CoAP (Constrained Application Protocol)

Miraj Pandey

BIT V SEM

1 Introduction to CoAP

1.1 What is CoAP?

CoAP (Constrained Application Protocol) is a **lightweight protocol** designed for **constrained devices** and **low-power networks**, making it ideal for **IoT (Internet of Things)** applications. It is designed to operate in **environments with limited bandwidth**, **minimal processing resources**, and **low power**, where traditional web protocols such as **HTTP** would be inefficient.

1.2 Why is CoAP Needed?

Traditional protocols like **HTTP** are **resource-intensive**, relying on **TCP** for reliable, connection-oriented communication. These are not suitable for low-power, low-bandwidth environments. **CoAP** solves this by:

- Using **UDP** for **faster** communication and **low-power** consumption.
- Supporting **multicast** communication for efficient data distribution.
- Enabling **asynchronous communication** with **confirmable (CON)** and **non-confirmable (NON)** messages.
- Providing **RESTful communication**, similar to HTTP.

2 Key Features of CoAP

- **Lightweight**: Ideal for **constrained devices** and **networks**.
- **UDP-Based**: Uses **UDP** for **low-latency** and **low-power** communication.
- **RESTful Interface**: Implements the **GET**, **POST**, **PUT**, **DELETE** methods for resource manipulation, similar to HTTP.
- **Multicast Support**: CoAP supports **multicast communication** for efficient interaction with multiple devices.
- **Proxy Support**: Can interface with **HTTP-based services** via proxies.
- **Security**: Supports **DTLS (Datagram Transport Layer Security)** for **encryption** and **authentication**.
- **Asynchronous Communication**: Uses **confirmable (CON)** and **non-confirmable (NON)** messages for flexible communication.

3 CoAP vs. HTTP - Key Differences

Feature	CoAP	HTTP
Transport Protocol	UDP	TCP
Overhead	Low	High
Multicast Support	Yes	No
Power Consumption	Low	High
Security	DTLS	TLS
Message Size	Small (binary format)	Large (text-based)
Best For	IoT, sensor networks	Web applications

4 CoAP Protocol Stack

The CoAP protocol stack is designed to operate efficiently over constrained networks. It sits atop **UDP** and provides lightweight communication features.

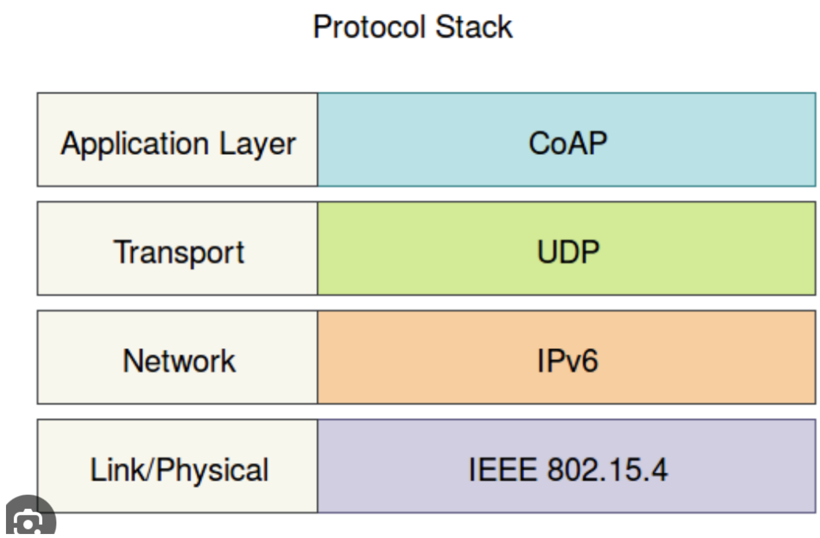

Figure 1: CoAP Protocol Stack

4.1 Explanation of Each Layer

- **Physical Layer**: Refers to the **radio communication** medium used for transmission, such as Wi-Fi, LoRa, Bluetooth, Zigbee, or IEEE 802.15.4.

- **Data Link Layer**: Handles **framing** and **reliability** of data transmission. It ensures data is transferred between devices on the same network, using protocols like **6LoWPAN**.

- **Network Layer**: Deals with **routing** and **addressing**, using protocols like **IPv6** and **RPL**.

- **Transport Layer**: Enables communication between devices using **UDP** and **DTLS** for security, ensuring **low-power** and **low-latency communication**.

- **Application Layer**: CoAP operates in the application layer, allowing devices to request and respond to resources like HTTP but optimized for **constrained environments**.

5 CoAP Communication Model

The CoAP communication model follows a **client-server architecture**:

- **CoAP Client**: Initiates requests, such as GET to retrieve data.
- **CoAP Server**: Responds to the client with the requested data.
- **CoAP Proxy**: Acts as an intermediary between CoAP clients and HTTP servers.

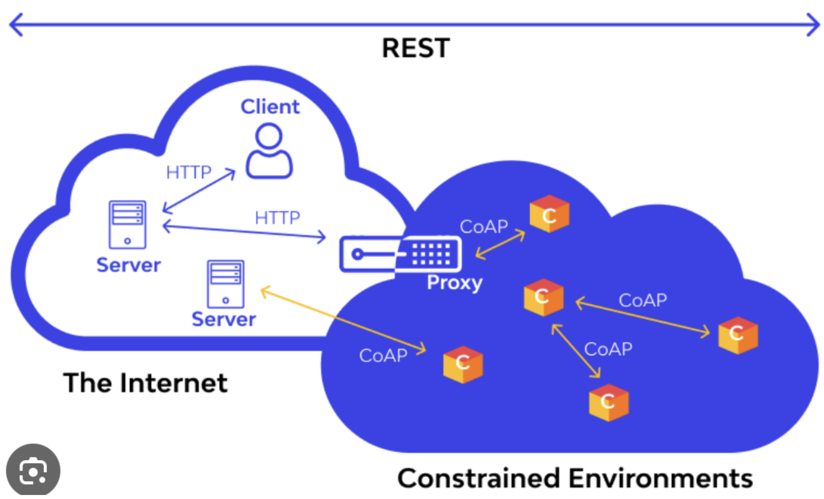

Figure 2: CoAP Client-Server Communication

5.1 How CoAP Uses URLs for Communication

CoAP URLs are similar to HTTP URLs but use `coap://` or `coaps://` for secure communication.

