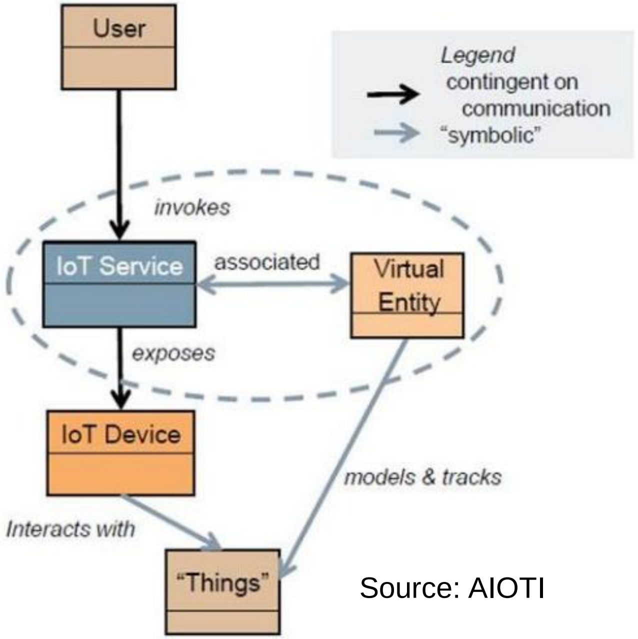
EIOT

Constrained Application Protocol (CoAP)

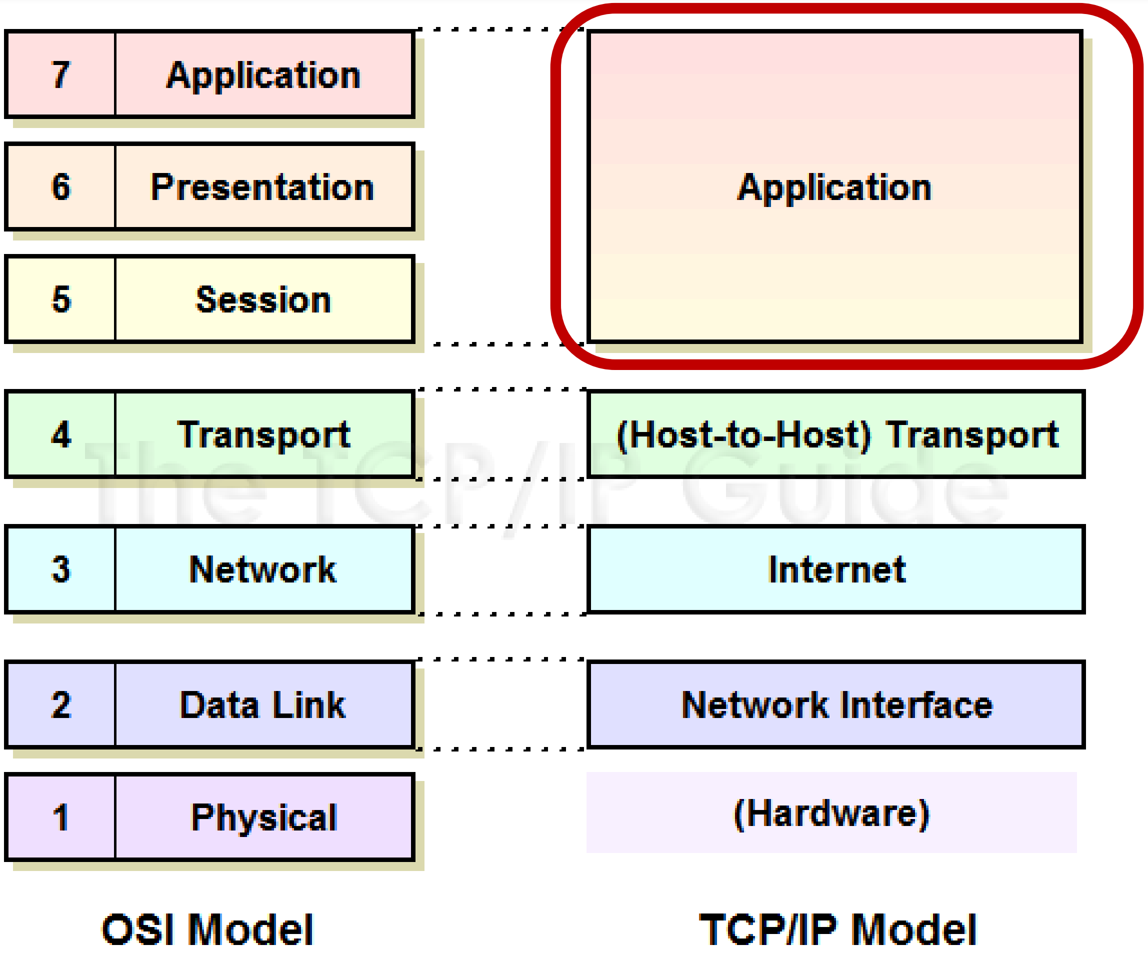
Jarosław Domaszewicz

Instytut Telekomunikacji Politechniki Warszawskiej



Source: AIOTI

APPLICATION LAYER (1/3)

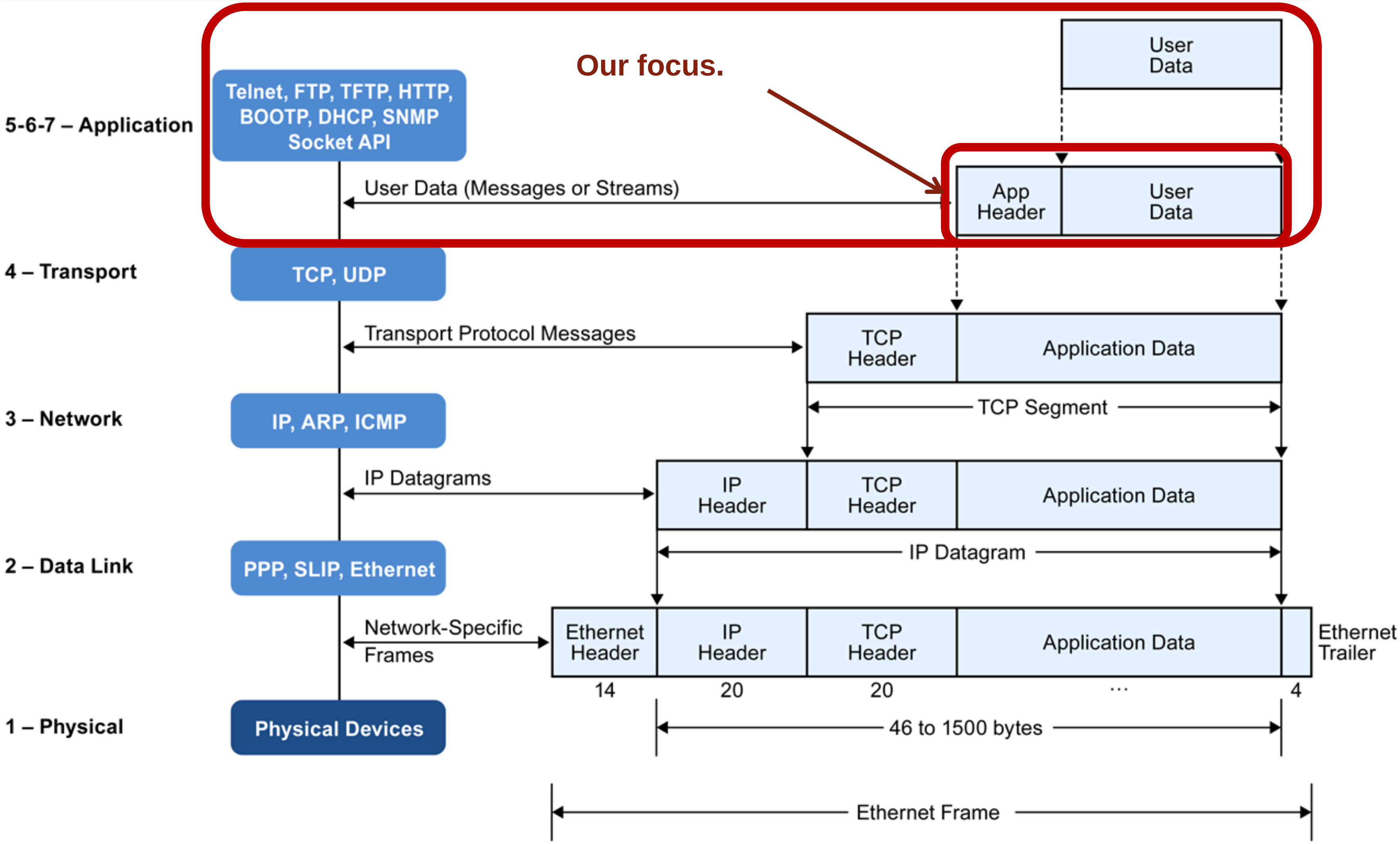


7	Application	Application
6	Presentation	
5	Session	
4	Transport	(Host-to-Host) Transport
3	Network	Internet
2	Data Link	Network Interface
1	Physical	(Hardware)
	OSI Model	TCP/IP Model

Źródło: http://www.tcpipguide.com

Innovations ? In the application layer!

