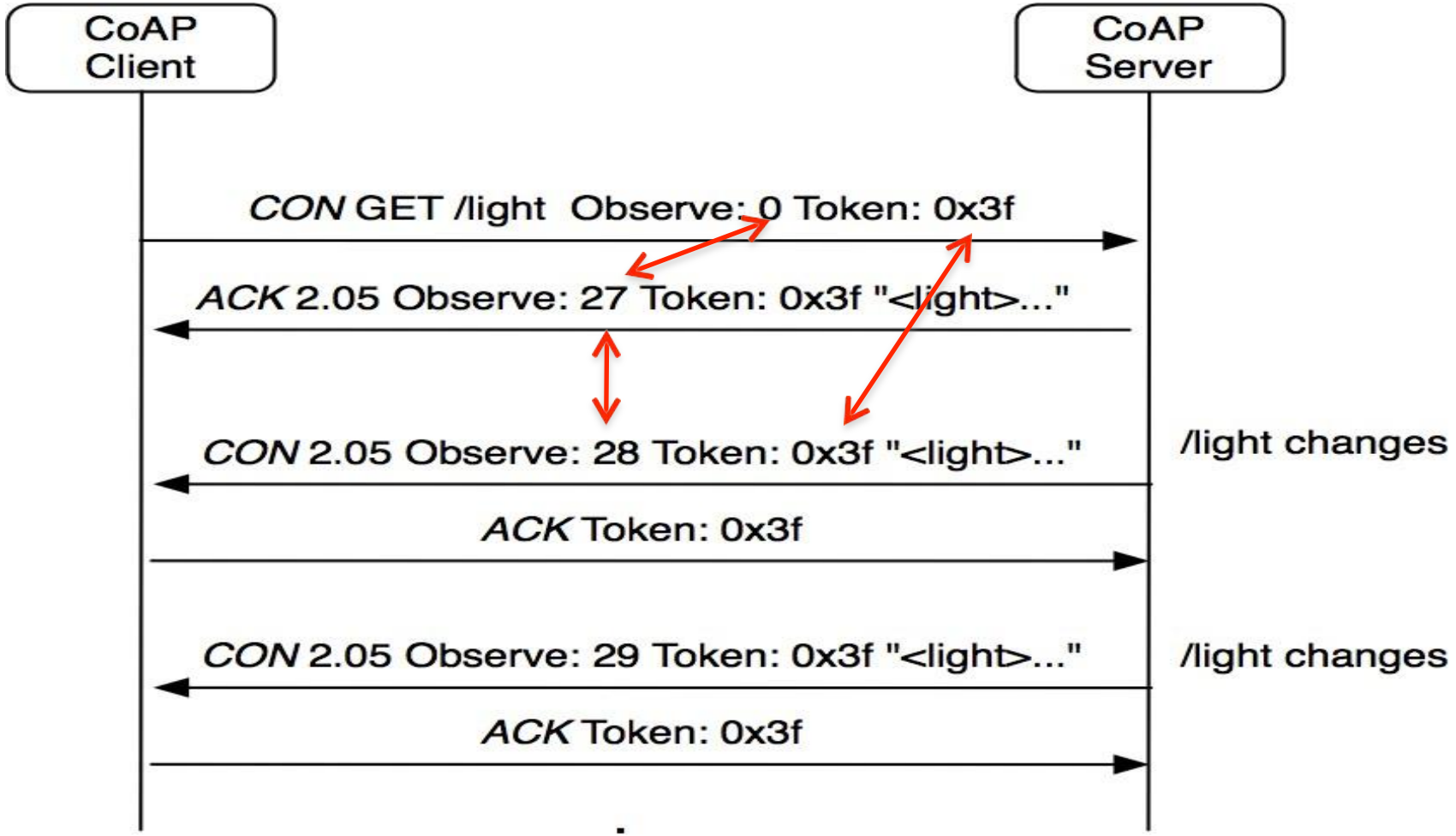
- CoAP includes a simple caching model
  - Store data on other devices
- Freshness model
  - Max-Age option indicates cache lifetime
- A proxy performs caching
  - On behalf of a constrained node
  - To reduce network load

