Protocols of IoT : CoAP

e-Yantra Team
Embedded Real-Time Systems (ERTS) Lab
Indian Institute of Technology, Bombay

IIT Bombay
February 12, 2021

Agenda for Discussion

1. Intro to CoAP
   - What is CoAP?
   - Features of CoAP
   - CoAP-Request Response
2. CoAP Overview
   - How is it made?
   - Message Layer
   - Messages Interaction
   - Request/Response Layer
3. CoAP in Detail
   - Resource Discovery
   - Resource Observation
   - Message Formats and Contents
   - CoAP Architecture
   - Implementation

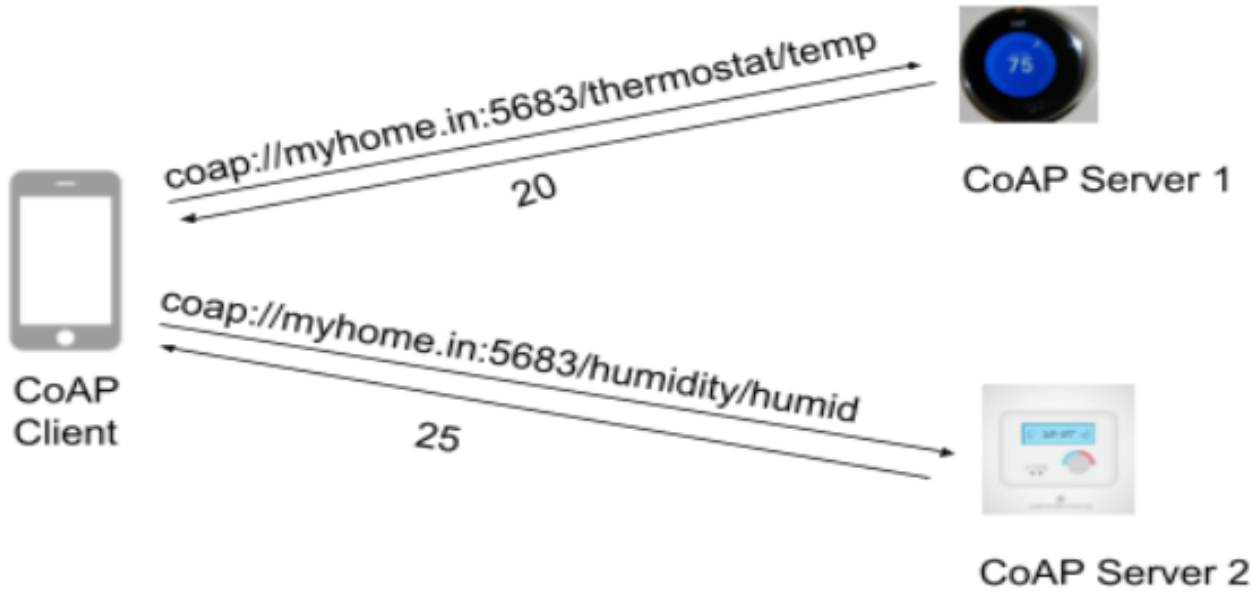
What is CoAP?

- **Constrained Application Protocol**- CoAP
- Designed by the **Internet Engineering Task Force** (IETF).
- Designed for reliability in **low bandwidth** and **high congestion** through its low power consumption and low network overhead.
- CoAP uses **UDP**.
- CoAP can use both **Client/Server** and **Publish/Subscribe model**.

Features of CoAP

- Asynchronous Message exchange
- Low Overhead
- URI representations for resources. coap-URI=**“coap:” “//” host [“:” port] path- [“?” query]**
- TCP complexities are reduced by using UDP.
- Resource discovery capability.
- **Block transfers** for large files.
- **Observe requests** functions to receive automatic server notifications for a resource.

