CoAP-Constrained Application Protocol

Prof SRN Reddy

Abbreviations

CoRE: Constrained RESTful Environment

REST: Representational State Transfer

RESTfull Architecture : independence and scalability, as well as minimize communication and latency

DTLS: Datagram Transport Layer Security

RST- Rest Message

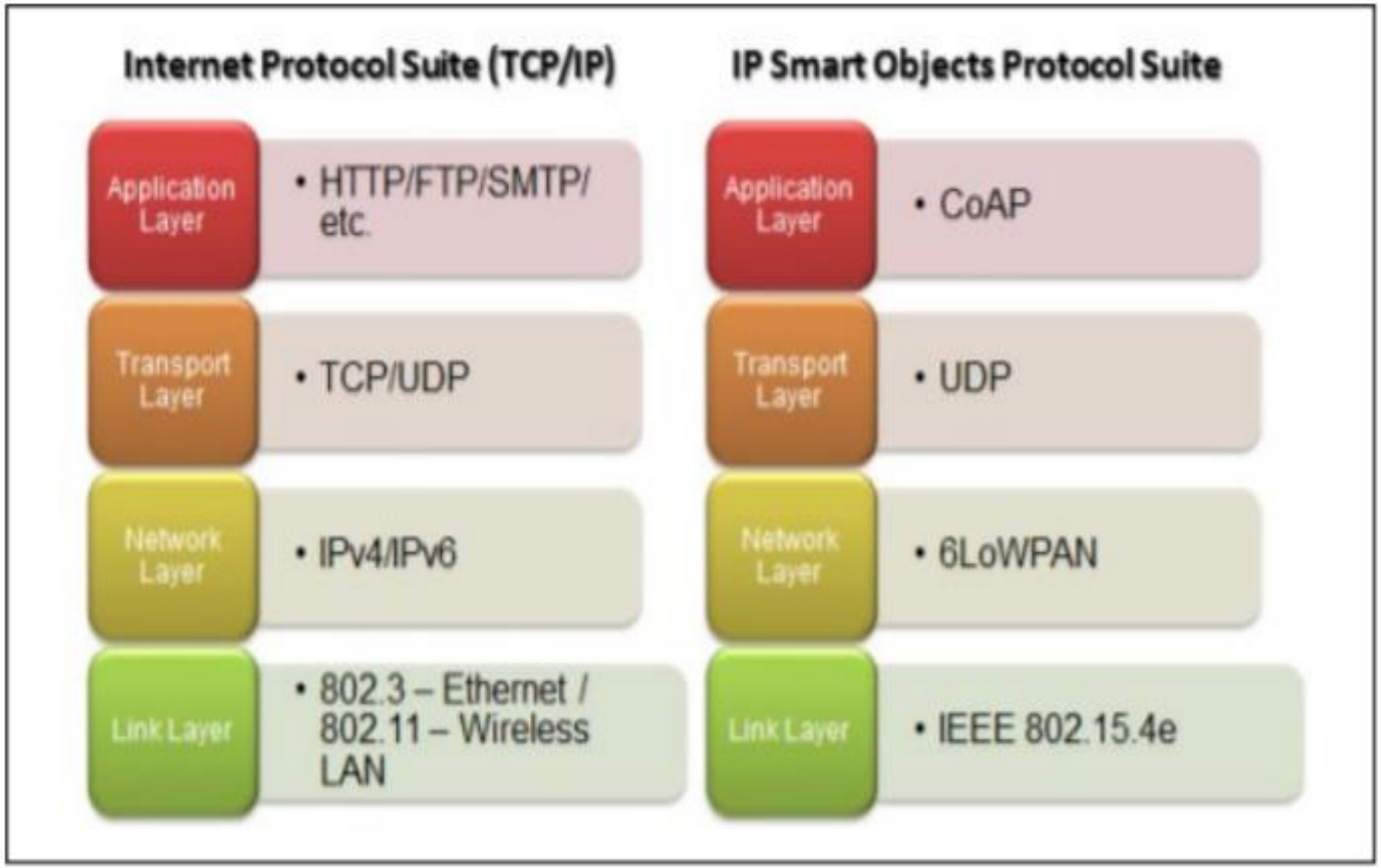
CON: Confirmable message

NON: Non-confirmable message

MIME: Multipurpose Internet Mail Extensions

JSON: JavaScript Object Notation

IETF: Internet Engineering Task Force


Internet Protocol Suite (TCP/IP)
Application Layer
• HTTP/FTP/SMTP/ etc.
Transport Layer
• TCP/UDP
Network Layer
• IPv4/IPv6
Link Layer
• 802.3 – Ethernet / 802.11 – Wireless LAN
IP Smart Objects Protocol Suite
Application Layer
• CoAP
Transport Layer
• UDP
Network Layer
• 6LoWPAN
Link Layer
• IEEE 802.15.4e

CoAP Terminology

- Endpoint - Partcipating entity.
- Sender - Orginating entity.
- Recipient - Terminating entity.
- Client - Originating endpoint of request.
- Server - Destination endpoint of response.
- Origin Server - Resource resides here.
- Intermediary - Common endpoint which acts as both client and server.
- Proxy - Forwards requests and performs caching. Forward - endpoint selected by client, Reverse - endpoint stands in for other servers , Cross - proxy to translate between protocols.