Proxying and caching



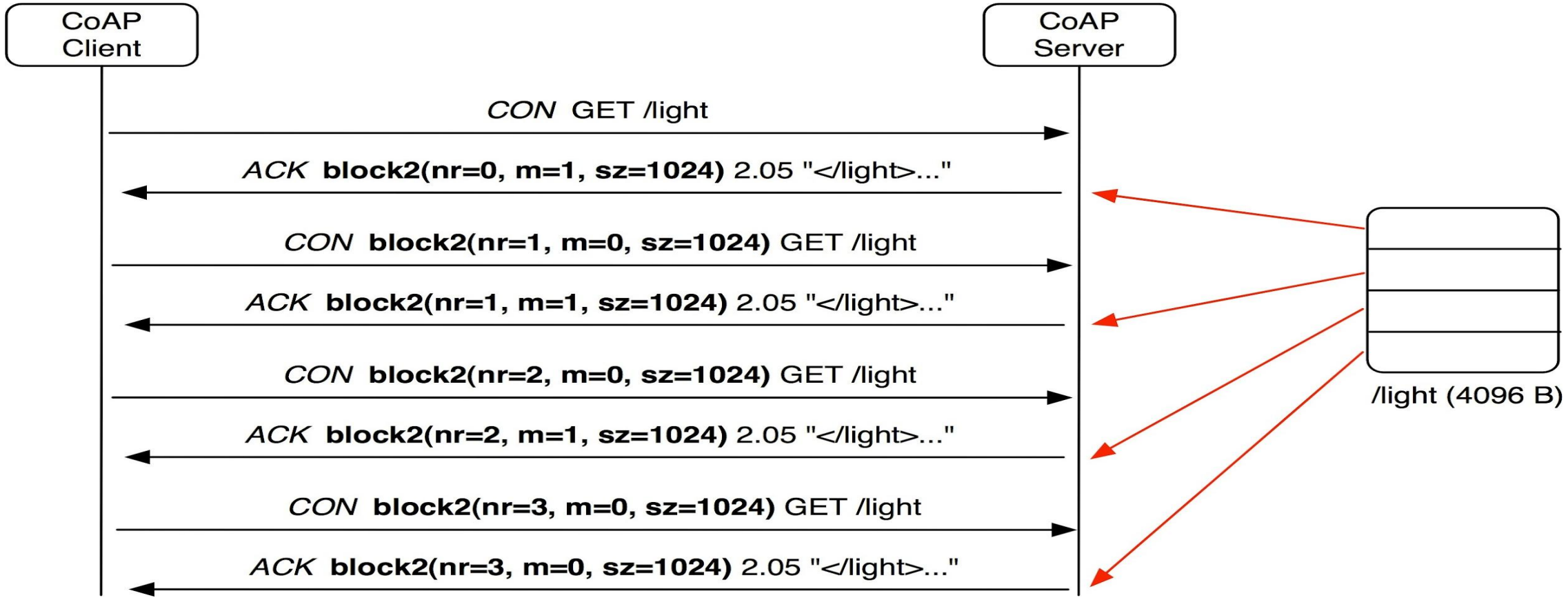
Source: https://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf

Observing Resources (Subscription)



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

Block-wise transfer



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

- Used in IoT and M2M
- Dr. Andy Stanford-Clark (IBM) and Arlen Nipper (Eurotech) in 1999
- OASIS standard in 2013
- ISO recommendation (ISO/IEC 20922)
- Public and royalty-free license
- Used in Cloud apps and web services
  - Amazon Web Services, IBM WebSphere MQ
  - Microsoft Azure IoT, Adafruit, Facebook Messenger, etc.

MQTT Features

- Small code footprint
- Ideal for resource constrained devices (processor, memory)
- Ideal if bandwidth is low or network is unreliable
- Publish/subscribe pattern
  - publishers
  - subscribers
  - broker
- Works on top of TCP/IP
  - usually over TCP

MQTT Features

- 3 Quality of Service levels:
  - at least once, at most once, exactly once
- Client libraries
  - C, C++, C#, Java, JavaScript, .NET, etc.
  - Android, Arduino
- Security:
  - authentication using username and password
  - authentication & 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 the broker and publish data to topics
- Multiple clients connect to the broker and subscribe to topics
-