CoAP-Request Response

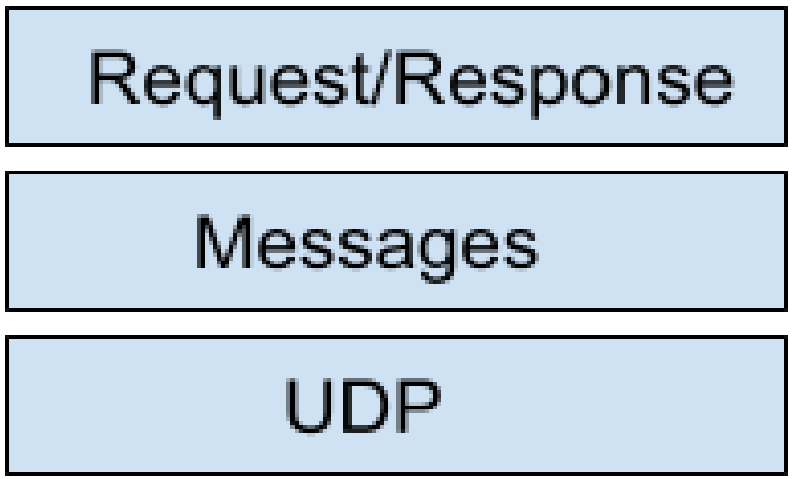


coap://myhome.in:5683/thermostat/temp

- Name of the protocol
- Port (5683 is default port CoAP uses)
- Name of the device
- Name of the parameter device controls (temperature here)

How is it made?

CoAP is made of 2 (sub)layers layered on top of UDP:



- Message layer deals with the UDP and it supports 4 types of messages: **CON, NON, ACK, RST.**
- Request/Response layer manages the interaction between client and server. It contains methods like **GET, PUT, POST and DELETE.**

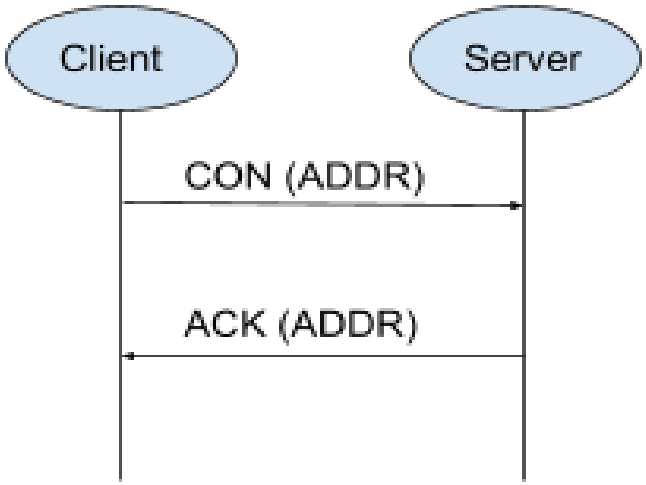
Message Layer

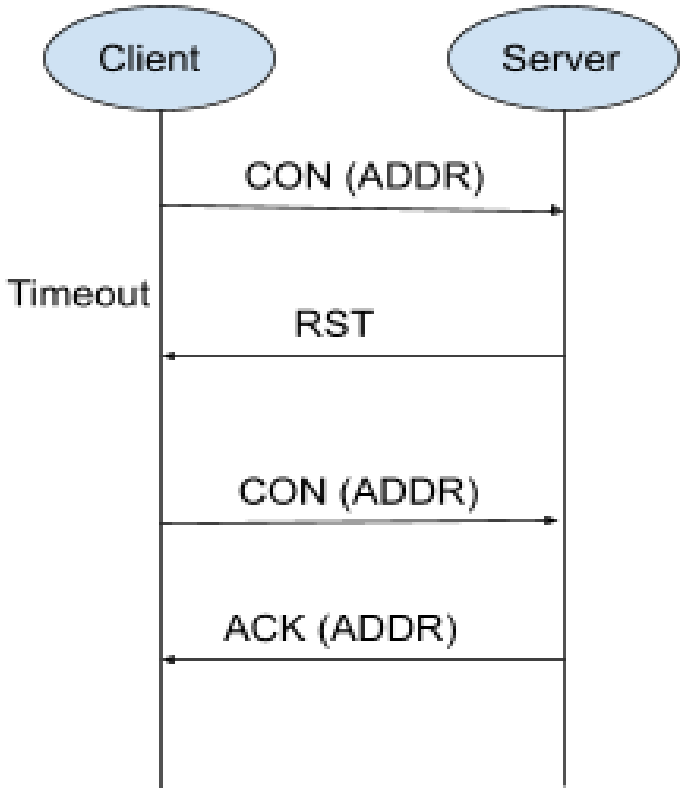
It supports 4 types of Messages:

- **Confirmable (CON)**
- **Non-Confirmable (NON)**
- **Acknowledgement (ACK)**
- **Reset (RST)**

Confirmable (CON)

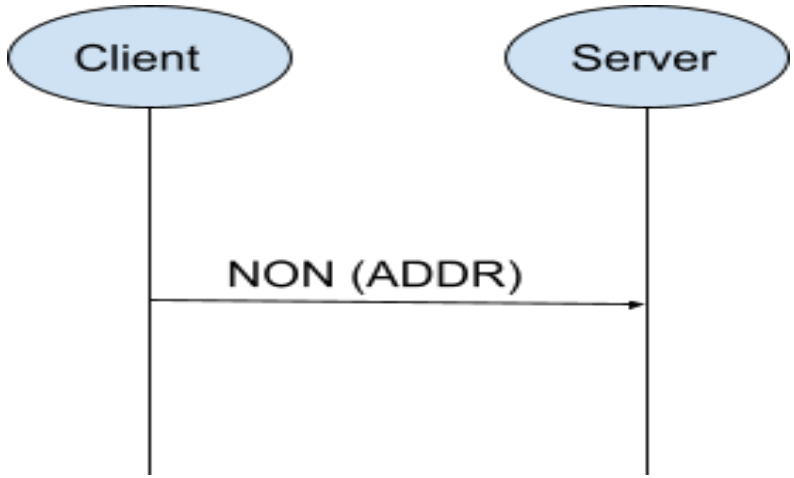
- Reliable messaging
- **Retransmit:** Until acknowledgement arrives
- **Timeout/Recipient fails to process message** - Will send RST with the response.





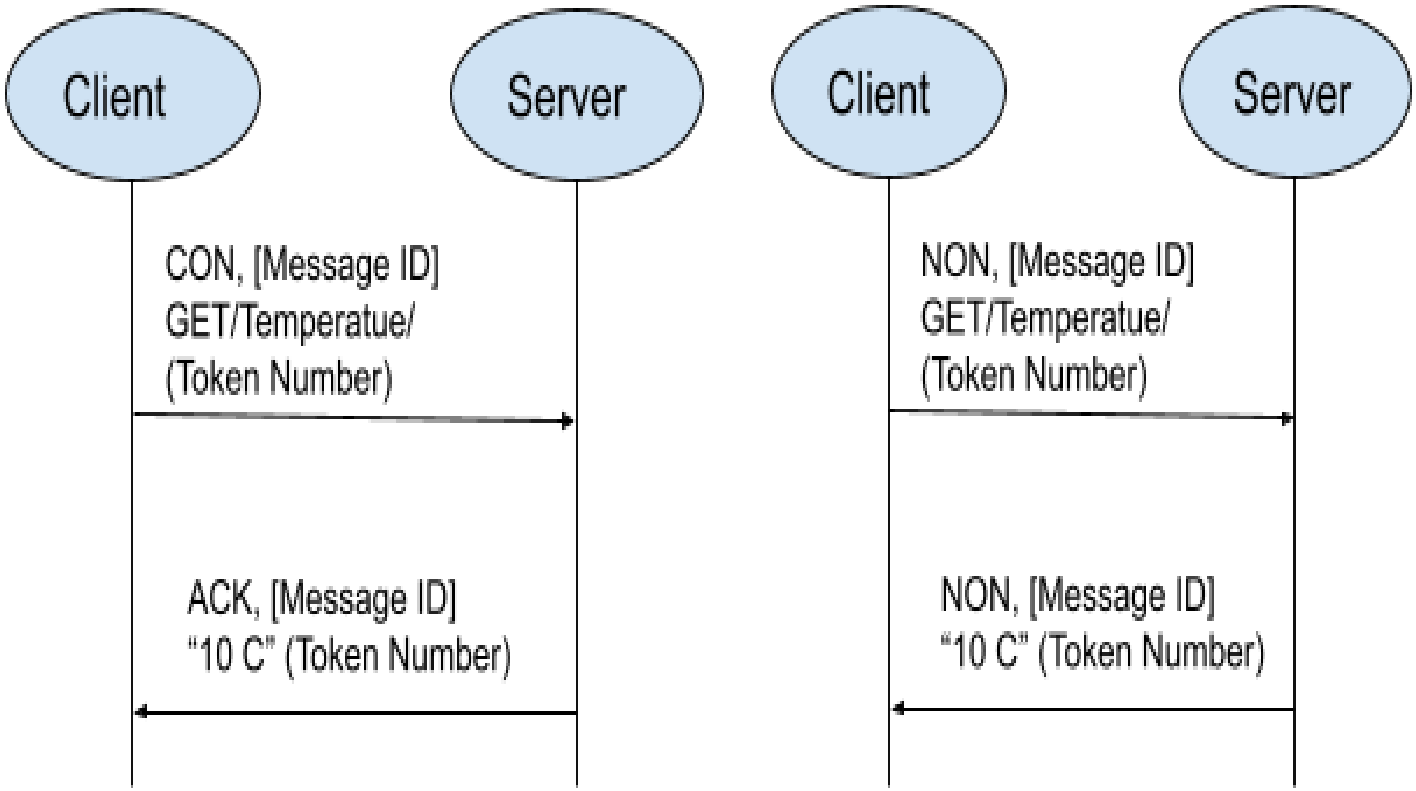
Non-Confirmable (NON)