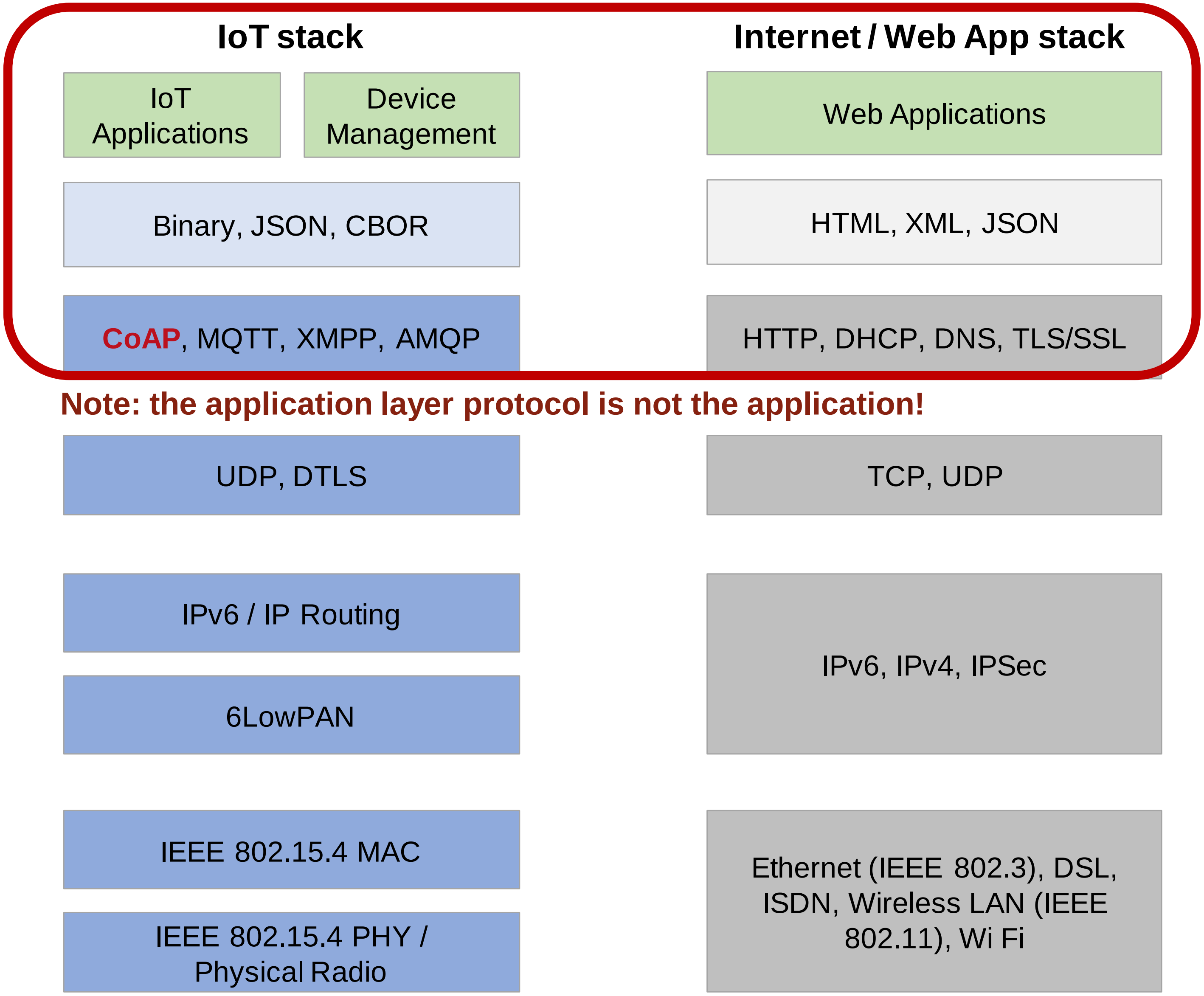
APPLICATION LAYER (2/3)



Source: https://www.micrium.com/iot/internet-protocols

APPLICATION LAYER (3/3)

Source:
Constrained Application Protocol (Web Protocol for IoT)
A. Chakrabarti, www.slideshare.net



IoT stack

- IoT Applications | Device Management
- Binary, JSON, CBOR
- **CoAP**, MQTT, XMPP, AMQP
- UDP, DTLS
- IPv6 / IP Routing
- 6LowPAN
- IEEE 802.15.4 MAC
- IEEE 802.15.4 PHY / Physical Radio

Internet / Web App stack

- Web Applications
- HTML, XML, JSON
- HTTP, DHCP, DNS, TLS/SSL
- TCP, UDP
- IPv6, IPv4, IPSec
- Ethernet (IEEE 802.3), DSL, ISDN, Wireless LAN (IEEE 802.11), Wi Fi

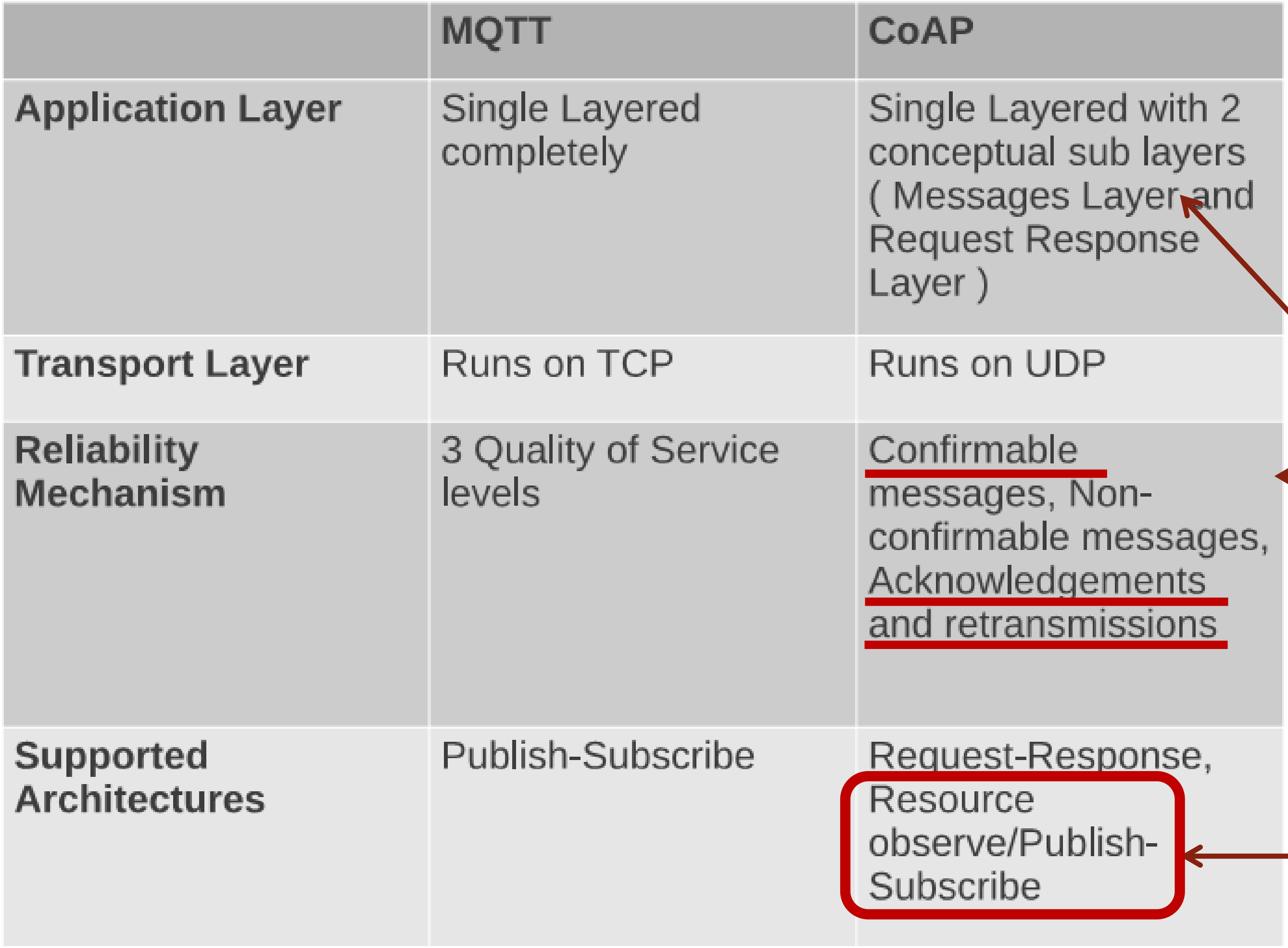
Note: the application layer protocol is not the application!

APPLICATION LAYER COMPETITORS (1/2)

- CoAP (Constrained Application Protocol)
  - developed by CoRE,Constrained RESTful Environments WG of IETF
  - an Internet (IETF) standard
  - runs on top of UDP
  - enables HTTP-like interactions in IoT: client/server, restful APIs

- MQTT (formerly Message Queuing Telemetry Transport, now MQTT)
  - developed by industry (IBM, Arcom)
  - supported by a major IBM product (MQ series)
  - now an OASIS standard and ISO standard
  - runs on top of TCP
  - based on the publish/subscribe interaction paradigm

APPLICATION LAYER COMPETITORS (2/2)



	MQTT	CoAP
Application Layer	Single Layered completely	Single Layered with 2 conceptual sub layers (Messages Layer and Request Response Layer)
Transport Layer	Runs on TCP	Runs on UDP
Reliability Mechanism	3 Quality of Service levels	Confirmable messages, Non-confirmable messages, Acknowledgements and retransmissions
Supported Architectures	Publish-Subscribe	Request-Response, Resource observe/Publish-Subscribe

Message Layer (reliability)

Observe option

Source: *Performance Evaluation of MQTT and CoAP via a Common Middleware*,
D. Thangavel et al., 2014 IEEE Ninth Intl. Conf. on Intelligent Sensors, Sensor Networks and Information Processing, 2014

CoRE (Constrained Restful E.) *Constrained?*

Name	data size (e.g., RAM)	code size (e.g., Flash)
Class 0, C0	<< 10 KiB	<< 100 KiB
Class 1, C1	~ 10 KiB	~ 100 KiB
Class 2, C2	~ 50 KiB	~ 250 KiB

Table 1: Classes of Constrained Devices (KiB = 1024 bytes) [RFC7228]

Source: *Terminology for Constrained-Node Networks*, RFC7228
C. Bormann, M. Ersue, A. Keranen , May 2014

