CoAP - Constrained Application Protocol

KEY FEATURES :

- single application layer (message sublayer, request/response sublayer on message sublayer)

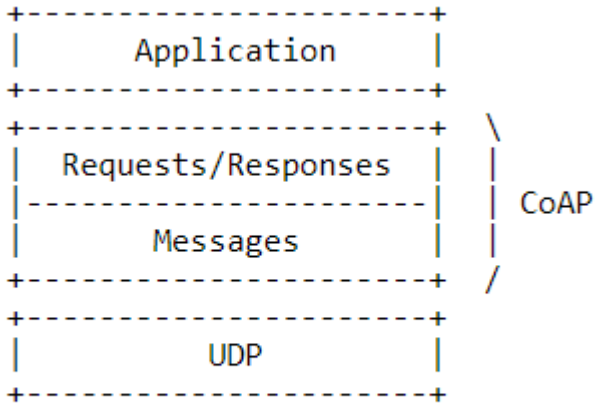

Figure 1: Abstract Layering of CoAP

- for low-power, lossy networks, constrained environments, for M2M applications
- on top of Ipv6
- UDP binding with optional reliability supporting unicast/multicast requests
- asynchronous message exchanges
- provides request/response interaction between endpoints
- support discovery of services and resources
- support concepts of Web (URIs, Internet media types (Content-type), ...)
- support simple proxy and caching capabilities
    - cache is on one of the endpoints or on intermediary (proxy)
    - proxy used for : limit network traffic, improve performance, access resources of sleeping devices, security
    - CoAp is similar to HTTP -> support cross-protocol proxy
    - support GET, PUT, POST, DELETE
        - it converts method/response code, media type, options
- support security binding to Datagram Transport Layer Security

MESSAGE TRANSMISSION :

As CoAP is bound to unreliable transports such as UDP, CoAP messages may arrive out of order, appear duplicated, or go missing without notice. For this reason, CoAP implements a lightweight reliability mechanism, without trying to re-create the full feature set of a transport like TCP. It has the following features:

o Simple stop-and-wait retransmission reliability with exponential

back-off for Confirmable messages.

o Duplicate detection for both Confirmable and Non-confirmable

messages.

MESSAGE TYPES - Realiability Mechanism :

- Confirmable message : requires ack or reset as return message
- Non-confirmable message : no need of return message
- Acknowledgements : confirmation of receiving message, doesnt indicate success or failure of request
- Reset message : indicates that (non)confirmable message arrived, but because of some missing context, server could not proccess it
  - it can be used as CoAP ping (empty CON message)

Empty message : it contains only and only fixed 4B header -> code : 000

MESSAGING MODEL :

- exchanging of messages over UDP between endpoints
- 4B header woth optional compant binary options and payload
- every message contains messageID (2B -> 250 messages in a second)
  - for detection of duplicates and for optional reliability
- if receiver(server) is not able to process NON message , server sends RST message
- example

  1. client: CON(messageID)

  2. server: ACK (messageID)

  1. client: NON(nova messageId)

  2 . server: (may reply with RST)

REQUEST/RESPONSE MODEL :

- URI, payload media type and other HTTP options are stored in CoAP options.
- **Token is used for matching response to request** (**it is different from messageID !!!**)
- Request is part of CON/NON message, response is part of ACK (piggybacked response)
- examples of piggybacked response (response in ACK) :

  1. client: CON (messageID) , GET /temp [URI] , token

  2. server: ACK (messageID) , 205, token, payload [22.5 C]

  1. client: CON (messageID), GET /temp [URI], token

  2. server: ACK (messageID), 404, token, payload ["not found"]

- separate response -> if server cannot immediately reply :

  1. client: CON (messageID), GET /temp [URI], token

  2. server: cannot immediately reply, sends empty ACK (messageID)

  3. time pass ....