- Unreliable messaging.
- No ACK here.



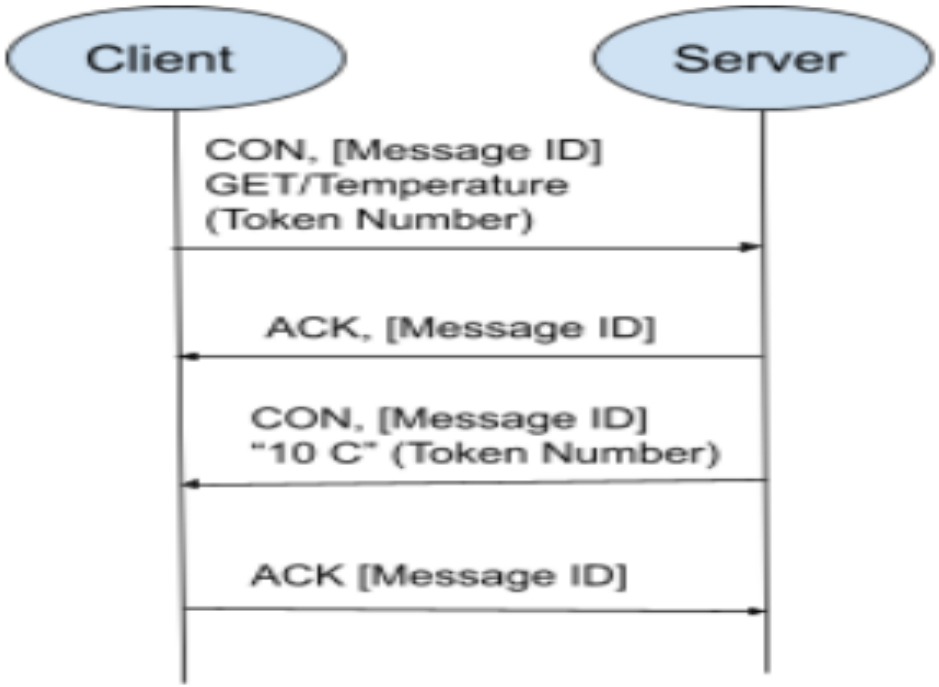
Piggybacked Response

Client sends request using CON type or NON type message and receives response ACK with confirmable message immediately.