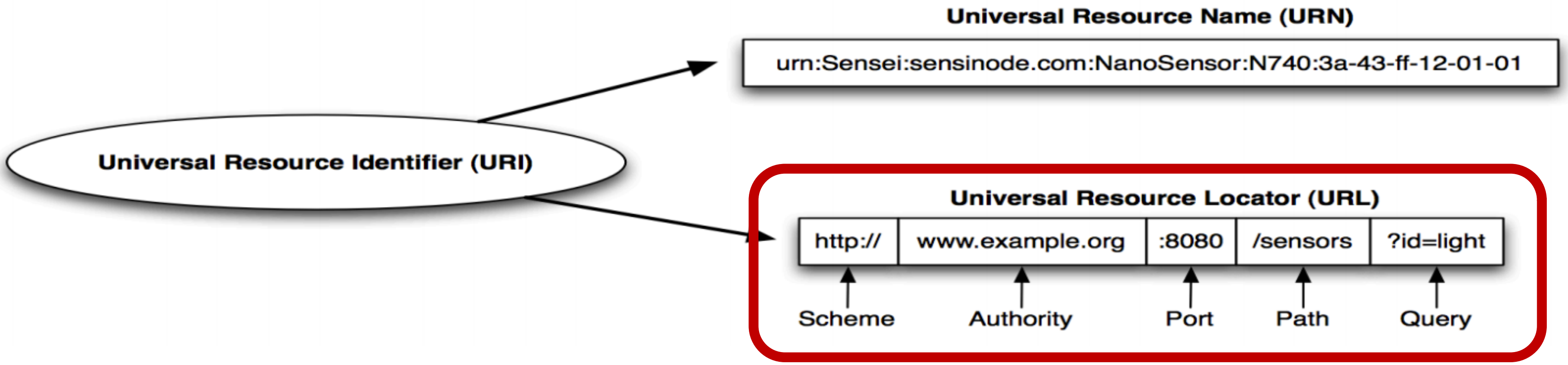
CoRE (Constrained RESTful E.) *RESTful?*

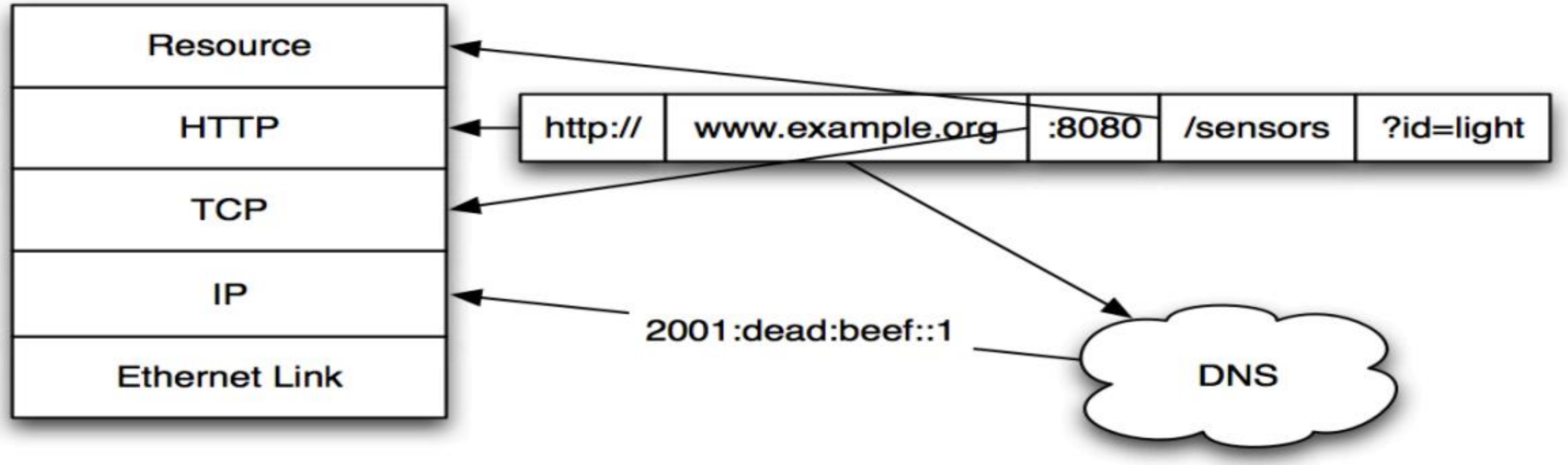
- note: the description below is (somewhat) simplified

- there are resources (e.g., data items, sensor readings, …, whatever)
- a resource has its URI
- a resource is hosted on a server
- a resource has its (possibly multiple) representation(s)
  - a resource representation has its media type
- the client uses the CRUD " verbs" (Create, Retrieve, Update, Delete) to transfer (work with) resource representations
  - these verbs are resource and application neutral
- no per-client state on the server (statelessness)
  - a request from a client must be understood by itself
  - the state is kept only on clients

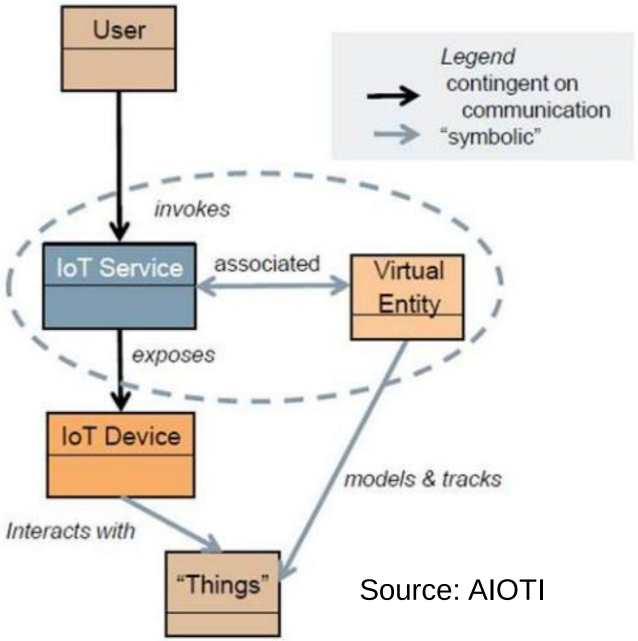
URIs

Source: *CoAP: The Web of Things Protocol*, ARM IoT Tutorial, Z. Shelby, 2014


Universal Resource Name (URN)
urn:Sensei:sensinode.com:NanoSensor:N740:3a-43-ff-12-01-01
Universal Resource Identifier (URI)
Universal Resource Locator (URL)
http://
www.example.org
:8080
/sensors
?id=light
Scheme
Authority
Port
Path
Query



CoAP – fundamentals



Source: AIOTI

CoAP: KEY RFCs

- [RFC7252] "The Constrained Application Protocol (CoAP)"
  - Z. Shelby, K. Hartke, C. Bormann, June 2014

  the main CoAP specification, 112 pages

- [RFC7641] "Observing Resources in CoAP"
  - K. Hartke, September 2015

  how to be up to date about the state of a resource without too many requests

- [RFC7959] "Blockwise Transfers in CoAP"
  - C. Bormann, Z. Shelby, August 2016

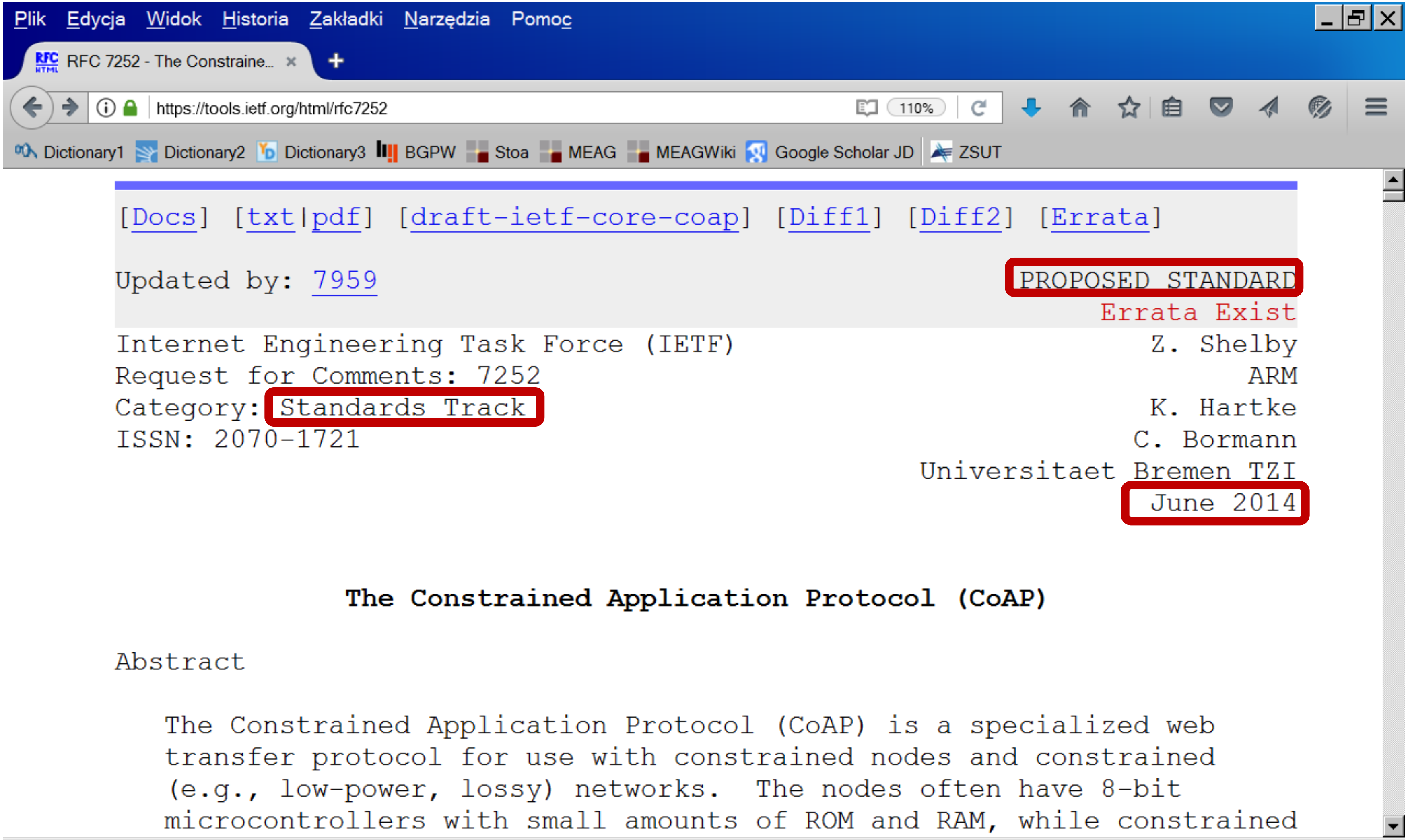
  how to transfer big resource representations

