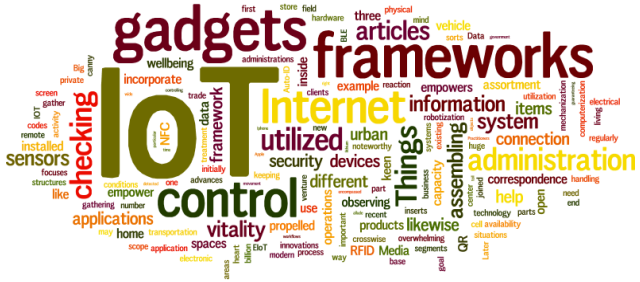
CS578: Internet of Things

CoAP: Constrained Application Protocol



Dr. Manas Khatua

Assistant Professor

Dept. of CSE, IIT Guwahati

E-mail: manaskhatua@iitg.ac.in

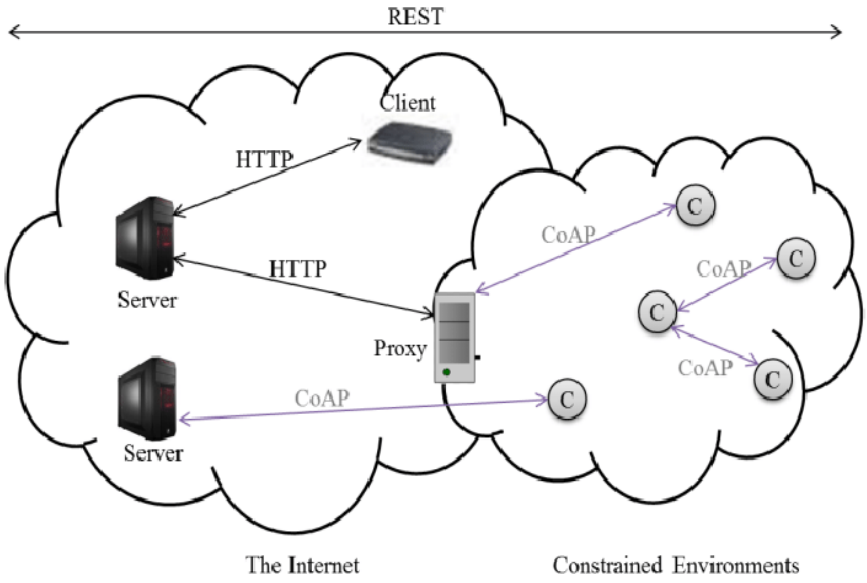
What is CoAP

- CoAP - Constrained Application Protocol.

- **Specialized** web transfer protocol.
- Devised for **constrained** and **low power** networks.

- CoAP is an **application layer protocol** (similar as HTTP).
- Inspired from HTTP, it follows the **request-response** pattern

- CoAP has a **transparent mapping** to HTTP.
- CoAP protocol spec is specified in **RFC 7252**.



Characteristics of CoAP

- It is very efficient **RESTful** protocol.
- It is Embedded web transfer protocol

- Low parsing complexity.
- Proxied to/from HTTP.
  - GET, POST, PUT and DELETE methods are used.
  - Uses subset HTTP response codes.
  - URI is supported.

- Low header overhead
- It uses small and simple **4 byte header.**

- Supports binding to UDP, and TCP.

- DTLS based certificate security is used.

What is REST?