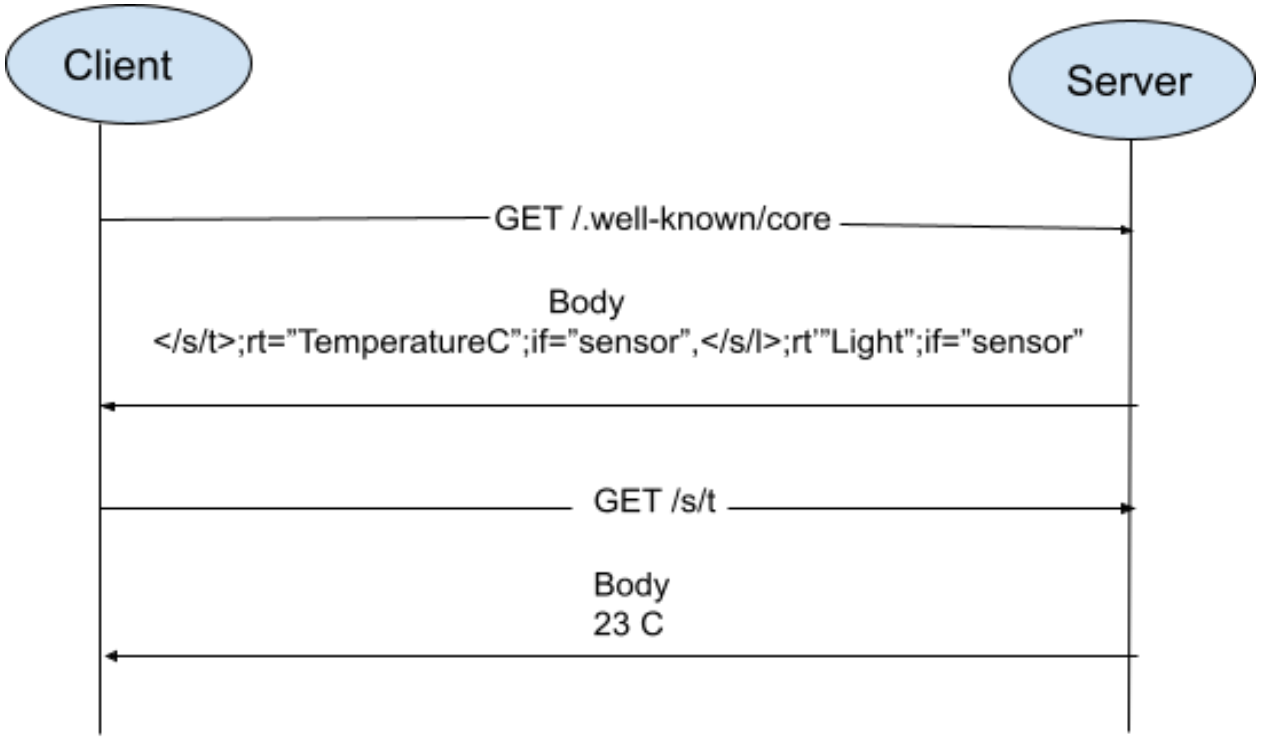
Separate Response

If server receive a CON type message but not able to response this request immediately, it will send an empty ACK in case of client resend this message. When server ready to response this request, it will send a new CON to client and client reply a confirmable message with acknowledgment.