- [RFC6690] "Constrained RESTful Environments (CoRE) Link Format"
  - Z. Shelby, August 2012

  how to discover resources hosted by a server

RFC 7252 (1/2)

- Document: https://tools.ietf.org/html/rfc7252



[Docs] [txt|pdf] [draft-ietf-core-coap] [Diff1] [Diff2] [Errata]

```
Updated by: 7959                                   PROPOSED STANDARD
                                                       Errata Exist
Internet Engineering Task Force (IETF)                    Z. Shelby
Request for Comments: 7252                                      ARM
Category: Standards Track                                 K. Hartke
ISSN: 2070-1721                                          C. Bormann
                                               Universitaet Bremen TZI
                                                           June 2014
```

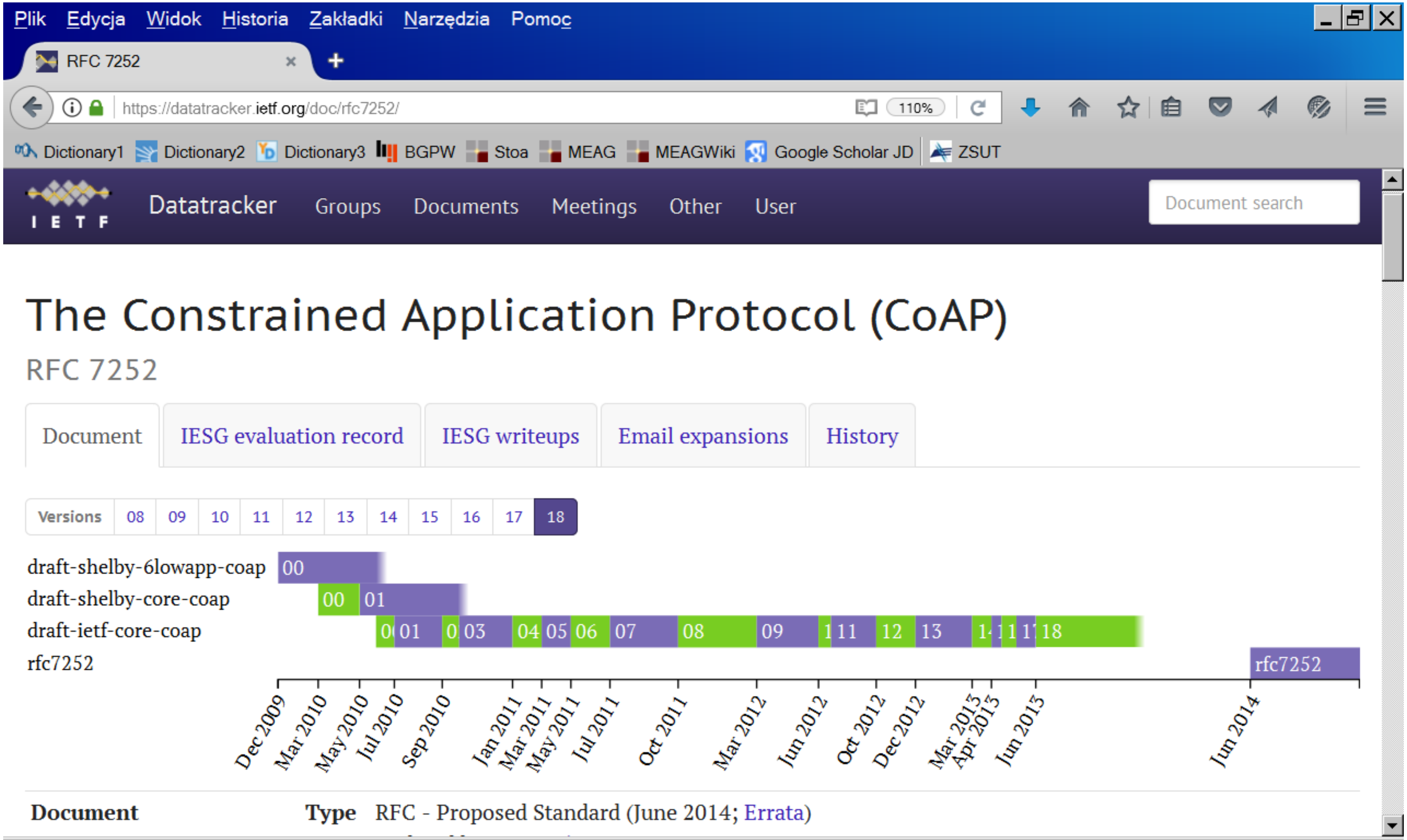
The Constrained Application Protocol (CoAP)

Abstract

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

RFC 7252 (2/2)

- History: https://datatracker.ietf.org/doc/rfc7252/

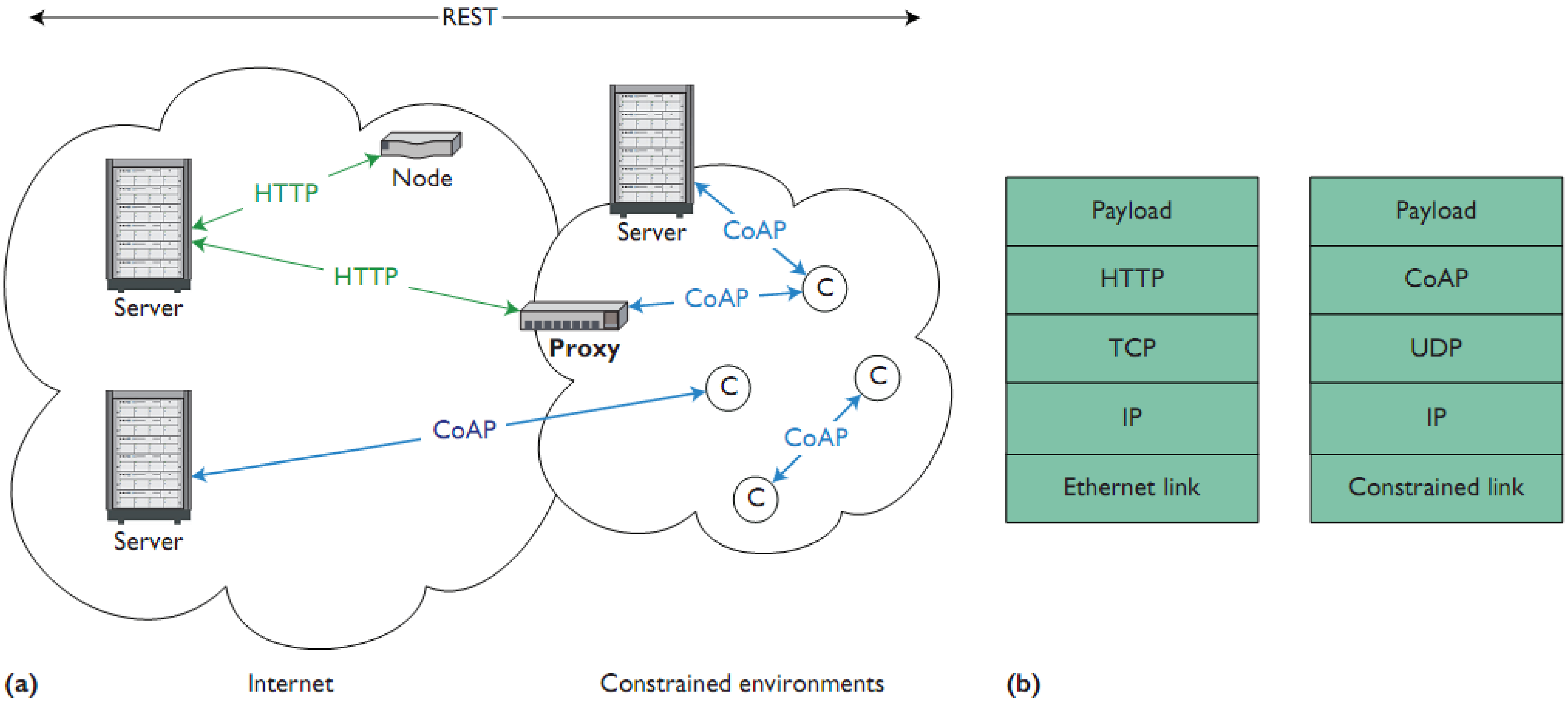


The Constrained Application Protocol (CoAP)

RFC 7252

Document **Type** RFC - Proposed Standard (June 2014; Errata)

CoAP system architecture

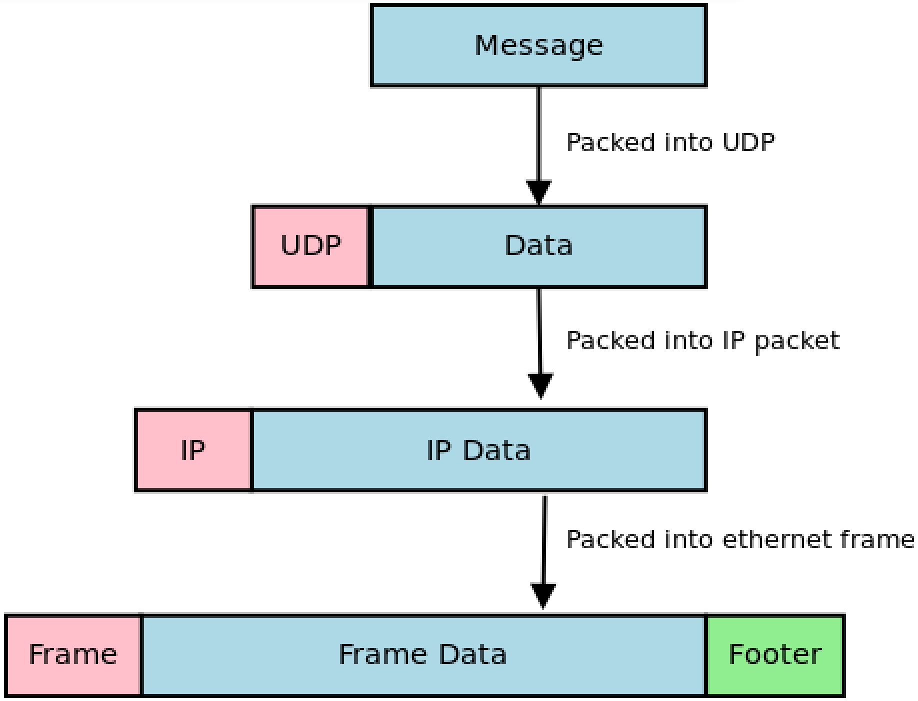


Payload	Payload
HTTP	CoAP
TCP	UDP
IP	IP
Ethernet link	Constrained link

Source: *CoAP: An Application Protocol for Billions of Tiny Internet Nodes*
C. Bormann, A. P. Castellani, Z. Shelby
IEEE INTERNET COMPUTING, 2012