- Message - Confirmable - messages which require acknowledgemet, Non-Confirmable - messages do not require acknowledgement, Acknowledgement - acknowledges reciept of confirmable message, Reset - acknowledges reciept of confirmable and non-confirmable messages.
- Response - Piggy-backed, Separate.
- Options - Critical - endpoint should undertand for proper decoding, Elective - option could be ignored by endpoint, Safe - proxy can forward it even if it does not understand , Unsafe - proxy would not forward unless it understands.
- Resource Discovery - Getting list of associated resources from server.
- Content Format - Format of all packets.

CoAP Outline[6]

- Messaging Model - Messages are exchanged over UDP endpoints, a confirmable message adds to the reliability.
- Request/Response Model - Requests and Responses are carried in CoAP messages. Method Codes for Requests and Response Codes for Responses. GET, PUT, POST and DELETE methods are used(similar to HTTP). Client parses the URI to get host, port, path and query components, thus URI support is simplied.
- Intermediaries and Caching - Responses are cached for faster reply to requests, proxies may be used which help in reducing network traffic.
- Resource Discovery - Machine to machine interactions require resource discovery, which is done using CoRE link format.

CoAP Features

- TCP complexities are reduced by using UDP.
- Request Methods: GET, POST, PUT, DELETE.
- Response Methods: 2.xx (success), 4.xx (client error), 5.xx (servererror).
- Message types: Confirmable, Non Confirmable, Acknowledgement and Reset.
- Unicast and multicast requests.
- Resource discovery capability.
- Block transfers for large files.

CoAP Features

Google

- Embedded Web Transfer Protocol (coap://)
- Small and simple 4 byte Header
- Open IETF Standard
- Asynchronous Transaction Model
- Strong Datagram TLS Security over UDP
- UDP, TCP Support
- Easy to proxy to/from HTTP
- GET, PUT, POST, DELETE Methods.

CoAP

- Designed for constrained nodes and constrained networks in the IoT
- Useful for M2M commination in smart energy and building automation
- It can carry different types of payloads, and can identify which payload type is being used.
- CoAP integrates with XML, JSON or any data format of your choice.

Choose your data model

- It can carry different types of payloads, and can identify which payload type is being used.
- CoAP integrates with XML, JSON, CBOR, or any data format of your choice.

H/W Requirement

- IoT nodes be inexpensive.
- CoAP has been designed to work on microcontrollers with as low as
  RAM: 10 KiB
  Flash: 100 KiB of code space

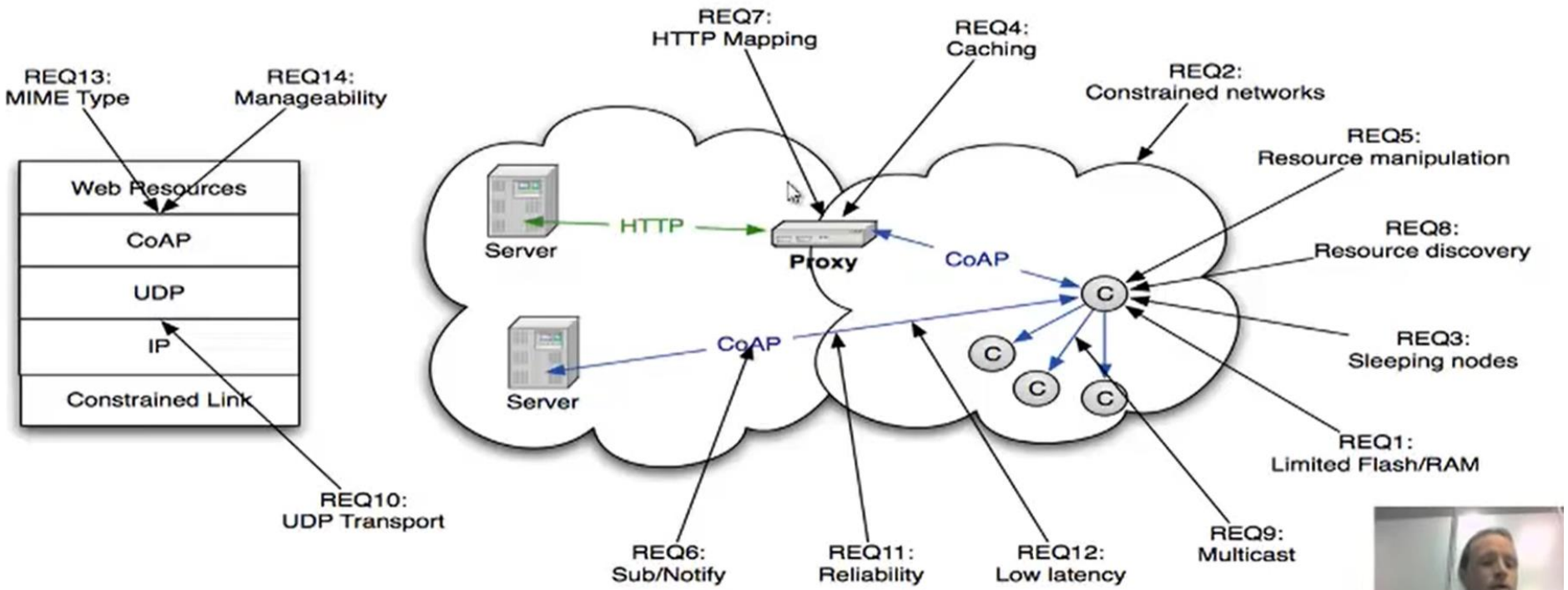
Protocol Stack

- CoAP runs on UDP on IP instead of TCP to reuse the code size
- A 4-byte fixed header and a compact encoding of options enables small messages that cause no or little fragmentation on the link layer