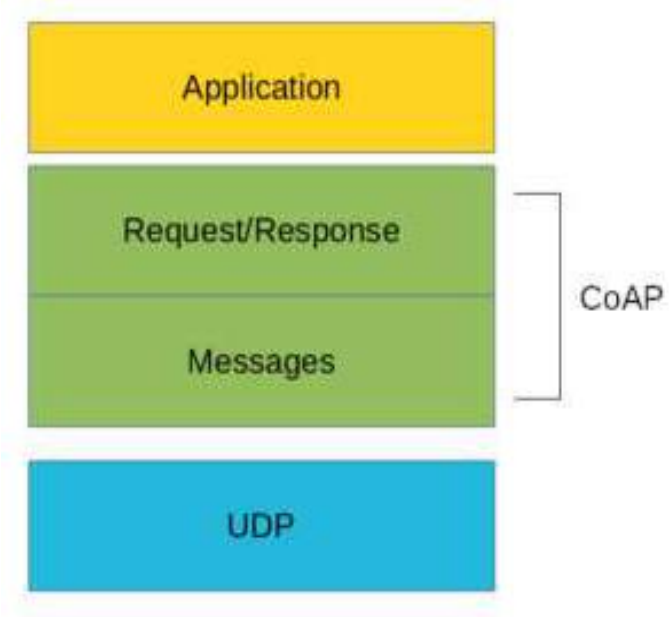
- Representational State Transfer (REST) is a software architecture that imposes conditions on how an API should work.

- API developers can design APIs using several different architectures. APIs that follow the REST architectural style are called REST APIs or RESTful web APIs.

- Some of the constraints of the REST architectural style:
  - Uniform interface
  - Statelessness
  - Cacheability
  - Code on demand
  - Layered system
  - Etc.

CoAP Structure

- There are **two different layers** that make CoAP Protocol:
  - Message
  - Request/Response.



- The **Messages layer**
  - ✓ deals with UDP
  - ✓ deals with asynchronous messages.
  - ✓ is meant for Re-transmitting lost packets.

- **Message layer** supports four types of messages:
  - Confirmable (CON)
  - Non-confirmable (NON)
  - Acknowledgment (ACK)
  - Reset (RST)

  For Request

  For Response

- The **Request/Response layer**
  - ✓ manages request/response interaction based on request/response messages.

- Request/Response layer uses different methods
  - Request Methods: **GET**, **POST**, **PUT**, and **DELETE**.
  - Response Methods: 2.xx (**success**), 4.xx (**client error**), 5.xx (**server error**).

CoAP Status Codes in Response

CoAP Status Codes

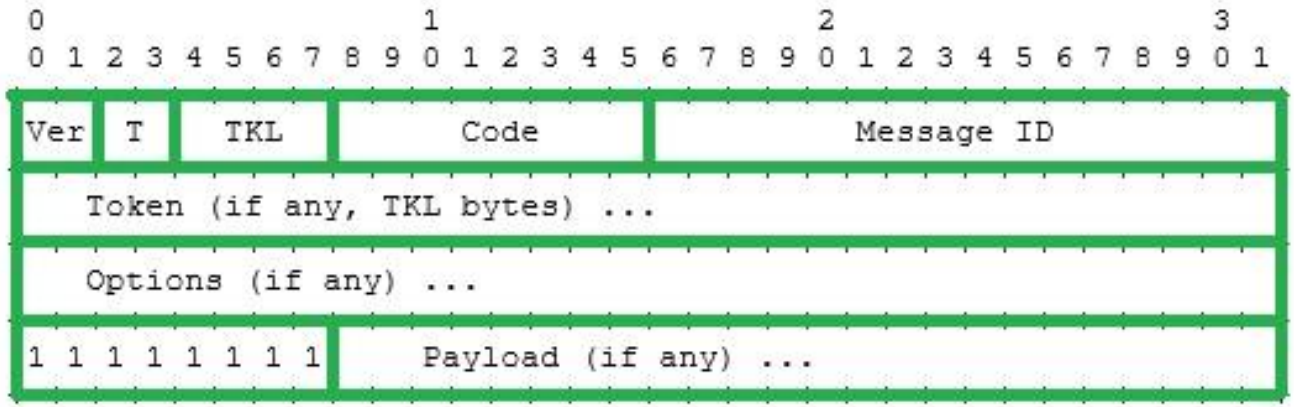
CoAP Status Code	Description
2.01	Created
2.02	Deleted
2.03	Valid
2.04	Changed
2.05	Content
2.31	Continue
4.00	Bad Request
4.01	Unauthorized
4.02	Bad Option
4.03	Forbidden
4.04	Not Found
4.05	Method Not Allowed
4.06	Not Acceptable
4.08	Request Entity Incomplete
4.12	Precondition Failed
4.13	Request Entity Too Large
4.15	Unsupported Content-Format
5.00	Internal Server Error
5.01	Not Implemented
5.02	Bad Gateway
5.03	Service Unavailable
5.04	Gateway Timeout
5.05	Proxying Not Supported