UDP BASICS

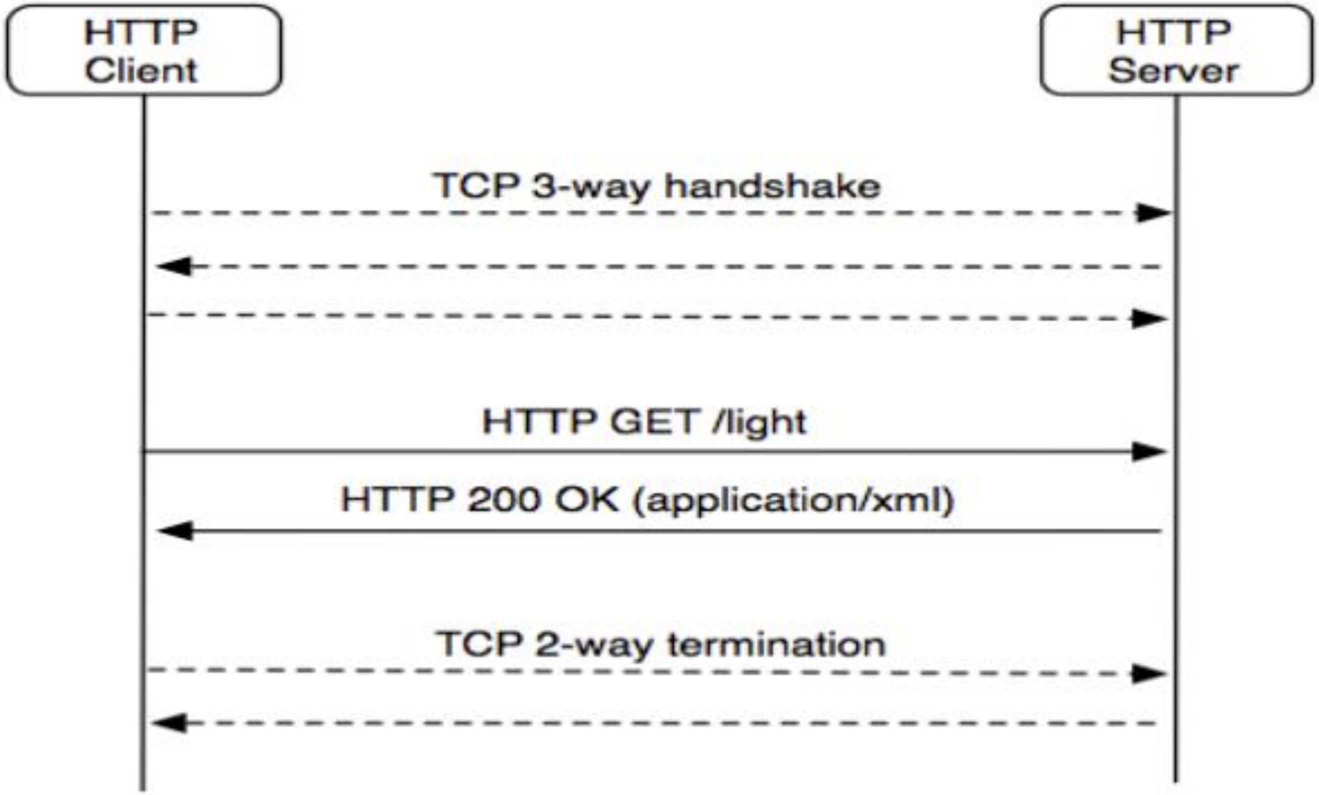
```
 0      7 8     15 16    23 24    31
+--------+--------+--------+--------+
|     Source      |   Destination   |
|      Port       |      Port       |
+--------+--------+--------+--------+
|                                   |
|     Length      |    Checksum     |
+--------+--------+--------+--------+
|
|          data octets ...
+---------------- ...

 User Datagram Header Format [RFC768]
```



http://jamesslocum.com/post/77759061182

- connectionless
- each user datagram results in a single IP datagram
- delivery: out-of-order, duplicated, missing
- offers the port abstraction
- aside: why would anybody want to use UDP?

Why not TCP?



Source: *CoAP: The Web of Things Protocol*, ARM IoT Tutorial, Z. Shelby, 2014

CoAP in the Protocol Stack

```
                              +----------------------+
                              |      Application     |
                              +----------------------+
                              +----------------------+  \
RESTful interaction           |  Requests/Responses  |  |
                              |----------------------|  | CoAP
lightweight  reliability      |       Messages       |  |
                              +----------------------+  /
                              +----------------------+
                              |          UDP         |
                              +----------------------+
```

Figure 1: Abstract Layering of CoAP [RFC7252]

- CoAP endpoint = IP address + UDP port number, port 5683
- each CoAP message occupies the data section of one UDP datagram
- CoAP over TCP (RFC 8323) is also possible
- CoAP is not the application itself (the application logic is up to you!)

CoAP MESSAGES

- CoAP client and server (one node may play both roles)

- requests/responses:
  - requests: from client to server
    method code (which action to perform on the resource): GET, PUT, POST, DELETE

  - responses: from server to client
    response code (similar to the HTTP status code)

- CON (confirmable)/NON (non-confirmable)/ACK/RST
  - CON+ACK: lightweight reliability
  - RST: recipient unable to process the message

CoAP MESSAGE FORMAT

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Message Format[RFC7252]