Security

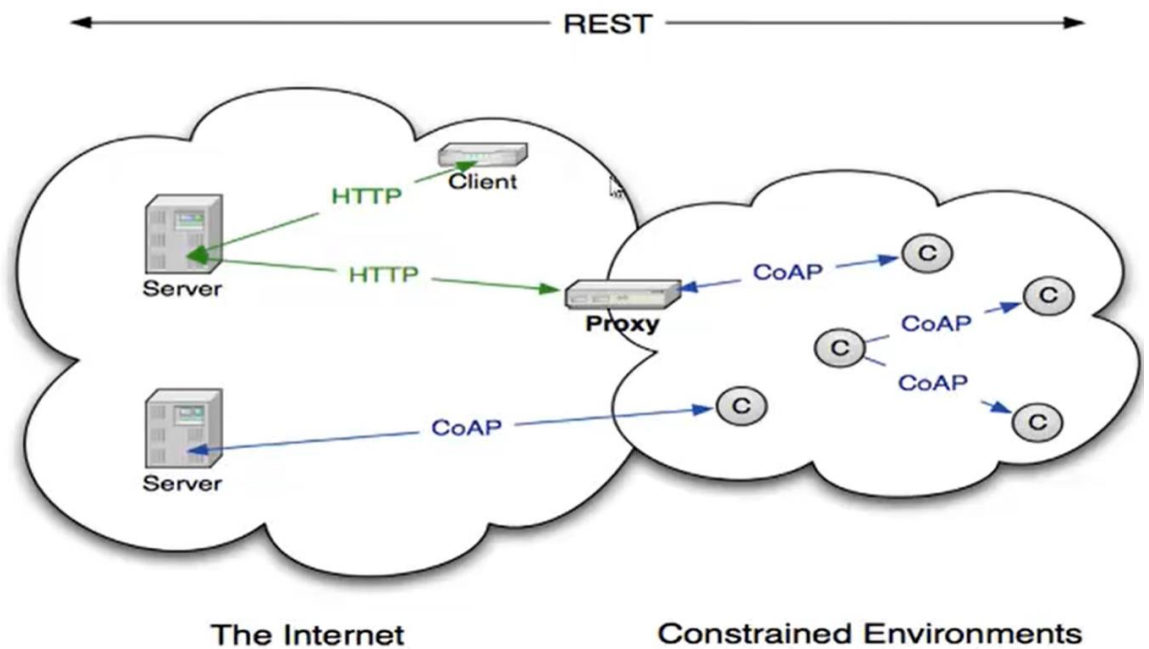
- CoAP provides strong security.
- CoAP's default choice of DTLS parameters is equivalent to 3072-bit RSA keys
- It runs fine on the smallest nodes.

