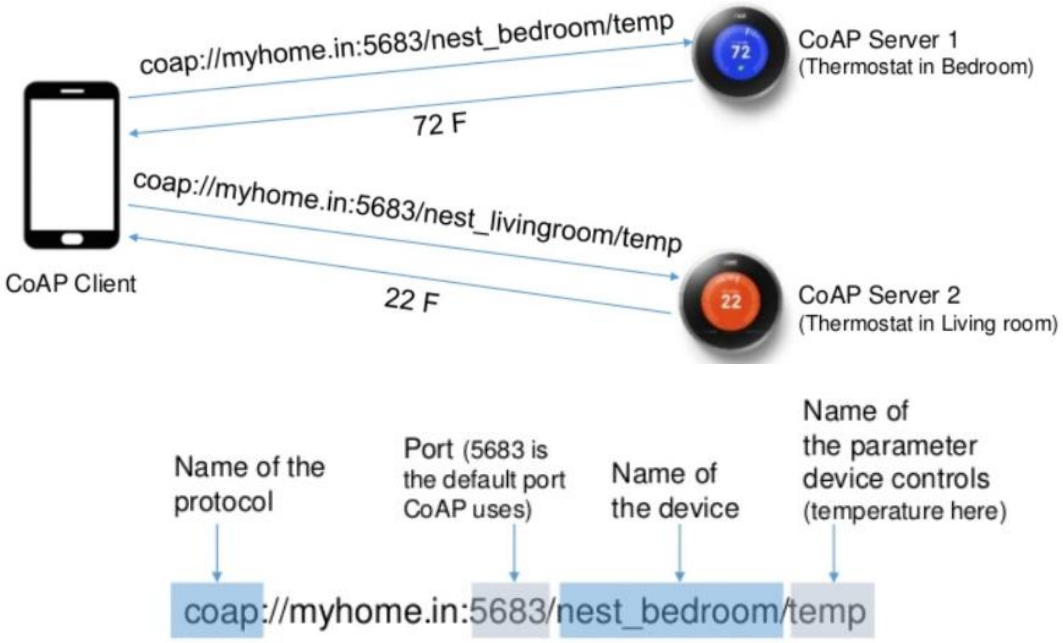
Constrained Application Protocol (CoAP):

Ref: https://datatracker.ietf.org/doc/html/rfc7252#section-1.1

Ref: https://devopedia.org/constrained-application-protocol

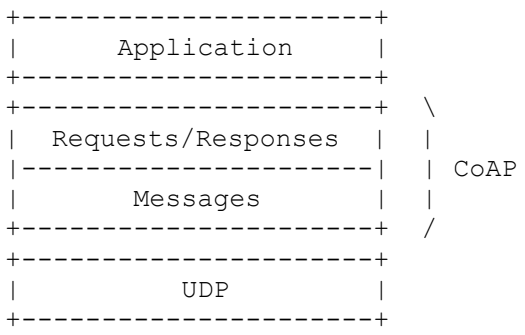
- The **Constrained Application Protocol (CoAP)** is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks.
- The **nodes** often have 8-bit microcontrollers with small amounts of ROM and RAM, while constrained networks such as IPv6 over Low-Power Wireless Personal Area **Networks** (6LoWPANs) often have high packet error rates and a typical throughput of 10s of kbit/s.
- The protocol is designed for machine- to-machine (M2M) applications such as smart energy and building automation.
- CoAP provides **a request/response interaction model** between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types.
- CoAP is designed **to easily interface with HTTP** for integration with the Web while meeting specialized requirementssuch as multicast support, very low overhead, and simplicity for constrained environments.



Constrained Application Protocol

- The interaction model of CoAP is similar to the **client/server** modelof HTTP.
- However, **machine-to-machine interactions** typically resultin a CoAP implementation acting in both client and server roles.
- A CoAP request is equivalent to that of HTTP and is sent by a client to**request** an action (using a Method Code) on a resource (identified by a URI) on a server.
- The server then sends a **response** with a ResponseCode; this response may include a resource representation.