shortest CoAP message: 4B

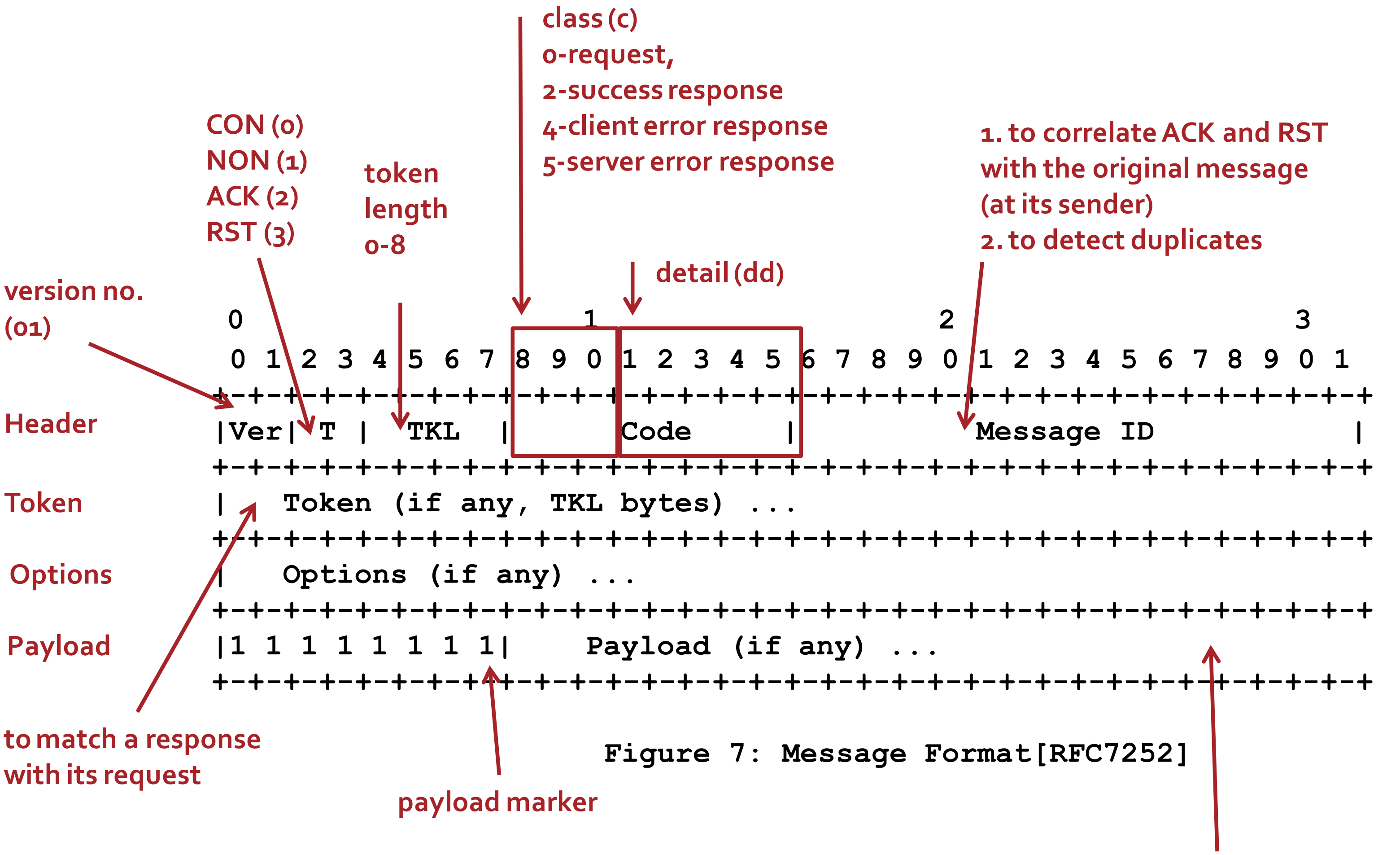
CoAP MESSAGE FORMAT

Consider a GET on a resource. Here is what different fields are for.

```
 GET (in the request) and Content (in the response)    to match an ACK with  the message
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Ver| T |  TKL  |      Code       |          Message ID           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Token (if any, TKL bytes) ...  to match the response with the request
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Options (if any) ...  URI (in the request) goes here
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |1 1 1 1 1 1 1 1|    Payload (if any) ...  a resource representation (in the response)
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Message Format[RFC7252]

CoAP MESSAGE FORMAT



class (c)
0-request,
2-success response
4-client error response
5-server error response

CON (0)
NON (1)
ACK (2)
RST (3)

token
length
0-8

1. to correlate ACK and RST with the original message (at its sender)
2. to detect duplicates

detail (dd)

version no.
(01)

Header

Token

Options

Payload

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

to match a response with its request

Figure 7: Message Format[RFC7252]

payload marker

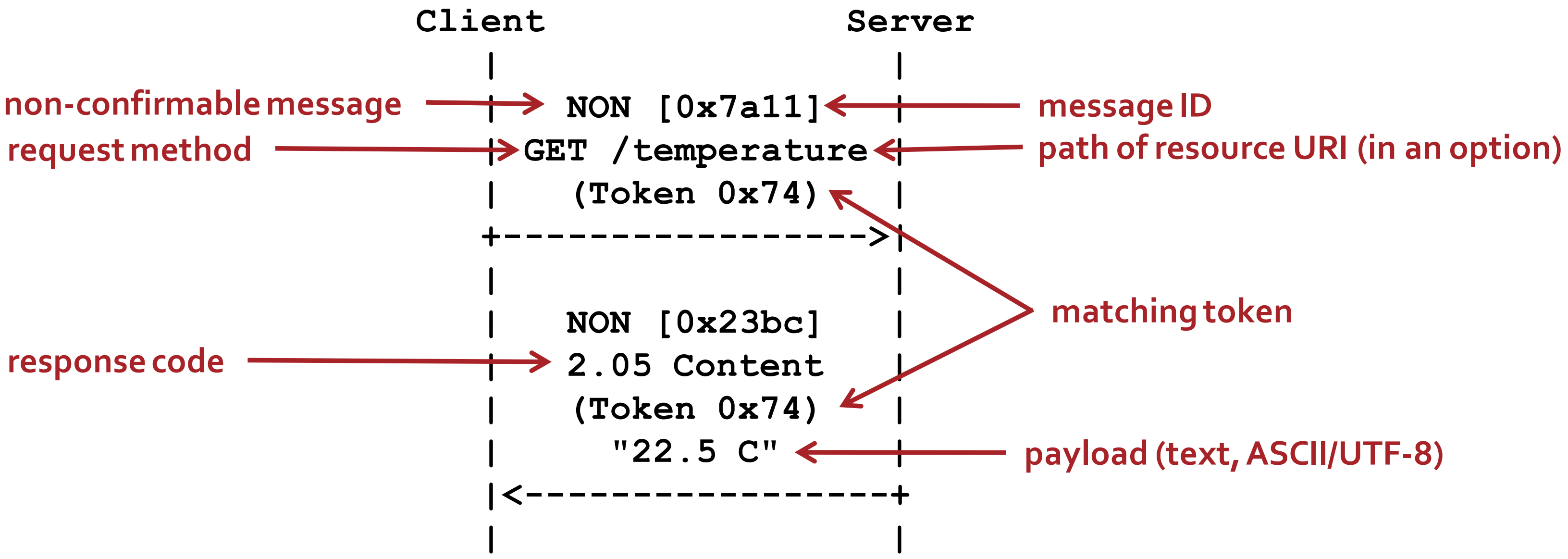
payload length calculated is from the UDP datagram size

c.dd indicates a Request Method or a Response Code

0.00 Empty message
0.01 GET
0.02 POST
0.03 PUT
0.04 DELETE

2.dd success
4.dd client error
5.dd server error

NO RELIABILITY: NON-CONFIRMABLE MESSAGES



```
        Client              Server
           |                  |
           |  NON [0x7a11]    |
           | GET /temperature |
           |   (Token 0x74)   |
           +----------------->|
           |                  |
           |  NON [0x23bc]    |
           |  2.05 Content    |
           |  (Token 0x74)    |
           |    "22.5 C"      |
           |<-----------------+
           |                  |
```

non-confirmable message

request method

response code

message ID

path of resource URI (in an option)

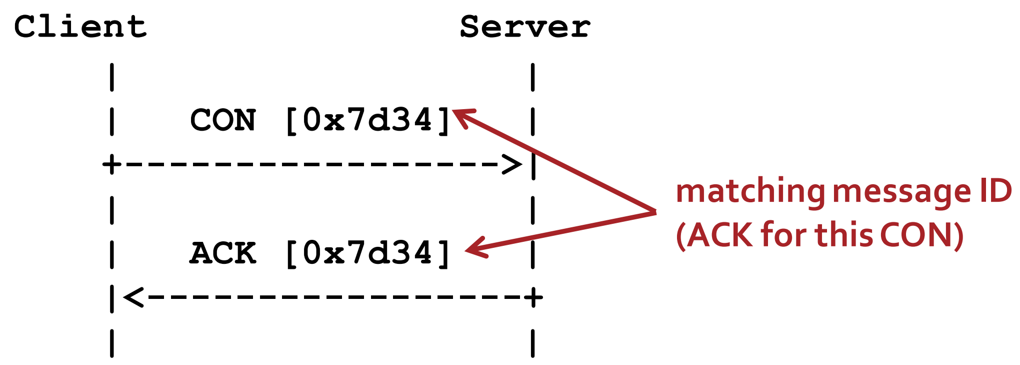
matching token

payload (text, ASCII/UTF-8)

Figure 6: A Request and a Response Carried in Non-confirmable Messages [RFC7252]

- reception not acknowledged
- the token is used to match a response with its request
- RST when the recipient unable to process a non-confirmable message

WITH RELIABILITY: CONFIRMABLE MESSAGES



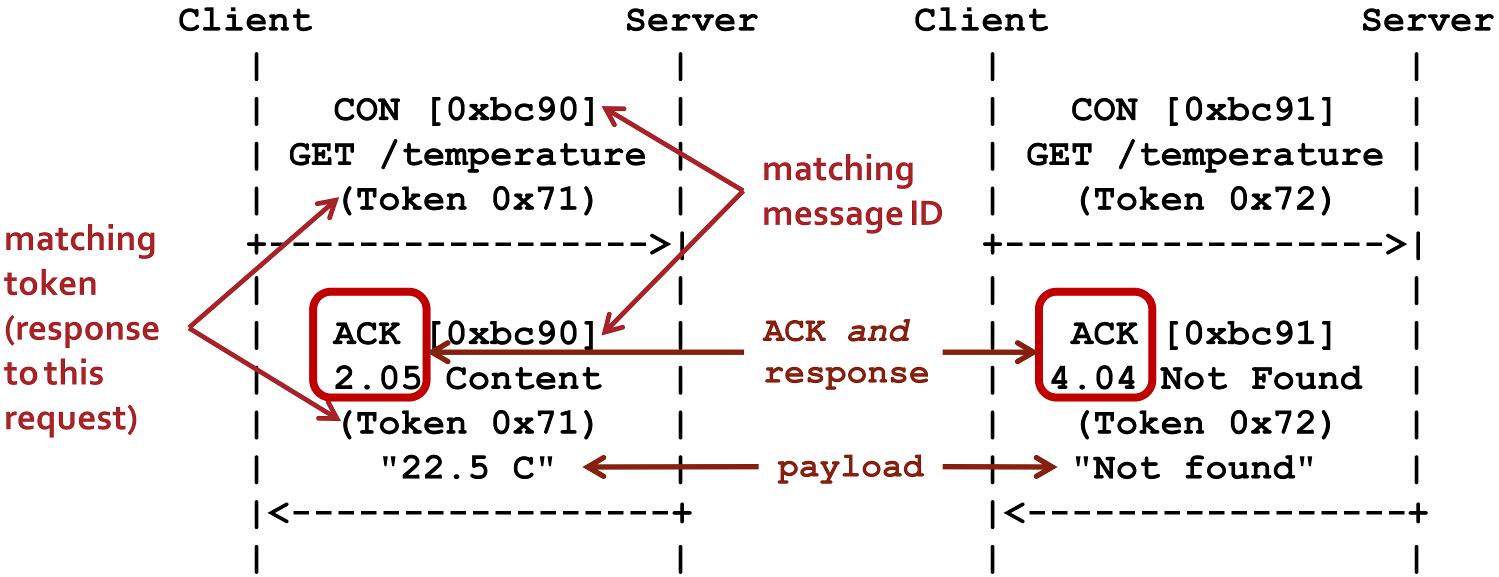
```
                     Client              Server
                        |                  |
                        |   CON [0x7d34]   |
                        +----------------->|
                        |                  |
                        |   ACK [0x7d34]   |
                        |<-----------------+
                        |                  |
```

matching message ID
(ACK for this CON)

Figure 2: Reliable Message Transmission [RFC7252]

- simple stop-and-wait
- wait for ACK (or RST) with timeout
- if no ACK, retransmit
- exponential back-off: timeout doubled each time
- continue until you run out of attempts (MAX_RETRANSMIT)
- RST when the recipient unable to process a confirmable message
- note: ACK (by itself) is <u>not</u> a response

PIGGYBACKED RESPONSE

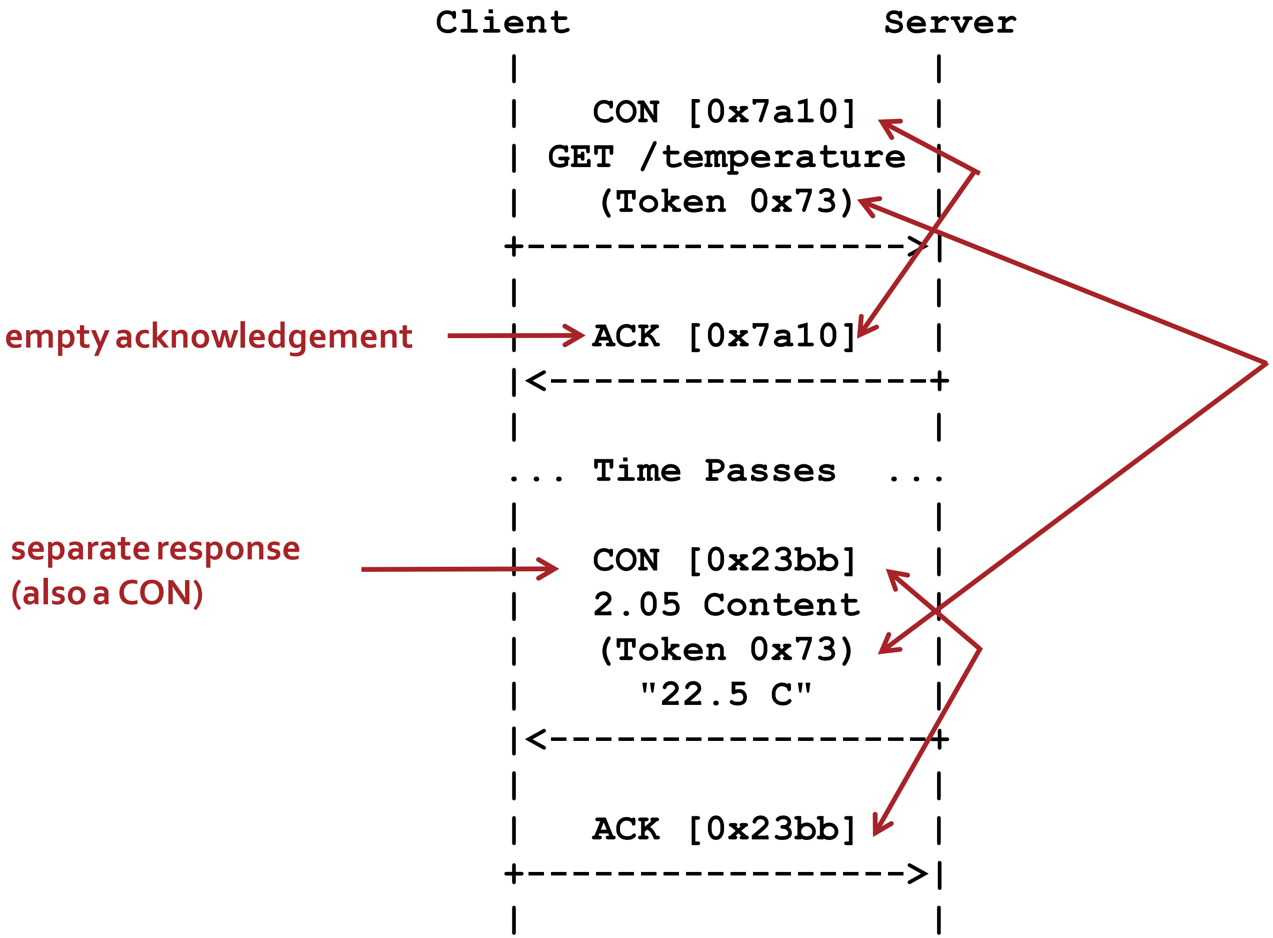