CoAP Design Requirements



arm

The CoAP Architecture



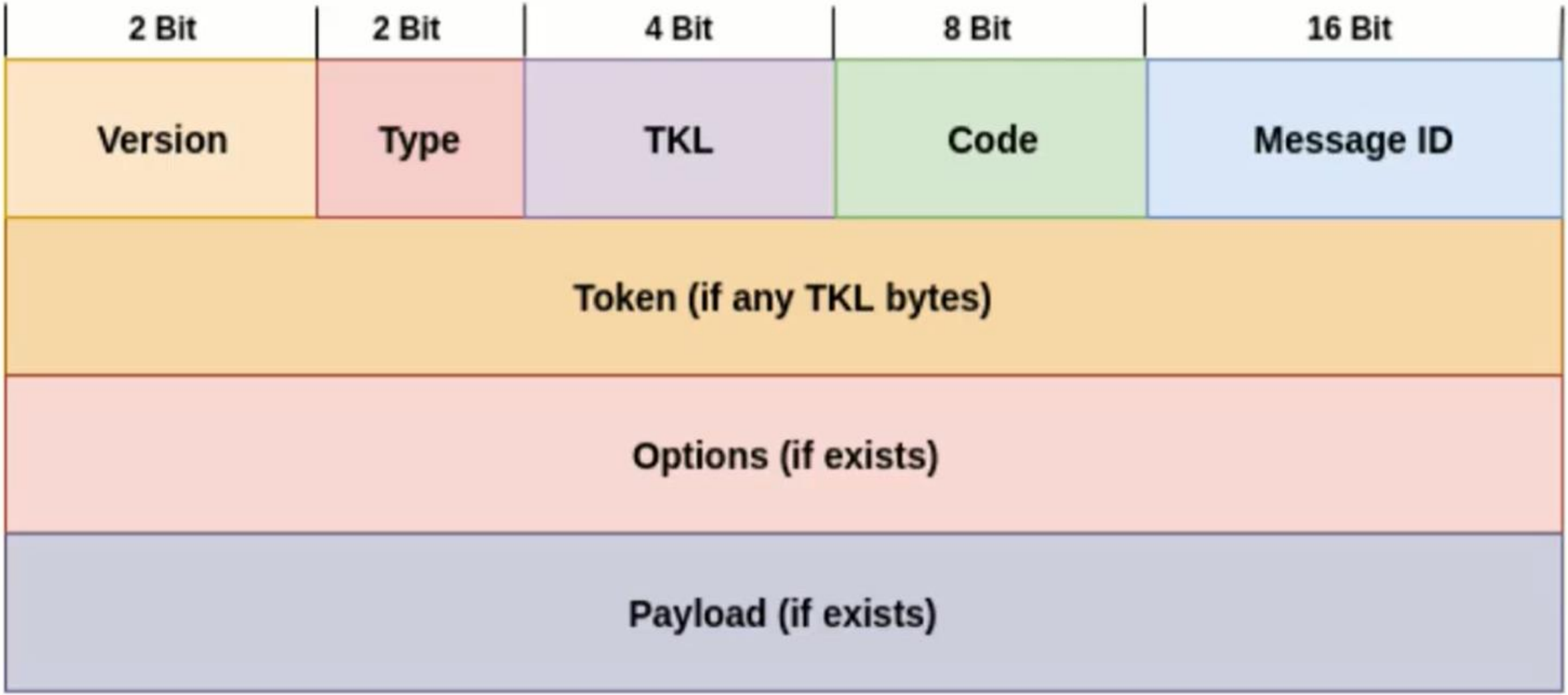
What CoAP is (and is not)

- Sure, CoAP is
  - A very efficient RESTful protocol
  - Ideal for constrained devices and networks
  - Specialized for M2M applications
  - Easy to proxy to/from HTTP

- But hey, CoAP is not
  - A general replacement for HTTP
  - HTTP compression
  - Restricted to isolated "automation" networks

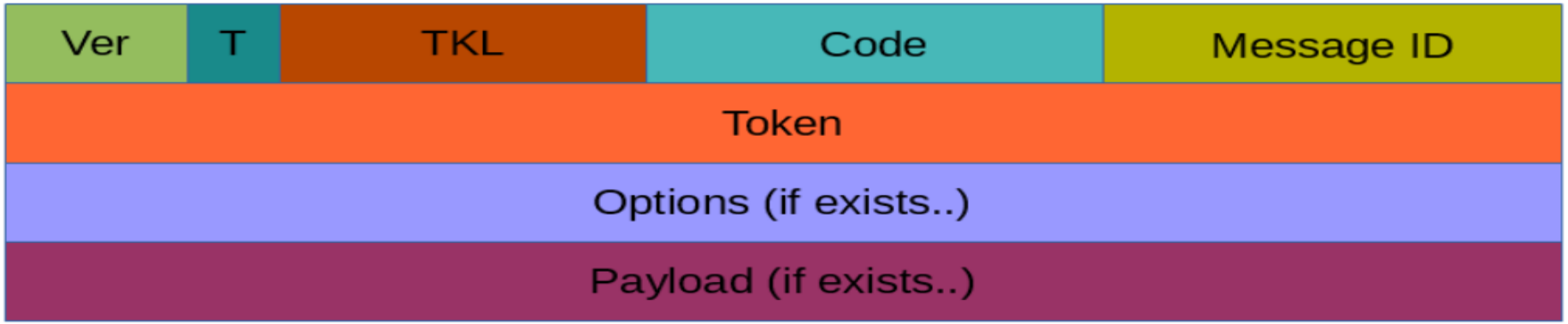
Message Communication Model

- CoAP requests and responses are transferred asychnronously wrapped in messages.
- Due to UDP, messages could be out of order, duplicate or get lost.
- Introduces a reliable lightweight protocol like TCP.
- Stop-and-wait protocol Binary exponential back-off for Confirmable messages. Duplicate detection for both Confirmable and Non-confirmable messages.
- Message Transmission is asynchronous between the endpoints.
- A CoAP endpoint is a source or destination of a message.
- Without security endpoints are identified by IP and Port number.
- With security its the security mode: NoSec, PreSharedKey, RawPublicKey and Certificate.