- Unlike HTTP, CoAP deals with these interchanges **asynchronously** over a datagram-oriented transport such as **UDP**.
- This is done logicallyusing **a layer of messages** that supports optional reliability (withexponential back-off).
- CoAP defines four types of messages:
    - Confirmable
    - Non-confirmable
    - Acknowledgement
    - Reset.
- **Method Codes and Response Codes** included in some of these messages make them carry requests or responses.
- The basic exchanges of the four types ofmessages are somewhat orthogonal to the request/responseinteractions; requests can be carried in Confirmable and Non-confirmable messages, and responses can be carried in these as wellas piggybacked in Acknowledgement messages.
- One could think of **CoAP logically as using a two-layer approach**, **a CoAP messaging layer used to deal with UDP and the asynchronous nature of the interactions, and the request/response interactionsusing Method and Response Codes** (see following Figure). CoAP is however asingle protocol, with messaging and request/response as just featuresof the CoAP header.



- CoAP makes use of GET, PUT, POST, and DELETE methods in a similar manner to HTTP.
- Where CoAP differs from HTTP is that UDP is used for transport instead of TCP. UDP handshaking is lighter and easier to implement on microcontrollers. CoAP header is only 4 bytes. CoAP can also use UDP's broadcast and multicast features. CoAP communication is using connectionless datagrams. Because datagrams are used, SMS and other packet-based protocols may be used**.**

Messaging Model

- CoAP uses **a short fixed-length binary header (4 bytes)** that may be followed by **compact binary options** and **a payload**.
- This message format is shared by requests and responses.
- The CoAP message format is specified later. Each message contains a Message ID usedto detect duplicates and for optional reliability.
- Reliability is provided by marking a message as **Confirmable** (CON.A Confirmable message is retransmitted using a default timeout and **exponential back-off between retransmissions**, **until** the recipientsends an **Acknowledgement message** (ACK)

with the same Message ID in this example, 0x7d34) from the corresponding endpoint; see the following Figure 2.

- When a recipient is not at all able to process a Confirmable message(i.e., not even able to provide a suitable error response), itreplies with a **Reset message (RST)** instead of an Acknowledgement(ACK).

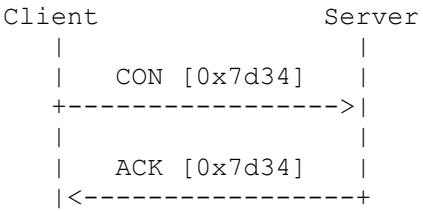


Figure 2: reliable Message Transmission

- A message that does not require reliable transmission (for example,each single measurement out of a stream of sensor data) can be sentas a **Non-confirmable** message (NON).
- These are **not acknowledged**, butstill have **a Message ID for duplicate detection** (in this example,0x01a0); see Figure 3.
- When a recipient is not able to process aNon-confirmable message, it may reply with a **Reset message (RST**)

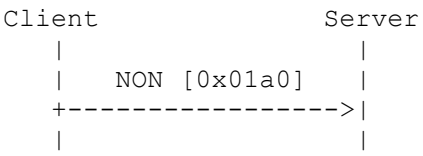


Figure 3: Unreliable Message Transmission

Request/Response Model

- CoAP request and response semantics are carried in CoAP messages, which include either a Method Code or Response Code, respectively.
- **Optional (or default) request and response information, such as the URI and payload media type are carried as CoAP options**.
- A Token is used to match responses to requests independently from the underlying messages
- A request is carried in a Confirmable (CON) or Non-confirmable (NON) message, and, if immediately available, the response to a request carried in a Confirmable message is carried in the resulting Acknowledgement (ACK) message. This is called a piggybacked response (There is no need for separately acknowledging a piggybacked response, as the client will retransmit the request if the acknowledgement message carrying the piggybacked response is lost.)
- Two examples for a basic GET request with piggybacked response are shown in Figure 4, one successful, one resulting in a 404 (Not Found) response.