```
Client                Server  Client                Server
   |                     |       |                     |
   |   CON [0xbc90]      |       |   CON [0xbc91]      |
   | GET /temperature    |       | GET /temperature    |
   |   (Token 0x71)      |       |   (Token 0x72)      |
   +------------------->|       +------------------->|
   |                     |       |                     |
   |   ACK [0xbc90]      |       |   ACK [0xbc91]      |
   |   2.05 Content      |       |   4.04 Not Found    |
   |   (Token 0x71)      |       |   (Token 0x72)      |
   |     "22.5 C"        |       |   "Not found"       |
   |<-------------------+       |<-------------------+
   |                     |       |                     |
```

Figure 4: Two GET Requests with Piggybacked Responses [RFC7252]

- the response carried in ACK (if available immediately)

EMPTY ACK AND SEPARATE RESPONSE



```
  Client                Server
     |                    |
     |   CON [0x7a10]     |
     | GET /temperature   |
     |   (Token 0x73)     |
     +------------------->|
     |                    |
     |   ACK [0x7a10]     |
     |<-------------------+
     |                    |
     ... Time Passes  ...
     |                    |
     |   CON [0x23bb]     |
     |   2.05 Content     |
     |   (Token 0x73)     |
     |     "22.5 C"       |
     |<-------------------+
     |                    |
     |   ACK [0x23bb]     |
     +------------------->|
     |                    |
```

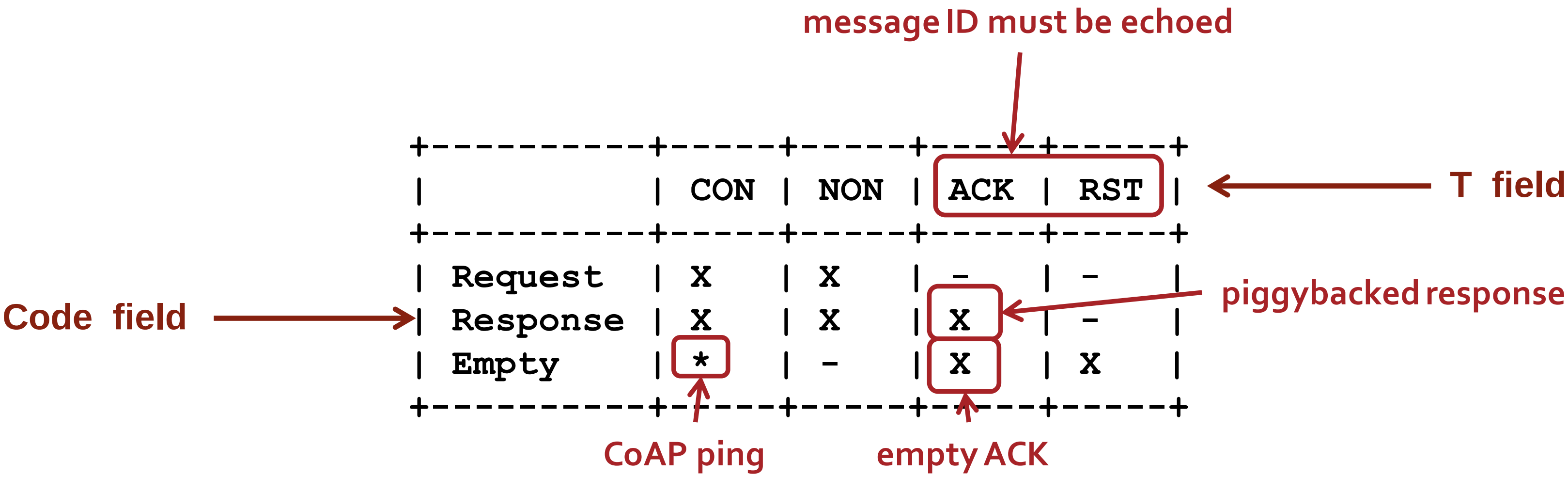
empty acknowledgement

separate response
(also a CON)

Figure 5: A GET Request with a Separate Response[RFC7252]

- if the response not available immediately (say, it takes some time to take a sensor reading)

USAGE OF MESSAGE TYPES



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

message ID must be echoed

T field

Code field

piggybacked response

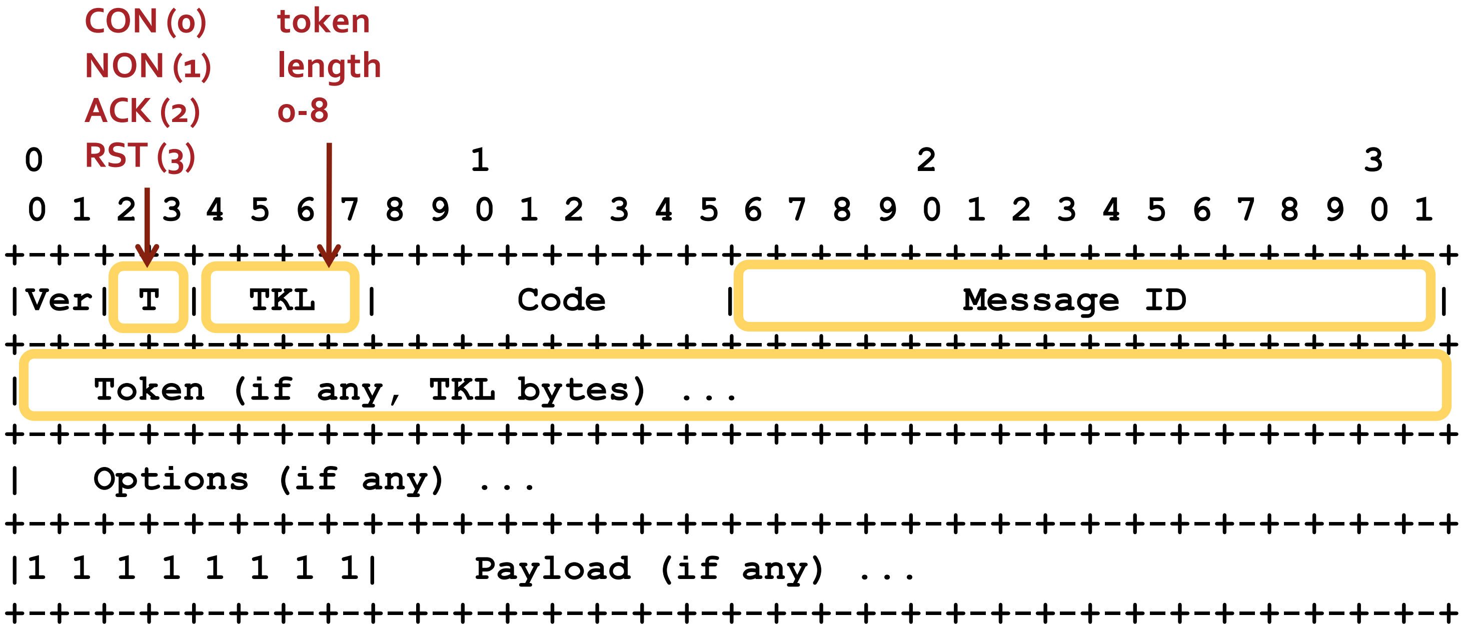
CoAP ping

empty ACK

Table 1: Usage of Message Types [RFC7252]

- CoAP ping: to elicit a reset message (RST), not in normal operation

CON, NON, ACK, RST, MESSAGE ID, TOKEN IN MESSAGE



CON (0)
NON (1)
ACK (2)
RST (3)

token
length
0-8

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Message Format[RFC7252]

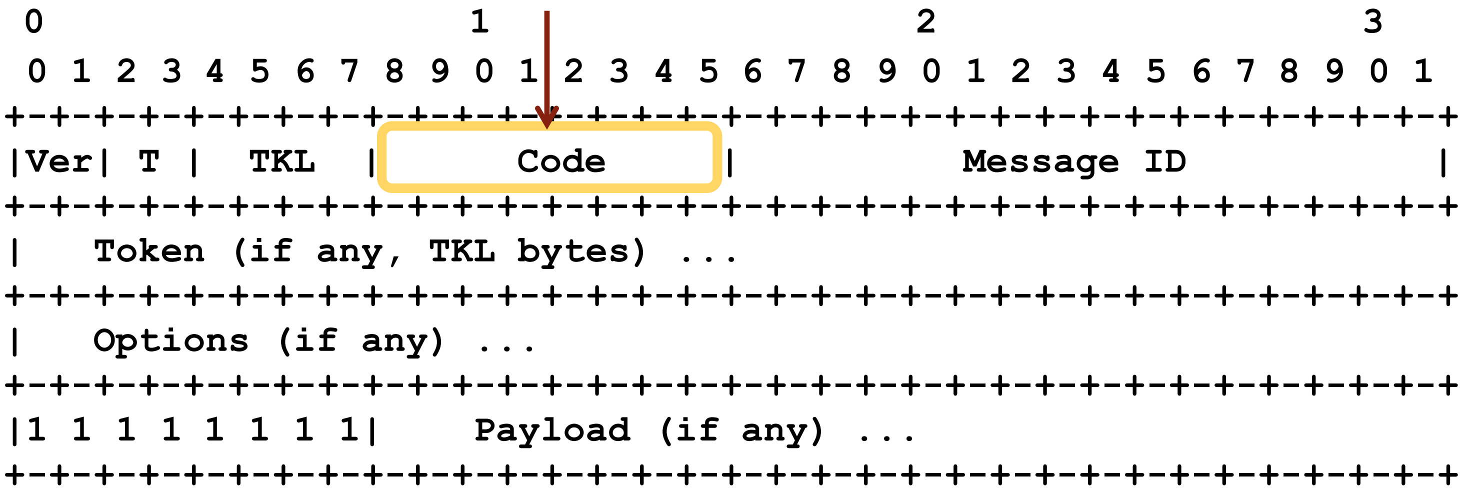
REQUEST METHODS

- GET, PUT, POST, and DELETE
- these are similar to those of HTTP
- an URI (partially given in options) identifies a resource

- GET: retrieves a representation of the identified resource
- POST: requests that the representation enclosed in the request be processed
  - the actual function performed by the POST method is determined by the server and dependent on the target resource
  - it usually results in a new resource being created or the target resource being updated (the target resource may also be deleted)
- PUT: requests that the identified resource be updated or created with the enclosed representation
- DELETE: requests that the identified resource be deleted

METHOD CODES IN MESSAGE

0.01 GET
0.02 POST
0.03
0.04 DELETE



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Message Format[RFC7252]

RESPONSES

- a response is matched to the request by means of a client-generated token

- three classes of Response Codes: **kody odpowiedzi**
  - 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

RESPONSE CODES IN MESSAGE: SUCCESS 2.XX

2.01	Created	POST and PUT
2.02	Deleted	DELETE and POST
2.03	Valid	the response identified by the entity-tag is valid (used in validation for caching purposes)
2.04	Changed	PUT and POST
2.05	Content	GET
2.31	Continue	in block-wise transfers; a block has been received successfully, but the total update has not been completed yet