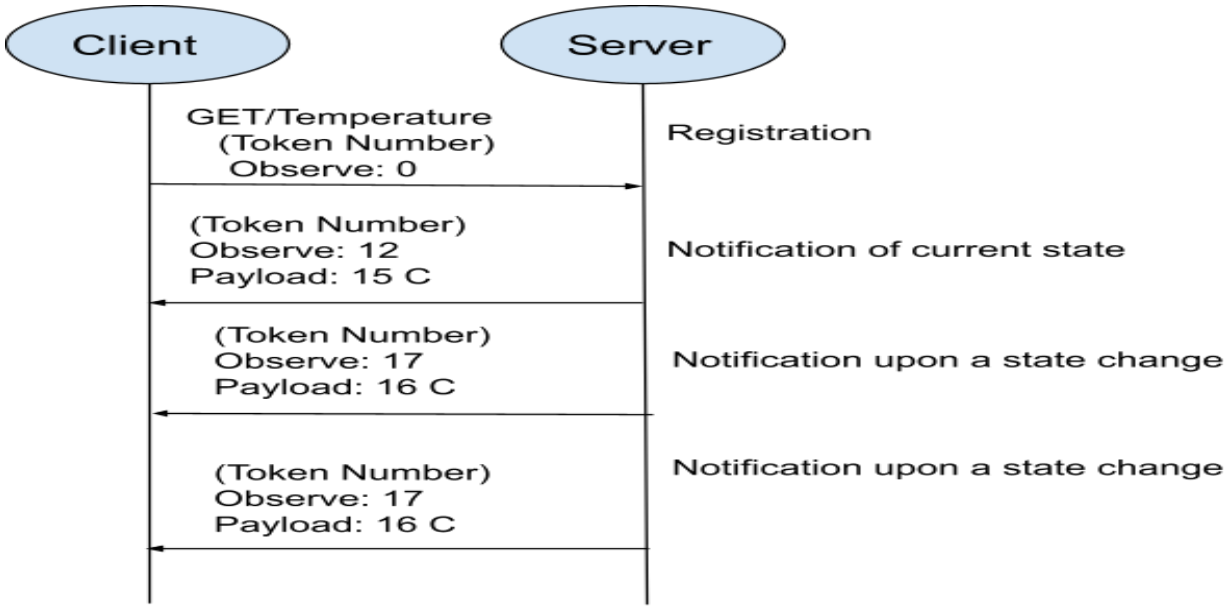
Resource Discovery

- Getting a list of associated resources from the server.
- A well-known relative URI —"/.well-known/core" is defined as a default entry-point for requesting a list of links to resources hosted by a server.

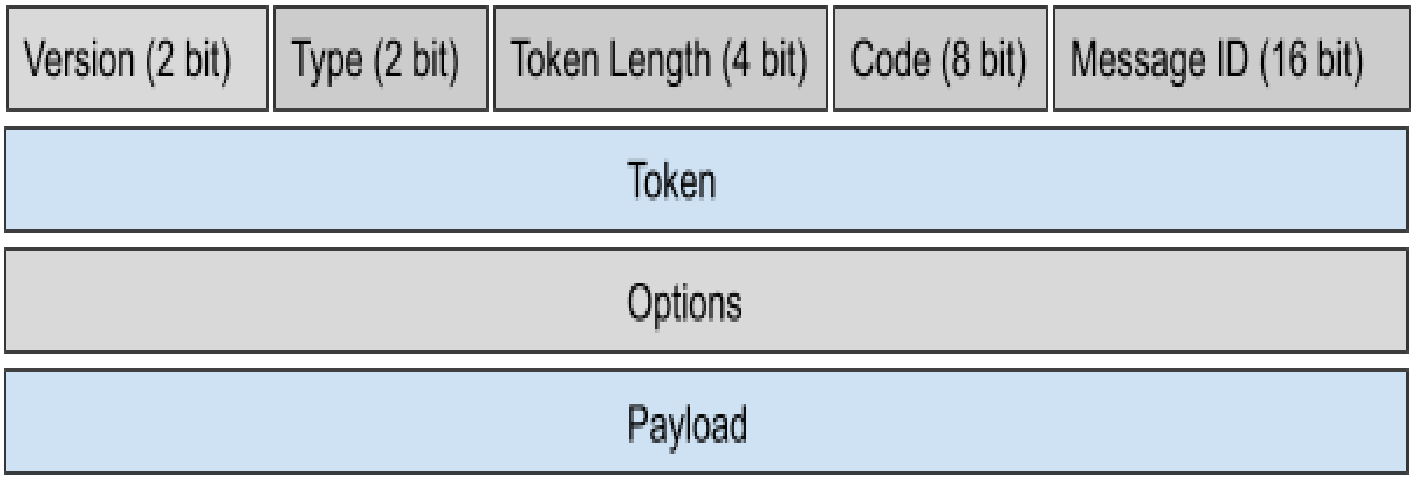


Resource Observation

- Observe requests are used to receive automatic server notifications for a resource.
- The client becomes an observer of an observable resource by sending an observe request to the resource.
- Observe is to retrieve a representation of a resource and keep this representation updated by the server over a period of time.



Message Formats and Contents



- **Version:** It mentions the CoAP version. Set to 1.
- **Type:** Indicates message type viz. CON(0), NON(1), ACK(2), RST(3).
- **Token Length:** Indicates length to token (0-8 bytes)
- **Code:**Response Code
- **Message ID:** Identifier for each message sent
- **Token:** Optional response matching token

CoAP Method Codes

- **c.dd**
  - **c**- Code Class
  - **dd**- Decimal Number for Details

Code	Name
0.01	GET
0.02	POST
0.03	PUT
0.04	DELETE

l0.25

- Code Class
  - **0**- Request
  - **2**- Response
  - **4**- Client Error Response
  - **5**- Server Error Response
- Code 0.00 indicates Empty Message