CoAP Message Format

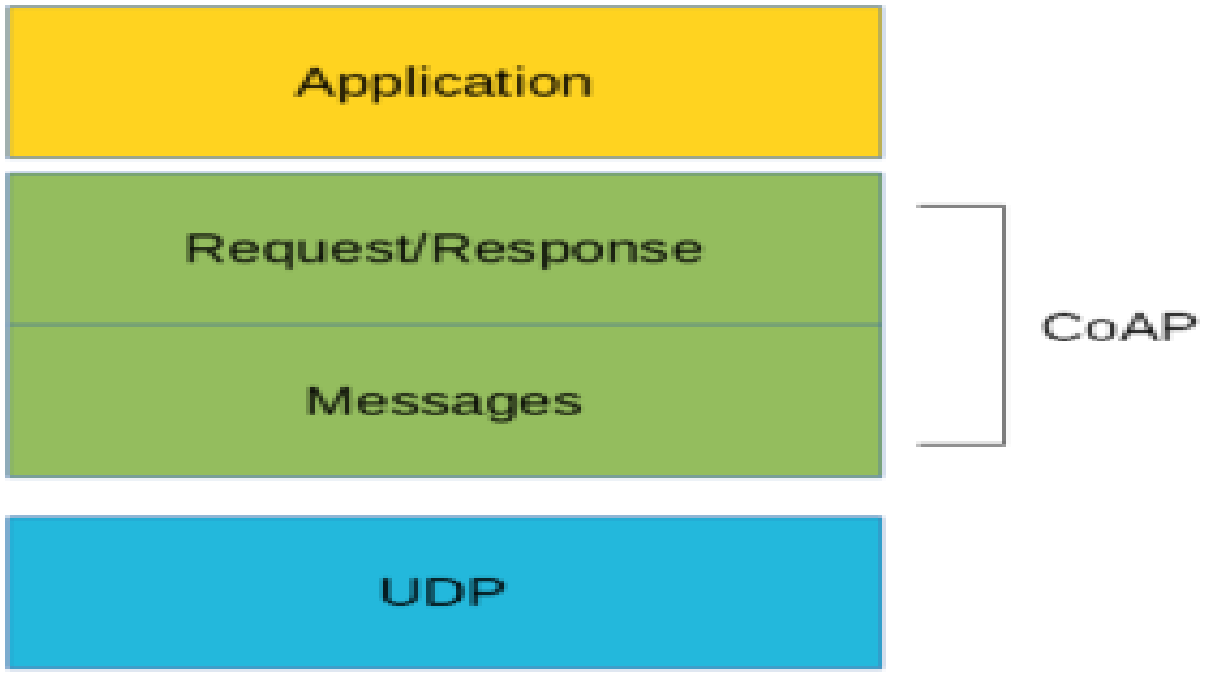
Header



Where:

- **Ver**: It is a 2 bit unsigned integer indicating the version
- **T**: it is a 2 bit unsigned integer indicating the message type: 0 confirmable, 1 non-confirmable
- **TKL**: Token Length is the token 4 bit length
- **Code**: It is the code response (8 bit length)
- **Message ID**: It is the message ID expressed with 16 bit