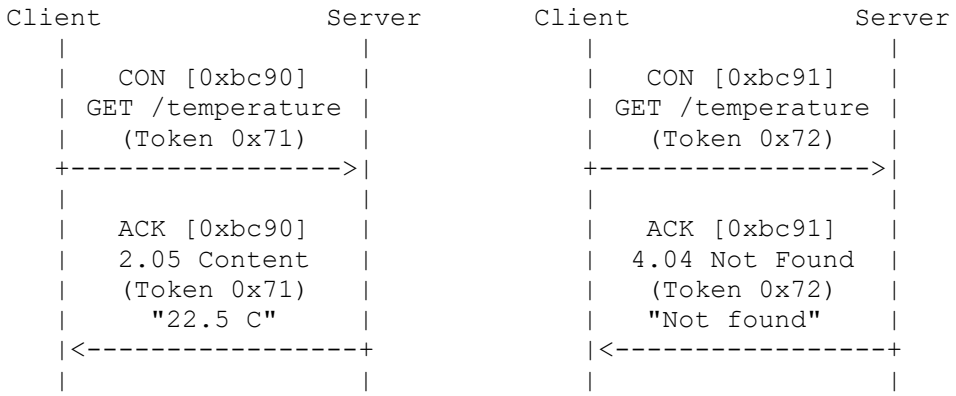


Figure 4: Two GET Requests with Piggybacked Responses

- If the server is not able to respond immediately to a request carried in a Confirmable message, it simply responds with an Empty Acknowledgement message so that the client can stop retransmitting the request.
- When the response is ready, the server sends it in a new Confirmable message (which then in turn needs to be acknowledged by the client). This is called a "separate response", as illustrated in Figure 5.

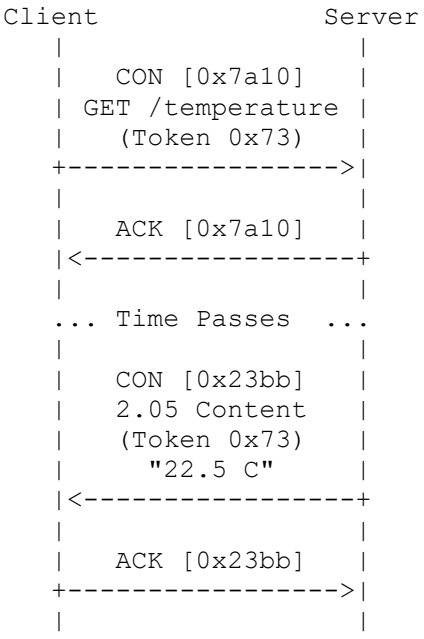


Figure 5: A GET Request with a Separate Response

- If a request is sent in a Non-confirmable message, then the response is sent using a new Non-confirmable message, although the server may instead send a Confirmable message. This type of exchange is illustrated in Figure 6.

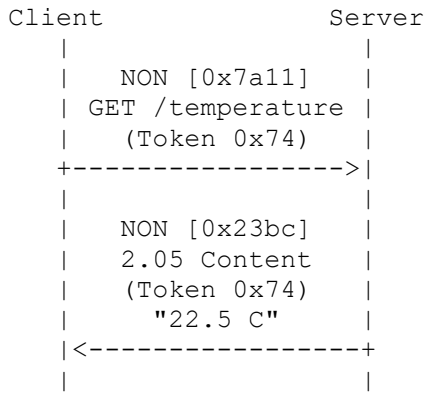


Figure 6: A Request and a Response Carried in Non-confirmableMessages

Reset Message

A Reset message indicates that a specific message (Confirmable or Non-confirmable) was received, but some context is missing to properly process it. This condition is usually caused when the receiving node has rebooted and has forgotten some state that would be required to interpret the message. Provoking a Reset message (e.g., by sending an Empty Confirmable message) is alsouseful as an inexpensive check of the liveness of an endpoint ("CoAP ping").