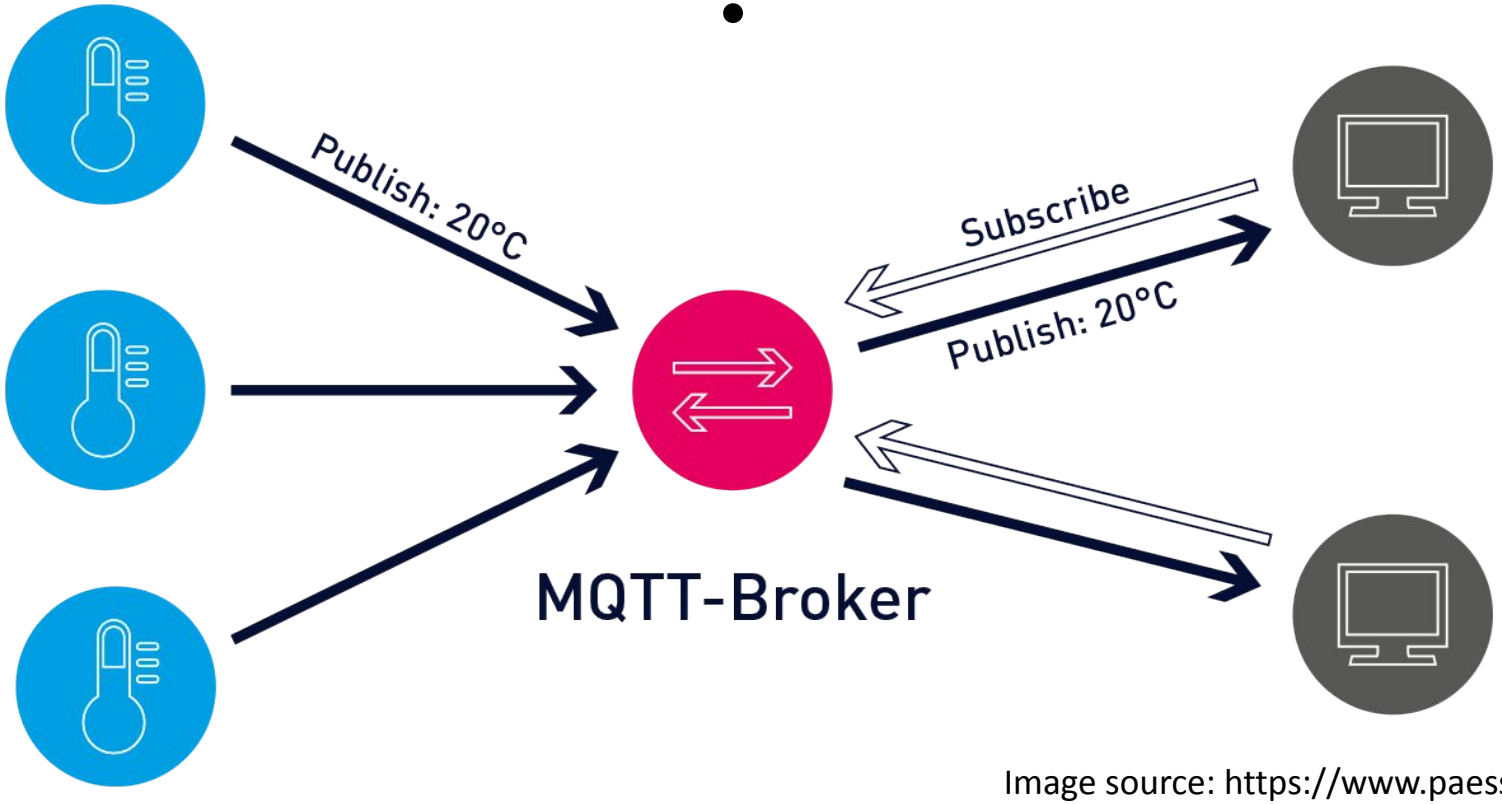


Image source: https://www.paessler.com/it-explained/mqtt

Topics

- Topics are treated as a hierarchy, using a slash (/) as a separator
  - multiple sensor devices may publish temperature readings on the topic
  - *sensors/DEVICE_NAME/temperature*
- A subscription may be to an explicit topic or it may include wildcards
  - **“+”** matches a single sublevel
  - **“#”** matches all sublevels
  - *office/room1/+/temperature*
    - all temperature data from devices in room 1
    - + = device name