CoAP Status Codes

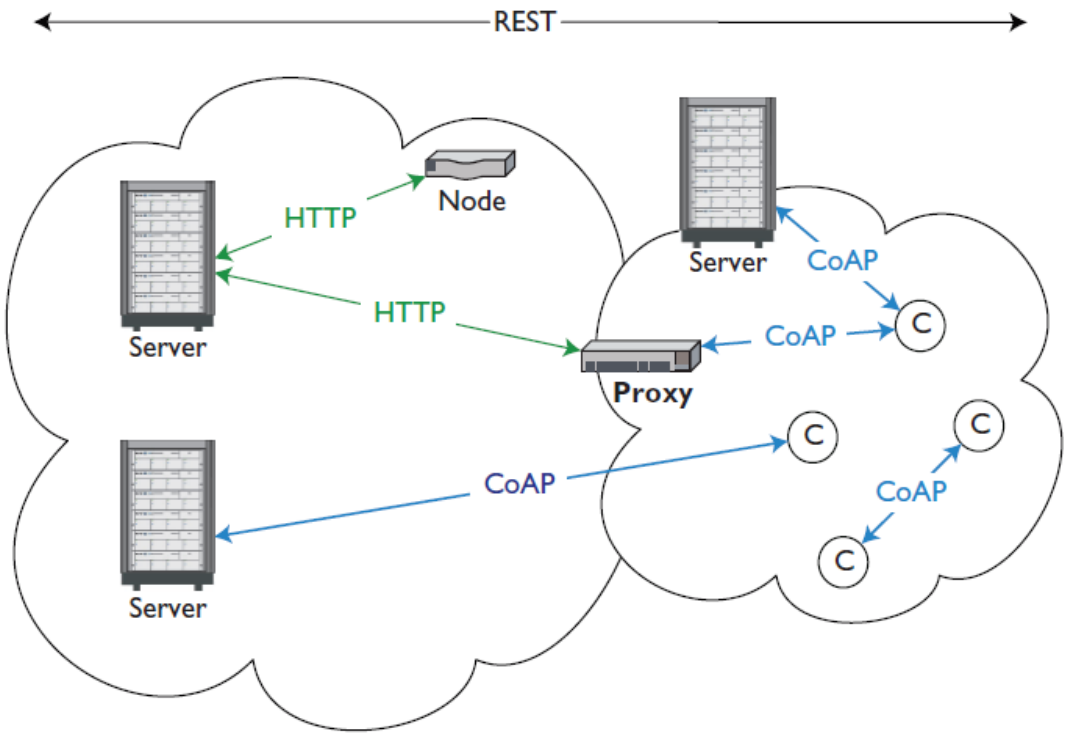
CoAP Status Code	Description
2.01	Created
2.02	Deleted
2.03	Valid
2.04	Changed
2.05	Content
4.00	Continue
4.01	Bad Request
4.02	Unauthorized
4.03	Bad Option
4.04	Forbidden
4.05	Not Found
4.06	Method Not Allowed
4.08	Not Acceptable

Contd.