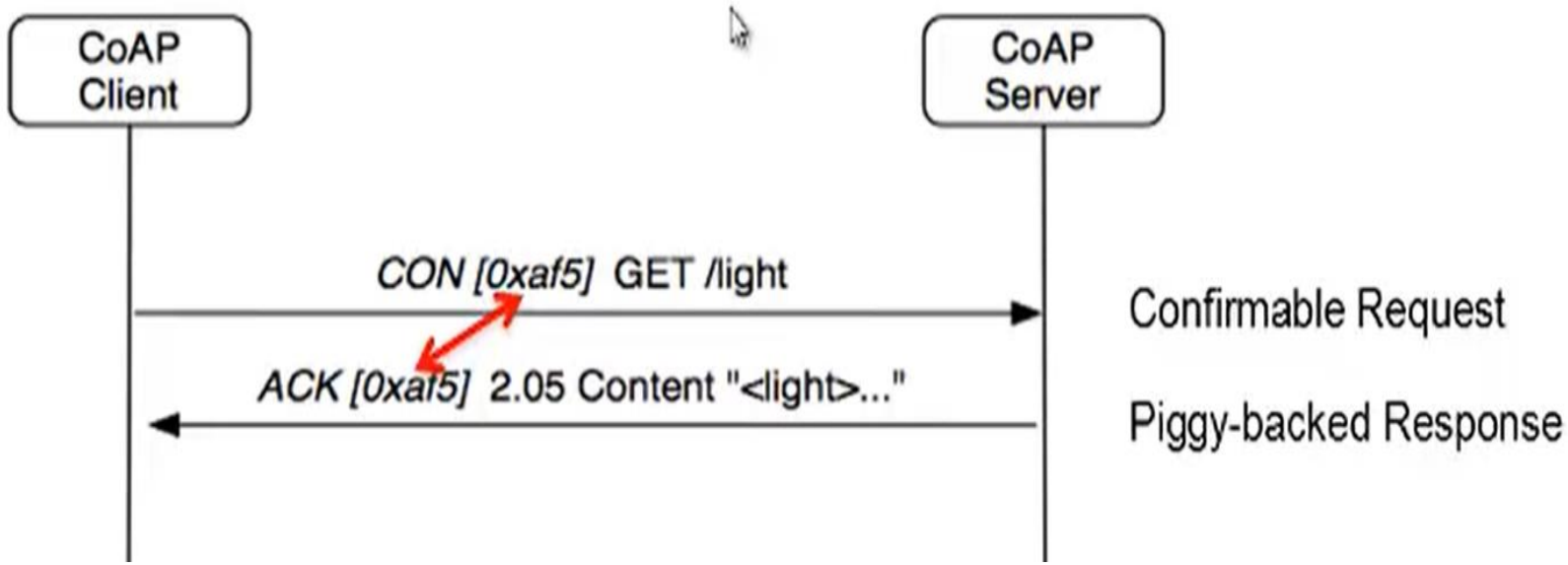
CoAP Abstraction Model



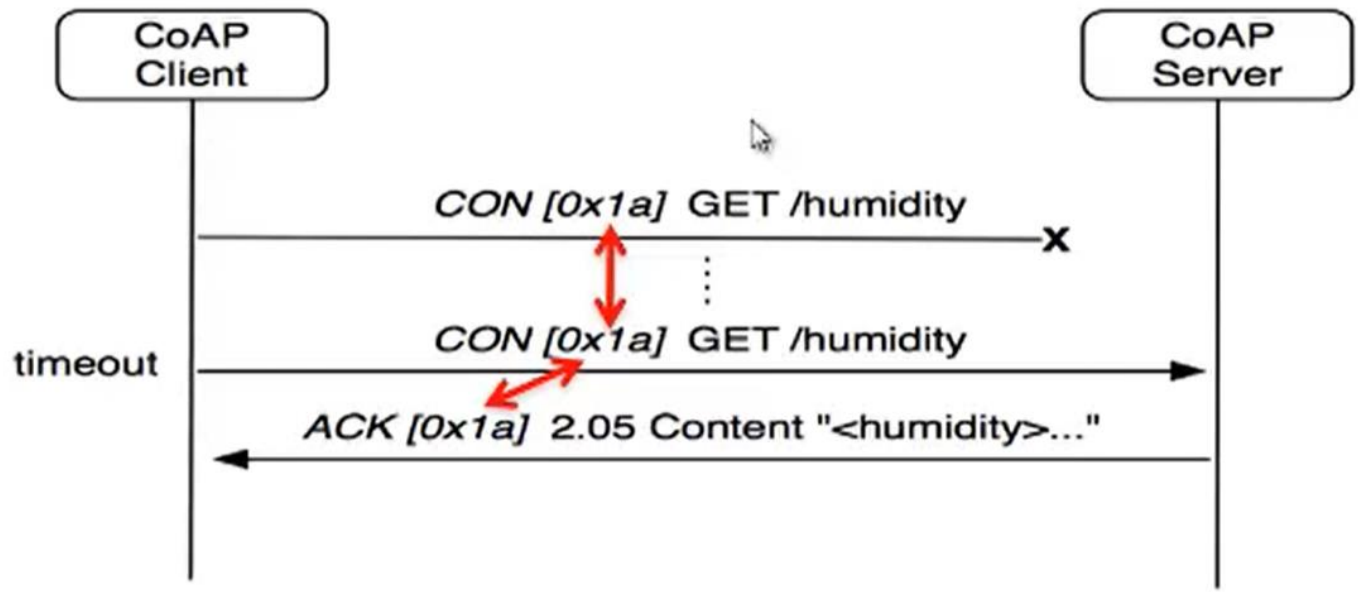
- The Messages layer deals with UDP and with asynchronous messages.
- The Request/Response layer manages request/response interaction based on request/response messages.

- CoAP supports four different message types:
    - **Confirmable [CON]**
    - **Non-confirmable[NON]**
    - **Acknowledgment[ACK**
    - **Reset[RST]**