Publish/Subscribe

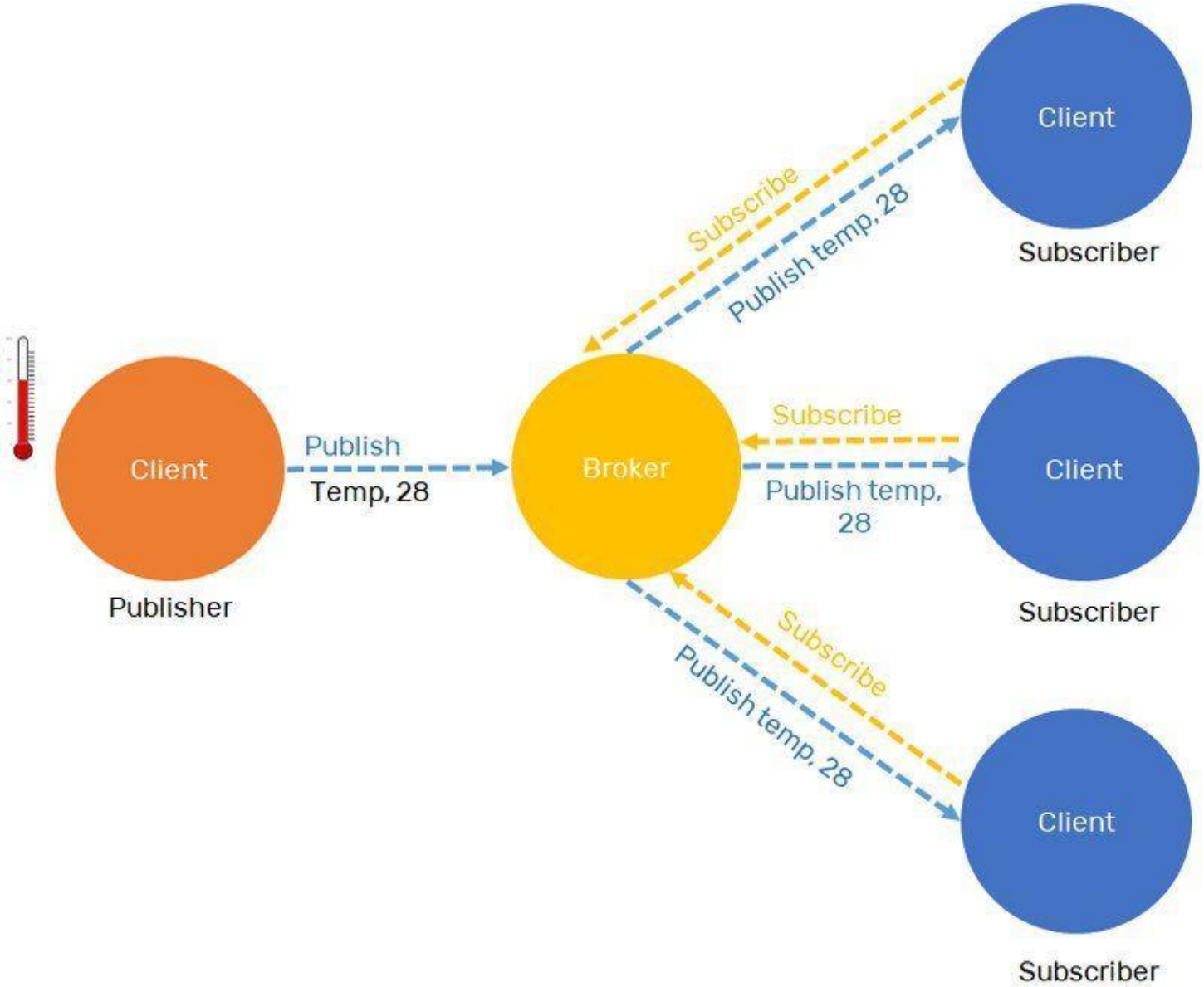


Image source: https://openlabpro.com/guide/introduction-to-mqtt-protocol/

Actions in MQTT

- 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

- Choice between minimizing data transmission and maximizing reliability
- **QoS 0 -> “At most once”**
  - Minimizing data transmission
  - Send only once, no Ack expected
  - Best effort
  - Data is not stored on the broker
  - When links are reliable, number of connections is limited
- **QoS 1 -> “At least once”**
  - Expect Ack
  - Retransmitted if Ack is not received within a period of time
  - The subscriber may receive duplicate messages
  - Compromise between QoS 0 and 2