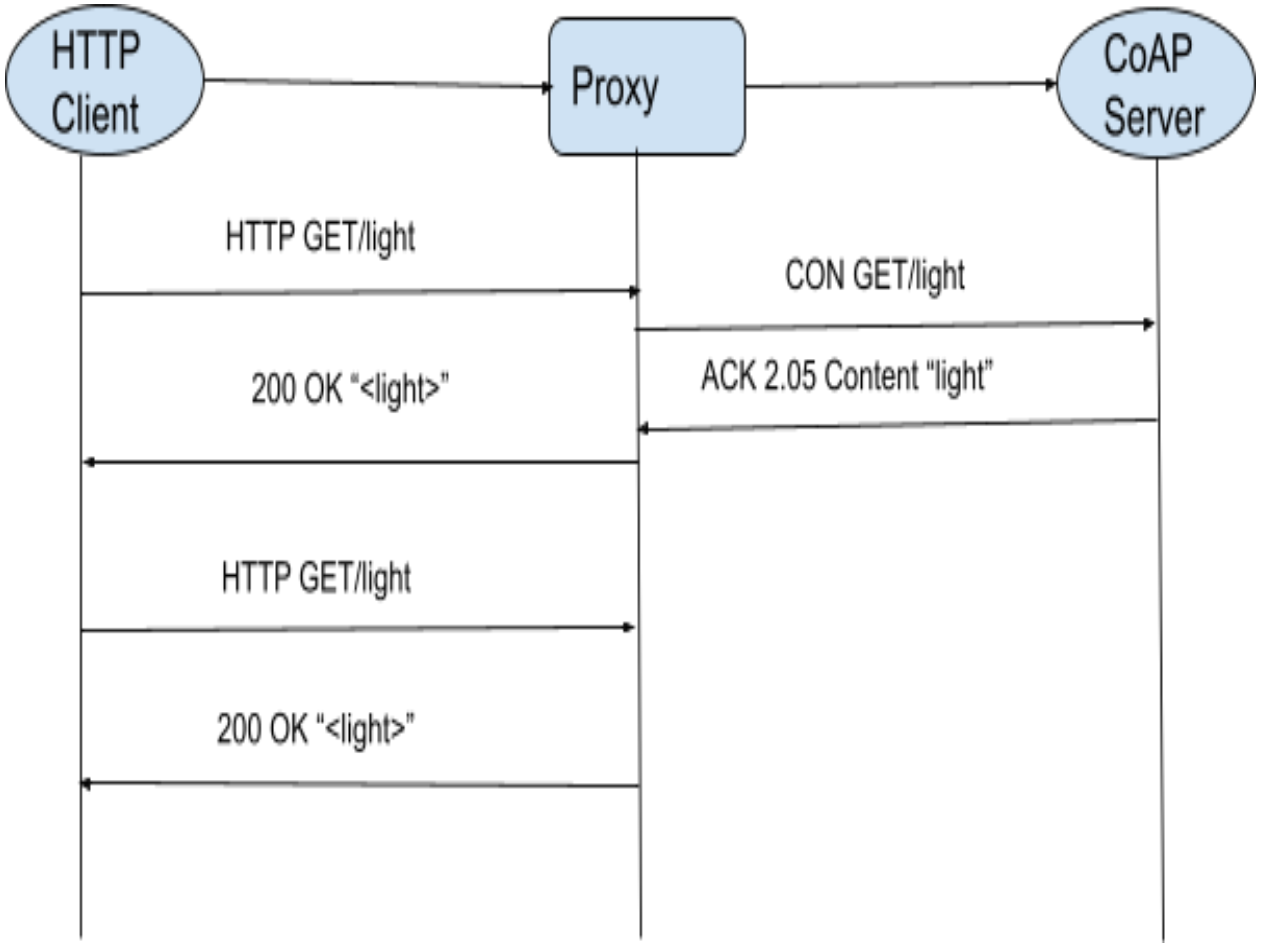
CoAP Status Code	Description
4.12	Request Entity Incomplete
4.13	Request Entity Too Large
4.15	Unsupported Content-Format
5.00	Internal Server Error
5.01	Not Implemented
5.02	Bad Gateway
5.03	Service Unavailable
5.04	Gateway Timeout
5.05	Proxying Not Supported

CoAP Architecture

- CoAP works with CoAP directly, but to work with HTTP it uses Proxy.
- Proxy behaves like a Server to a Client and then a Client to a Server.



CoAP Proxy Caching



Implementation using Coapthon

CoAP Client

```
from coapthon.client.helperclient import HelperClient

if __name__ == '__main__':
    host="13.250.13.141"
    port=5683
    client = HelperClient(server=(host,port))
    token = " "
    path = f"api/v1/{token}/attributes"
    payload = "{'temperature':'45', 'humidity':'70', '
    pressure':'70'}"
    response = client.post(path, payload=payload)
    print("response.code", response.code)
    client.stop()
```

References

1. CoAP https://tools.ietf.org/html/draft-shelby-core-resource-directory-04#page-8
2. CoAP https://www.sciencedirect.com/topics/engineering/constrained-application-protocol
3. CoAPthon https://github.com/Tanganelli/CoAPthon
4. txThings https://pypi.org/project/txThings//
5. libcoap https://libcoap.net/

Thank You!

Post your queries at: helpdesk@e-yantra.org/