HTTP Status Code	Description
1xx	Informational
2xx	Successful 200 – OK 201 – Created 202 – Accepted 204 – No Content
3xx	Redirection 301 - Moved Permanently 305 - Use Proxy 307 - Temporary Redirect
4xx	Client Error 400 – Bad Request 401 – Unauthorized 403 – Forbidden 404 - Not Found 405 – Method Not Found 408 – Request Timeout
5xx	500 – Internal Server Error 501 – Not Implemented 503 – Service Unavailable 504 - Gateway Timeout

Only mostly used HTTP Status Codes are listed here

Source: https://www.slideshare.net/aniruddha.chakrabarti/coap-web-protocol-for-iot

CoAP Message Format



- **Ver (2 bit):** indicating the CoAP **version.**
- **T (2 bit):** indicating the **message type**
  - Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK), Reset (RST)

- **TKL (4 bit)**: Specifies the size (0-8 bytes) of the Token field
- **Token (0-8 byte):** correlates requests and responses

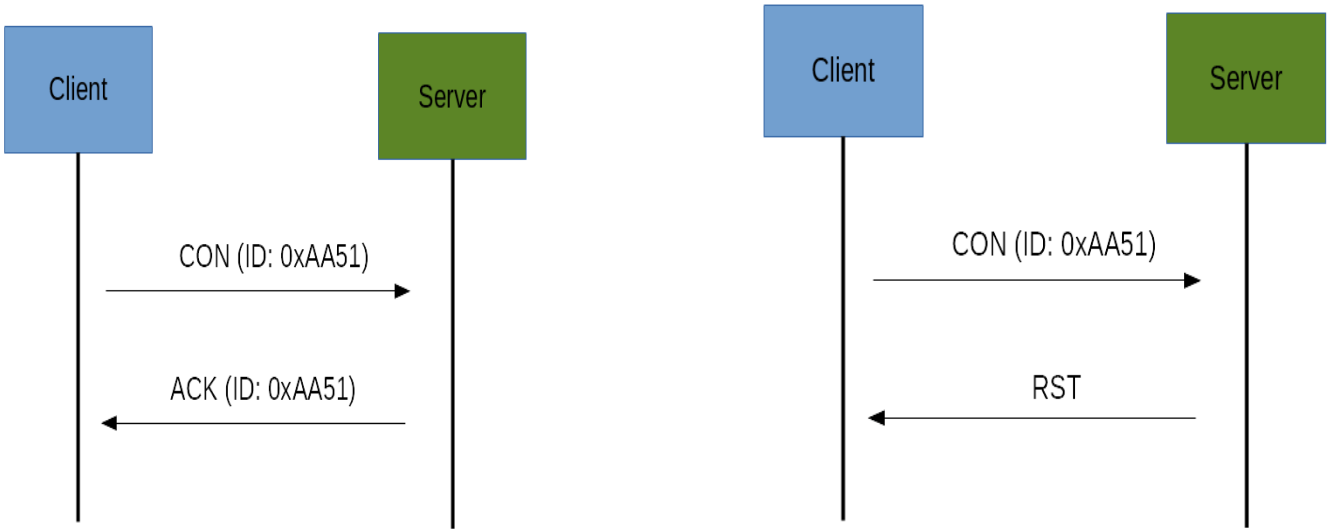
- **Code (8 bit)**: indicates
  - request method for a request message, e.g: GET is the request method
  - response code for a response message. e.g. 2.05 is the response code

- **Message ID (16 bit)**:
  - Detects message duplication
  - Used to match ACK and RST message types to CON and NON message types.