Example:

`coap://iot-device.com/temperature`

For secure communication, `CoAPS` (CoAP Secure) is used with **DTLS** encryption:

`coaps://secure-sensor.com/data`

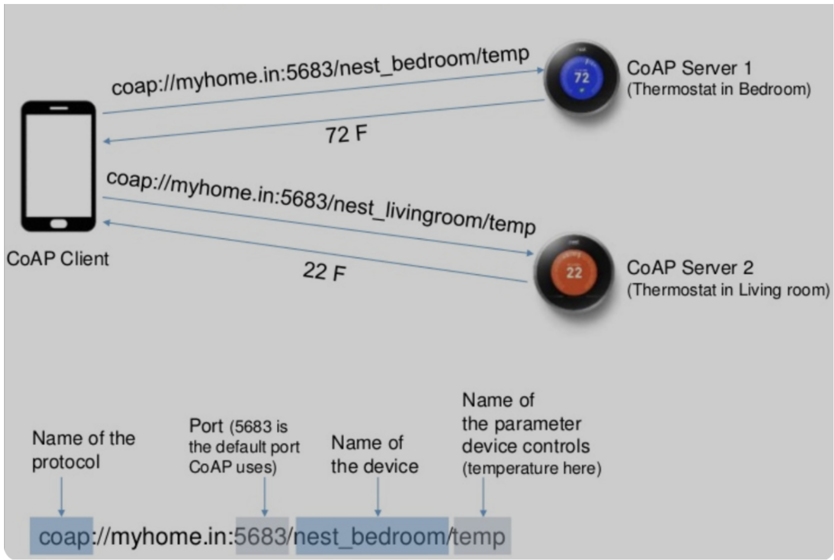

Figure 3: CoAP Client-Server Communication using URL's

6 CoAP Message Structure

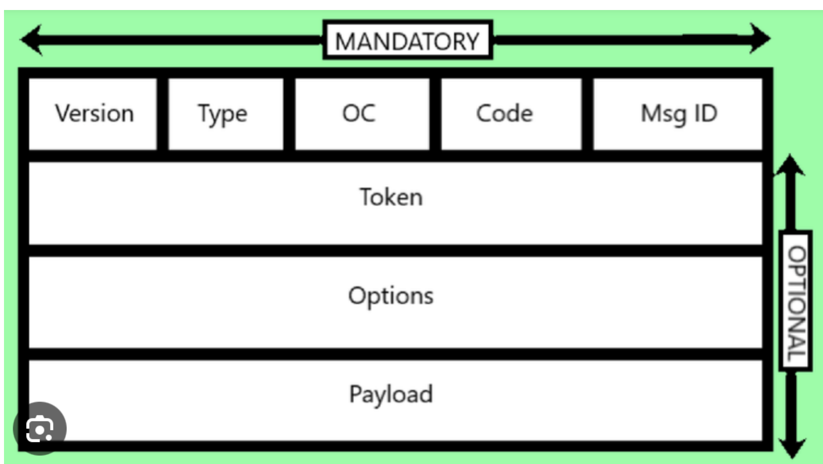

Figure 4: CoAP Message Format

The CoAP message consists of the following fields:

- **Version (2 bits)**: CoAP protocol version (default: 1).

- **Type (2 bits)**: Message type (CON, NON, ACK, RST).
- **Option Count (OC, 4 bits)**: Indicates the number of options included in the message.
- **Code (8 bits)**: Defines the type of message (e.g., GET, POST, 2.05 Content).
- **Message ID (16 bits)**: Unique identifier for matching request-response pairs.
- **Token (0-8 bytes)**: Token used to correlate requests and responses.
- **Options (Variable length)**: Metadata such as URI path, content type, etc.
- **Payload (Variable length)**: The data being transmitted (e.g., sensor readings).

7 CoAP Message Types

CoAP defines four message types:

- **Confirmable (CON) Messages**: These require an acknowledgment. Used for reliable communication.
- **Non-Confirmable (NON) Messages**: These do not require acknowledgment and are used for unreliable, low-priority communication.
- **Acknowledgment (ACK) Messages**: Used to acknowledge the receipt of confirmable messages.
- **Reset (RST) Messages**: Used to reset a transaction in case of errors or unexpected messages.

8 CoAP Security - DTLS (Datagram Transport Layer Security)

8.1 Why CoAP Uses DTLS Instead of TLS?

DTLS is designed for use over **UDP** and provides encryption, integrity, and authentication, ensuring secure communication without the overhead of TCP. It is the appropriate choice for CoAP, which relies on UDP for transport.

9 Summary

CoAP is an **efficient** and **lightweight protocol** tailored for **IoT applications**, offering **low-power**, **low-latency**, and **secure communication** suitable for devices with **constrained resources**. The protocol's use of **UDP**, **asynchronous communication**, and **confirmable/non-confirmable messages** makes it well-suited for scenarios where bandwidth and power are limited.