Message Format

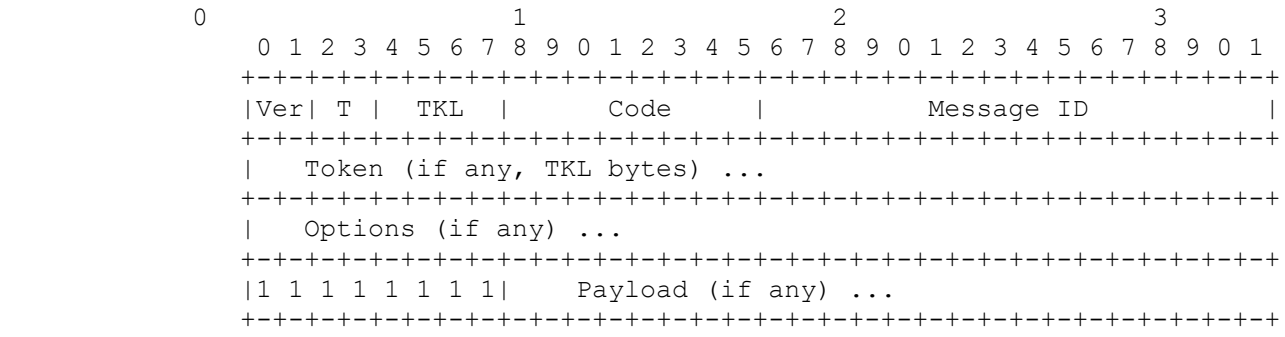


Figure 7: Message Format

The fields in the header are defined as follows:

Version (Ver): 2-bit unsigned integer. Indicates the CoAP versionnumber. Implementations of this specification MUST set this fieldto 1 (01 binary). Other values are reserved for future versions.Messages with unknown version numbers MUST be silently ignored.

Type (T): 2-bit unsigned integer. Indicates if this message is oftype Confirmable (0), Non-confirmable (1), Acknowledgement (2), or Reset (3).

Token Length (TKL): 4-bit unsigned integer. Indicates the length ofthe variable-length Token field (0-8 bytes). Lengths 9-15 are reserved, MUST NOT be sent, and MUST be processed as a message format error.

Code: 8-bit unsigned integer, split into a 3-bit class (most significant bits) and a 5-bit detail (least significant bits),documented as "c.dd" where "c" is a digit from 0 to 7 for the3-bit subfield and "dd" are two digits from 00 to 31 for the 5-bitsubfield. The class can indicate a request (0), a success response (2), a client error response (4), or a server errorresponse (5).

There are 3 classes of Response Codes:

2 - Success: The request was successfully received, understood, and accepted.

4 - Client Error: The request contains bad syntax or cannot be fulfilled.

5 - Server Error: The server failed to fulfill an apparently valid request.

(All other class values are reserved.) As a special case, Code 0.00 indicates an Empty message. In case of a request, the Code field indicates the Request Method; in case of a response, a Response Code.
Method codes:

Code	Name	Reference
0.01	GET	[RFC7252]
0.02	POST	[RFC7252]
0.03	PUT	[RFC7252]
0.04	DELETE	[RFC7252]

Table 5: CoAP Method Codes

All other Method Codes are Unassigned.
Response code:

2.01 Created
2.02 Deleted
…
2.05 Content
…
4.00 Bad Request
…
4.03 Forbidden
4.04 Not Found
…
5.00 Internal Server Error
…

Message ID: 16-bit unsigned integer in network byte order. Used to detect message duplication and to match messages of type Acknowledgement/Reset to messages of type Confirmable/Non-confirmable.

The header is followed by the Token value, which may be 0 to 8 bytes, as given by the Token Length field. The Token value is used to correlate requests and responses.

.

Header and Token are followed by zero or more Options. An Option can be followed by the end of the message, by another Option, or by the Payload Marker and the payload.

- **Optional (or default) request and response information, such as the URI and payload media type are carried as CoAP options**.