QoS Levels

- **QoS 2 -> “Exactly once”**
  - 4 step handshake
  - Request to send, Clear-to-send, message, Ack
  - links are not reliable, number of connections is not limited
- QoS 1&2
  - Messages are saved when clients are offline
  - Retransmitted when they are online

- **Retained Messages**
  - Server keeps messages even after sending them to all subscribers
  - New subscribers get the retained messages
  - Method of caching

MQTT Options

- **Clean Sessions** and **Durable Connections**
  - **Clean** 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 message that should be published if unexpected disconnection
  - Notify subscribers when publisher is disconnected
  - Send the last collected data

MQTT vs. CoAP

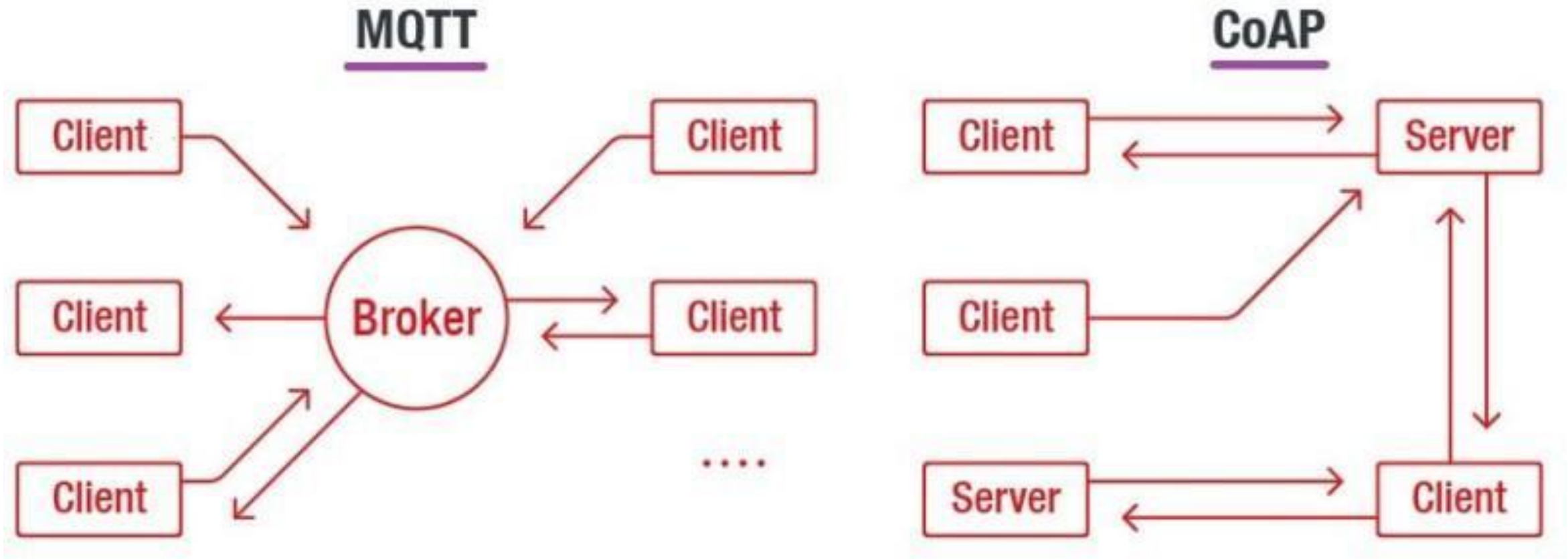


Image Source: https://iotbyhvm.ooo/coap-vs-mqtt/

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

IoT Sensor Node Crunch

Embedded Systems Laboratory

MQTT vs. CoAP

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
- https://youtu.be/4bSr5x5gKvA
- 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