CoAP Messaging Model

- CoAP deals with **UDP** for exchanging messages between endpoints.
- Each CoAP message has a **unique ID**
- **Unique ID** is useful to detect message duplicates.

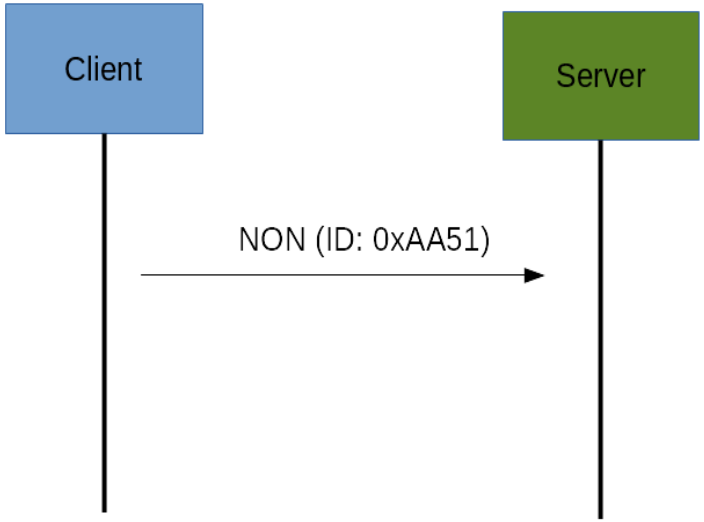
- **Reliable messaging** is obtained using a **Confirmable message (CON).**
  - A **CON** message is sent again and again until the other party sends an **acknowledge (ACK)** message **or** negative acknowledgement through reset message (RST).
    - ✓ The **ACK** message contains the same ID of the **CON** message
    - ✓ If the server has troubles managing the incoming request, send back a **Reset message (RST)** instead of the **ACK**.
    - ✓ Retransmissions are made until all attempts are exhausted



Cont…

- **Non-confirmable (NON)** messages **don't** require an **ACK** by the server.

- They are unreliable messages or in other words messages that **do not** contain critical information that must be delivered to the server.
  - **Example :** Messages that contain sensed values from sensors.
  - Even if these messages are unreliable, they have a **unique ID**.

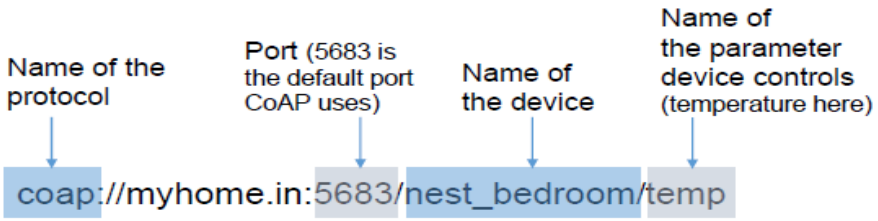


CoAP Request/Response Model

The CoAP **Request/Response** is the second layer in the CoAP Abstraction layer.

- The request is sent using a Confirmable (CON) or Non-confirmable (NON) message.
- There are several scenarios (e.g. Piggy-Backed, Separate Response) depending on if the server can answer immediately to the client request or if not available.