  4. server: server is able to reply -> sends new CON (**new** messageID), 205, token (**is the same**), payload [22.5 C]

  5. client: ACK (messageID)

- if client sends NON:

  1. client: NON (messageID), GET /temp, token

  2. server: NON (**new** messageID), 205, token (**is the same**), payload [22,5C]

	CON	NON	ACK	RST
Request	X	X	-	-
Response	X	X	X	-
Empty	*	-	X	X

MESSAGE FORMAT :

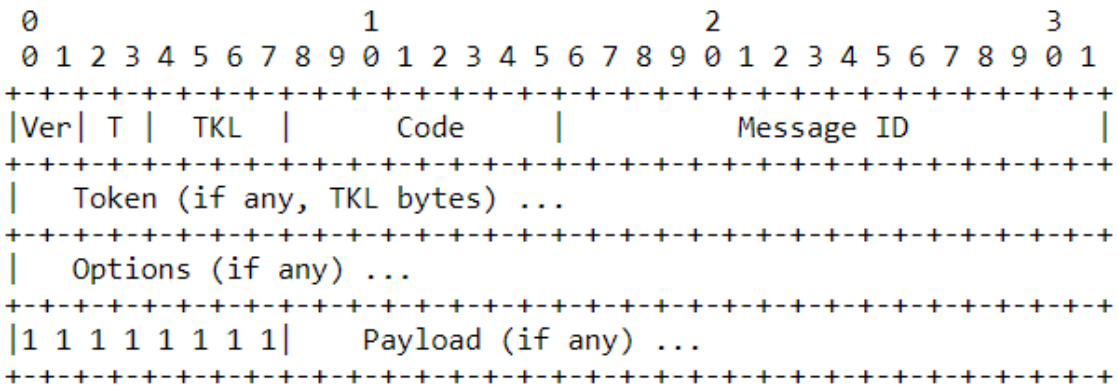

Figure 7: Message Format

- CoAP uses data section of UDP datagram
- CoAP does not support UDP-lite or UDP zero checksum
- *Fixed 4B header* :

  1. Version : 2b -> CoAp version number – must be '01'

  2. Type : 2b -> CON = 0; NON = 1, ACK = 2 , Reset = 3

3. Token Length : 4b -> indicates length of token (0-8B)

4. Code : 1B -> 3b class, 5bit detail [4.04]

5. messageID : 2B -> for matching ACK/RST to CON/NON

- variable-length Token value (up to 8B)
  - for corelation of request/response
  - *It is simply said requestID*
- CoAP options in Type-length-value format (0 - ... B)
  - CoAP options format : option number, length of option value, option value
  - CoAP defines a single set of options that are used in both requests and responses:

No.	C	U	N	R	Name	Format	Length	Default
1	x			x	If-Match	opaque	0-8	(none)
3	x	x	-		Uri-Host	string	1-255	(see below)
4				x	ETag	opaque	1-8	(none)
5	x				If-None-Match	empty	0	(none)
7	x	x	-		Uri-Port	uint	0-2	(see below)
8				x	Location-Path	string	0-255	(none)
11	x	x	-	x	Uri-Path	string	0-255	(none)
12					Content-Format	uint	0-2	(none)
14		x	-		Max-Age	uint	0-4	60
15	x	x	-	x	Uri-Query	string	0-255	(none)
17	x				Accept	uint	0-2	(none)
20				x	Location-Query	string	0-255	(none)
35	x	x	-		Proxy-Uri	string	1-1034	(none)
39	x	x	-		Proxy-Scheme	string	1-255	(none)
60			x		Size1	uint	0-4	(none)

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

Media type	Encoding	ID	Reference
text/plain; charset=utf-8	-	0	[RFC2046] [RFC3676] [RFC5147]
application/link-format	-	40	[RFC6690]
application/xml	-	41	[RFC3023]
application/octet-stream	-	42	[RFC2045] [RFC2046]
application/exi	-	47	[REC-exi-20140211]
application/json	-	50	[RFC7159]

Table 9: CoAP Content-Formats

- payload marker -> indicates the end of CoAP options and start of the payload
  - (1B - 0xFF)
  - If it is missing, payload also missing

- Implementation Note: The byte value 0xFF may also occur within an option length or value, so simple byte-wise scanning for 0xFF is not a viable technique for finding the payload marker. The byte 0xFF has the meaning of a payload marker only where the beginning of another option could occur.

- Payload (up to the end of UDP data section)

Examples at : https://datatracker.ietf.org/doc/html/rfc7252#appendix-A