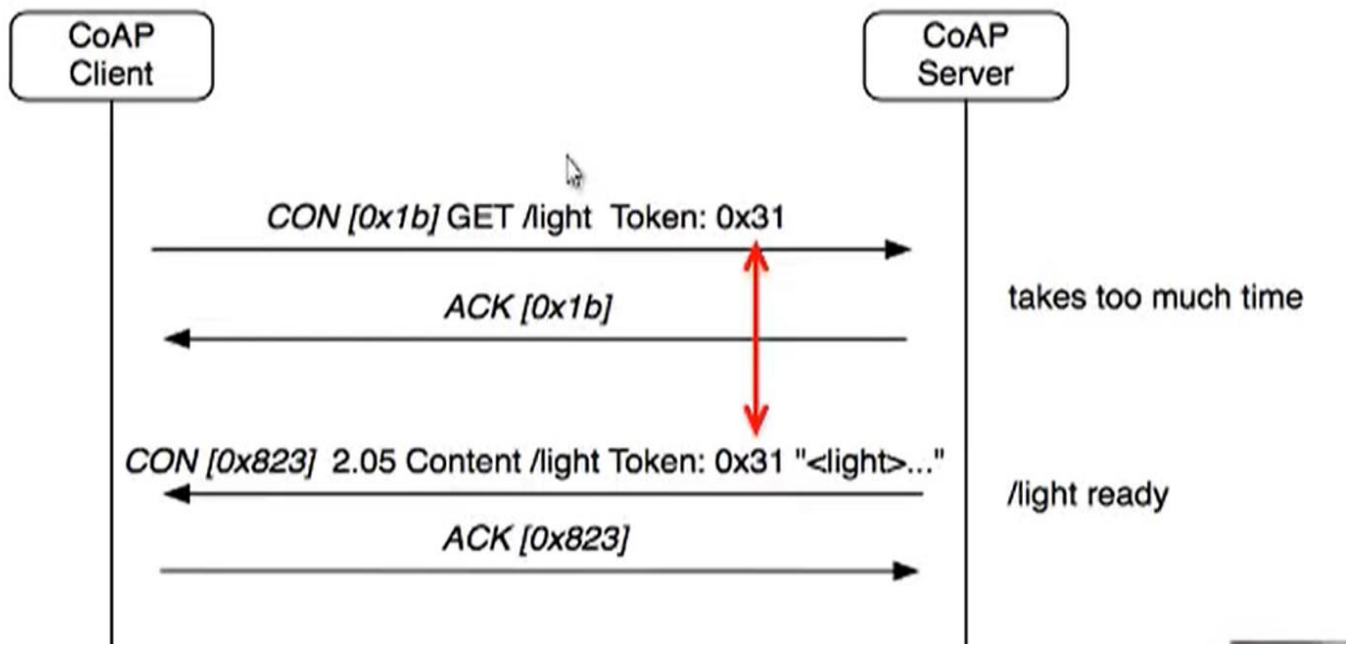
Request Example



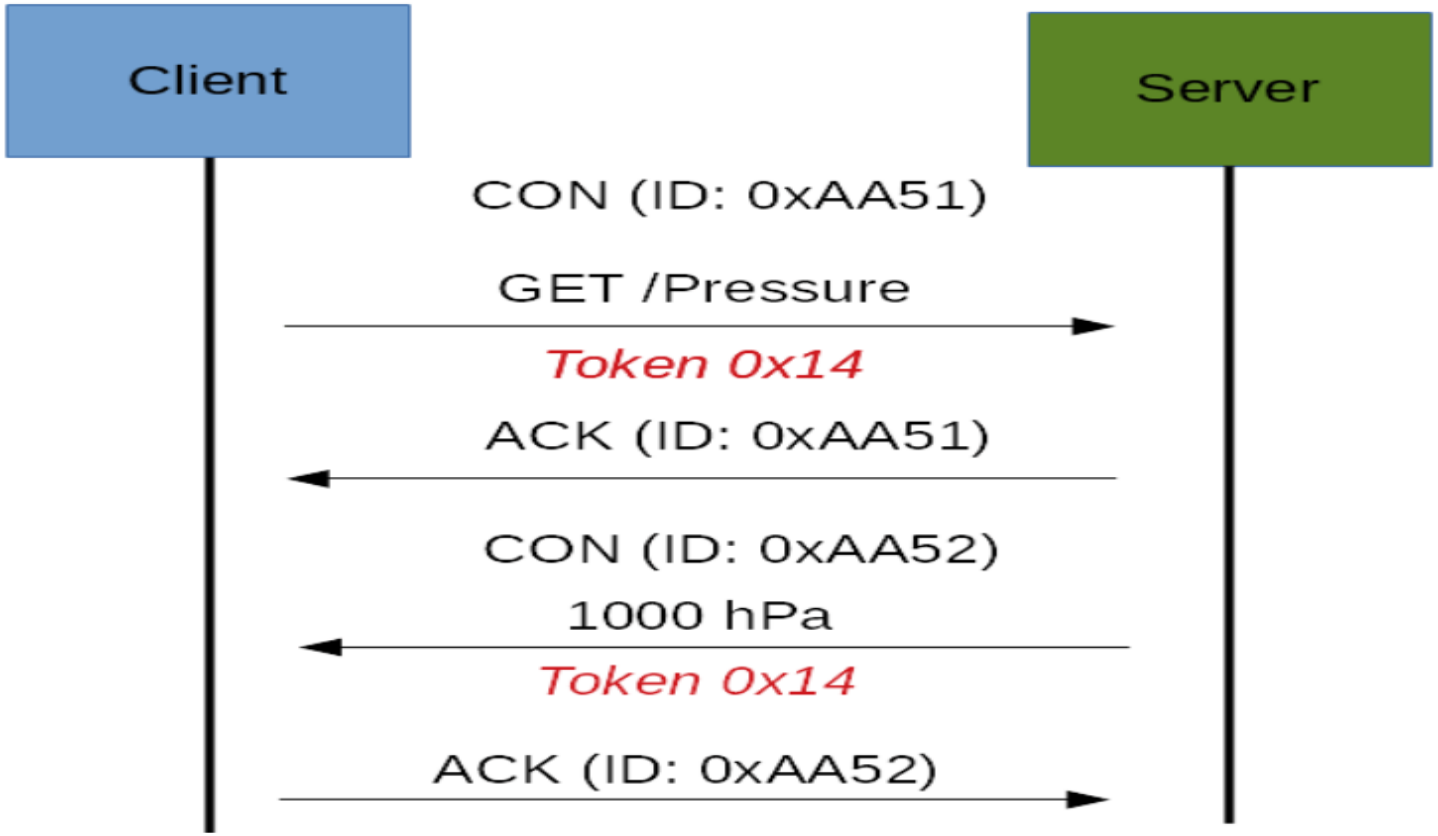
Dealing with Packet Loss



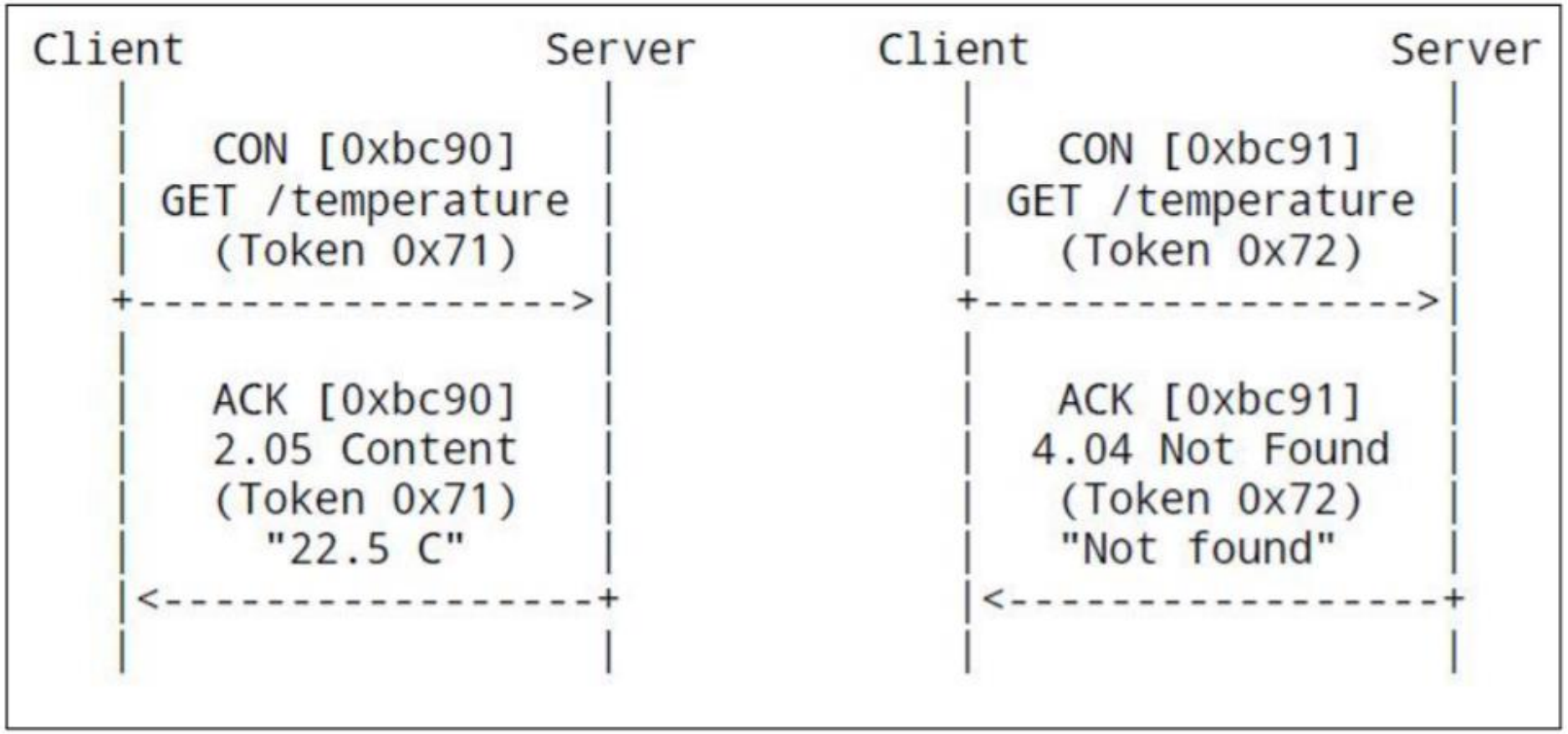
Separate Response



CoAp Request/Response Model: Server is Busy



Request Response Model



CoAP vs HTTP

- CoAP (Constrained Application Protocol) is an IoT protocol
- HTTP is complex and has large overhead compared to CoAP
- HTTP uses TCP, CoAP uses UDP.
- HTTP does not support multicast, while CoAP does (bcoz of UDP).
- HTTP work on client/server model, while CoAP can use both Client/server and Publish/Subscribe Model.
- HTTP needs Synchronous comm, while CoAP is Asynchronous.
- HTTP not efficient for sending a single piece of information
- CoAP will be lighter and faster than HTTP for IoT Applications.

References

1. https://coap.technology/impls.html
2. https://www.youtube.com/watch?v=4bSr5x5gKvA [ARM Video]
3. https://www.youtube.com/watch?v=WtRpFLx34BY
4. https://www.youtube.com/watch?v=Bd3BRv4hO-4
5. https://arxiv.org/pdf/1804.01747.pdf
6. Z. Shelby, Sensinode, K. Hartke, C. Bormann, and U. B. TZI, “Internet-draft - constrained application protocol (coap),” 2013
7. R. T. Fielding and R. N. Taylor, “Principled design of the modern web architecture,” ACM Trans. Internet Technol., vol. 2, no. 2, pp. 115–150, May 2002.
8. https://www.iiitd.edu.in/~amarjeet/EmSys2013/CoAPv5.1.pdf
9. https://dzone.com/articles/coap-protocol-step-by-step-guide
10. https://www.rfc-editor.org/rfc/pdfrfc/rfc7252.txt.pdf [Detail Specifications]