CoAP URI:

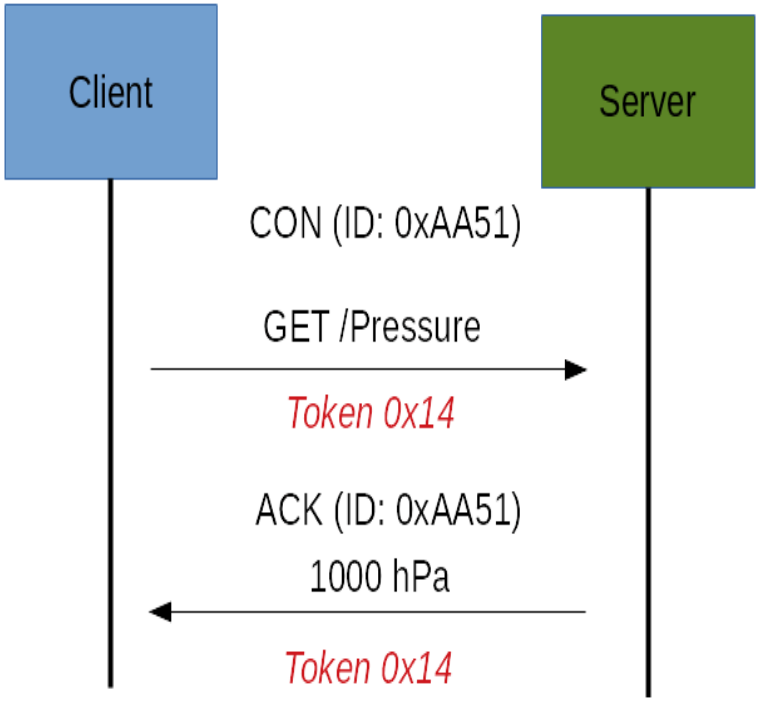


Cont…

If the server can answer immediately to the client request (**Piggy-Backed response**)

- If the **request** is carried using a Confirmable message (CON).
  - the server sends back an ACK message to client containing the **response or the error code**.

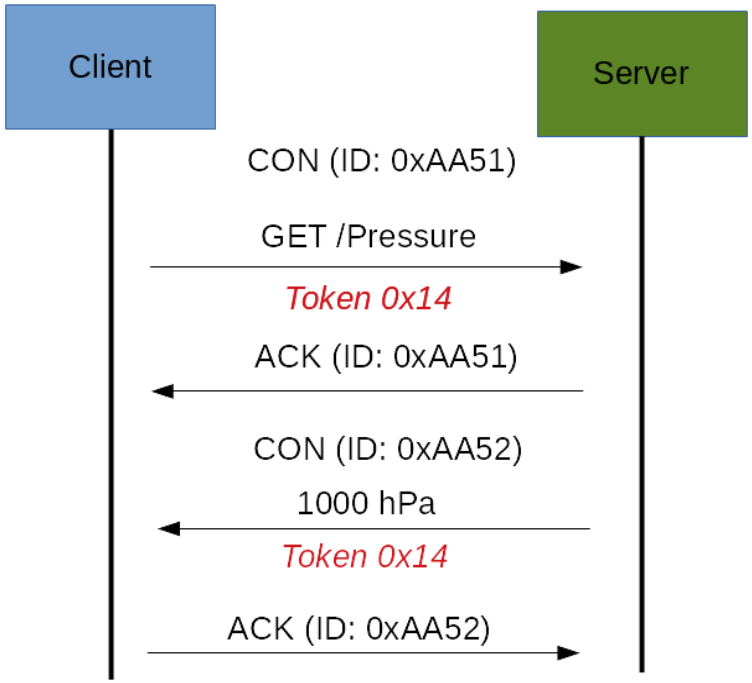
- **The Token** is different from the Message-ID and it is used to match the **request** and the **response**.



Cont…

If the server **can't** answer immediately to the request coming from the client **(Separate Response)**.

- It sends an ACK message with an **empty response**.
- When response is available, then the server sends a **new CON message** to the client containing the response.
- At this point, the client sends back an **ACK message**.



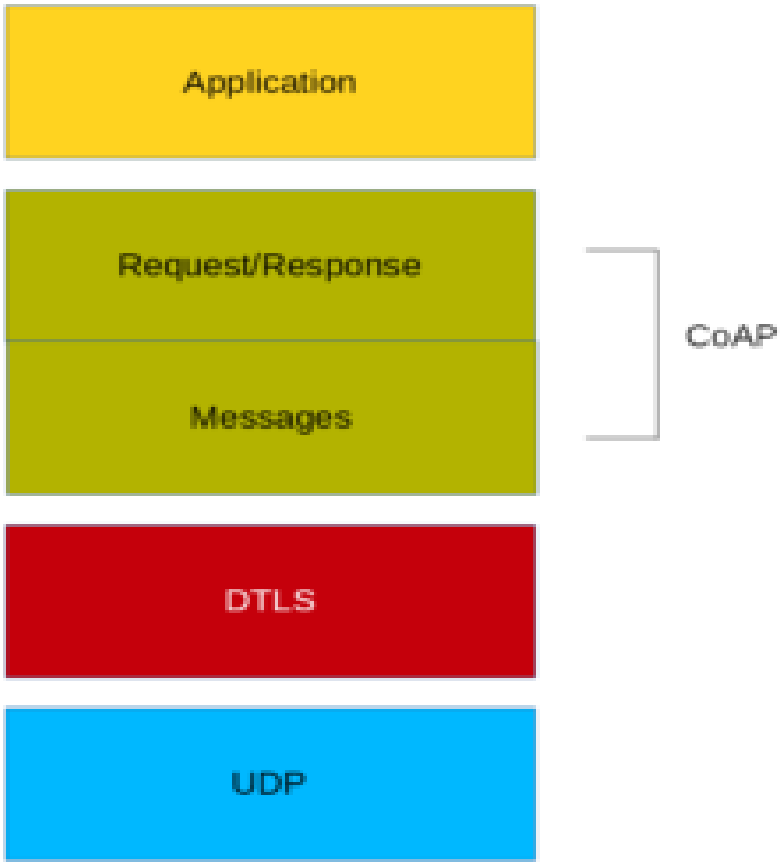
CoAP Security Aspects

- CoAP uses **UDP** to transport information.
- CoAP relies on UDP security aspects to protect the information.

- As HTTP uses TLS over TCP, **CoAP uses Datagram TLS over UDP.** DTLS supports RSA, AES.

- In some constrained devices, some of DTLS cipher suits may not be available.

- Some cipher suites introduces more **complexity** and constrained devices may not have resources enough to manage it.



CoAP Vs. MQTT

- MQTT uses a publisher -subscriber.
- MQTT uses a central broker to dispatch messages coming from the publisher to the clients.
- MQTT is an event-oriented protocol.
- MQTT uses Asynchronous messaging.

- CoAP uses a request-response paradigm
- CoAP is essentially a one-to-one protocol very similar to the HTTP protocol.
- While CoAP is more suitable for state transfer.
- CoAP uses both Asynchronous & Synchronous messaging

CoAP vs. HTTP

CoAP – How it's different from HTTP

- CoAP runs over UDP and not TCP (HTTP typically uses TCP, though it can use UDP also)
- CoAP replaces the text headers used in HTTPU (HTTP Unicast) with more compact binary headers
- It reduces the number of options available in the header
- CoAP also reduces the set of methods that can be used; it allows
  - GET
  - POST
  - PUT, and
  - DELETE
- Method calls can be made using confirmable & nonconfirmable message services
  - When a confirmable message is received, receiver always returns an acknowledgement. The sender resends messages if an acknowledgement is not returned within a given time.
  - When a nonconfirmable message is received, receiver does not return an acknowledgement.
- No of response code has also been reduced (to make implementation simpler)
- CoAP also broke away from the Internet Media Type scheme used in HTTP and other protocols and replaced this with a reduced set of Content-Formats.

http://www.iana.org/assignments/core-parameters/

Thanks!