```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|class|  detail |
+-+-+-+-+-+-+-+-+
```

c.dd

2.05 ↔ binary: 010.00101 ↔ decimal: 64+4+1=69

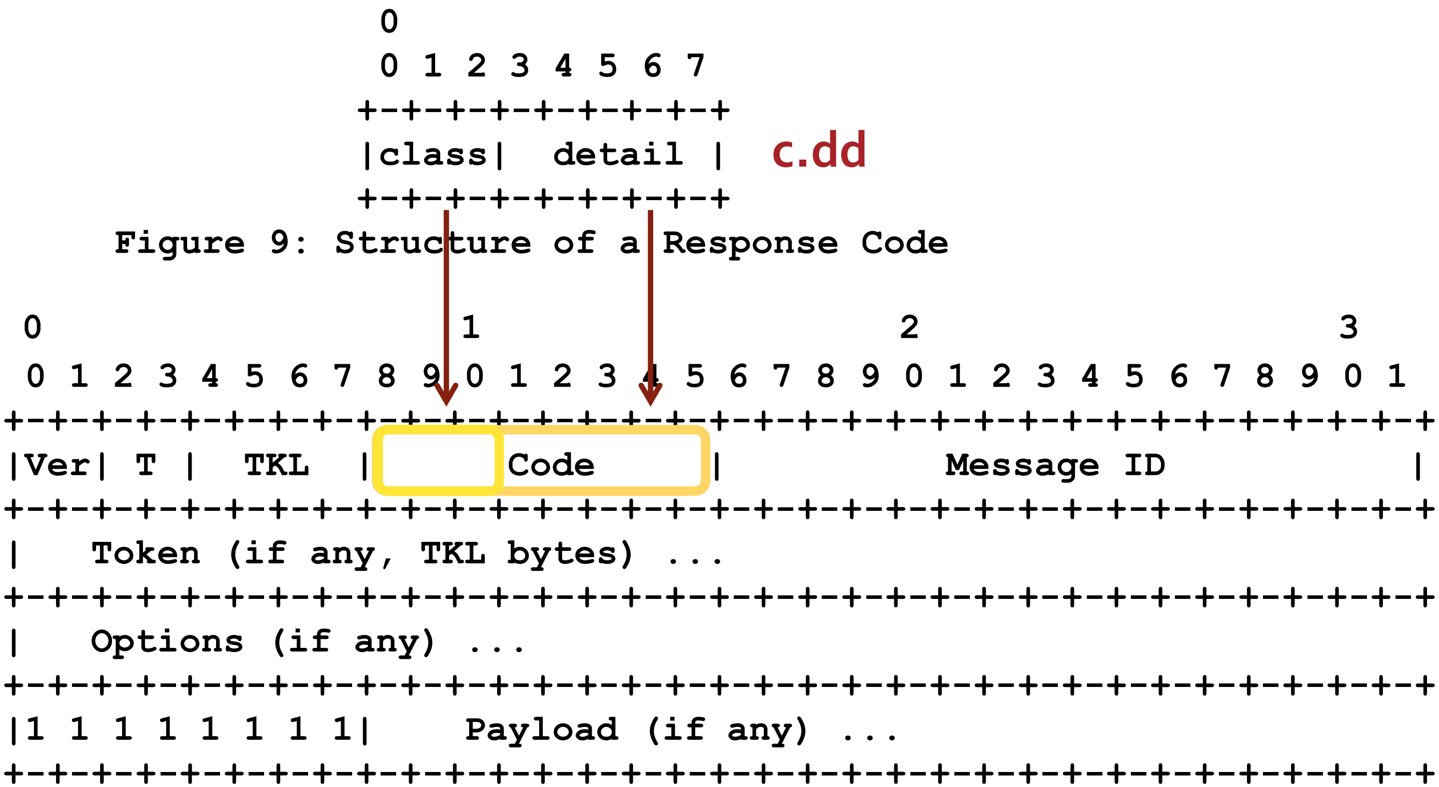
Figure 9: Structure of a Response Code

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Message Format[RFC7252]

RESPONSE CODES IN MESSAGE: CLIENT ERROR 4.XX

4.00	Bad Request (generic response code)
4.01	Unauthorized
4.02	Bad Option
4.04	Not Found
4.05	Method Not Allowed
4.15	Unsupported Content-Format
…	…



```
  0
  0 1 2 3 4 5 6 7
 +-+-+-+-+-+-+-+-+
 |class|  detail |
 +-+-+-+-+-+-+-+-+
```

c.dd

Figure 9: Structure of a Response Code

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Message Format[RFC7252]

RESPONSE CODES IN MESSAGE: SERVER ERROR 5.XX

5.00	Internal Server Error (generic response code)
5.01	Not Implemented
5.03	Service Unavailable (uses the Max-Age Option to indicate the number of seconds after which to retry
…	…

```
  0
  0 1 2 3 4 5 6 7
 +-+-+-+-+-+-+-+-+
 |class|  detail |
 +-+-+-+-+-+-+-+-+
```

c.dd

```
Figure 9: Structure of a Response Code
```

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
Figure 7: Message Format[RFC7252]
```

CoAP OPTIONS

option = (option number, option value)

option number

… of option value

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

Table 4: Options [RFC7252]

SELECTED OPTIONS (1/2)

- **`Content-Format`**
  - the representation format of the payload
- **`Etag`**
  - an entity-tag is intended for use as a resource-local identifier for a specific representation of a resource; generated by the server providing the resource; used for validation
- **`Max-Age`**
  - the maximum time a response may be cached before it is considered not fresh, default: 60s
- **`Accept`**
  - in a request, the client can indicate which content-format it prefers to receive

SELECTED OPTIONS (2/2)

coap-URI = "coap:" "//" host [":" port] path ["?" query]

- **`Uri-Host`**
  - default: the IP address of the request message
- **`Uri-Path`**
- **`Uri-Port`**
  - default: the destination UDP port
- **`Uri-Query`**

OPTIONS IN MESSAGE

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...  ...       ...         ...       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

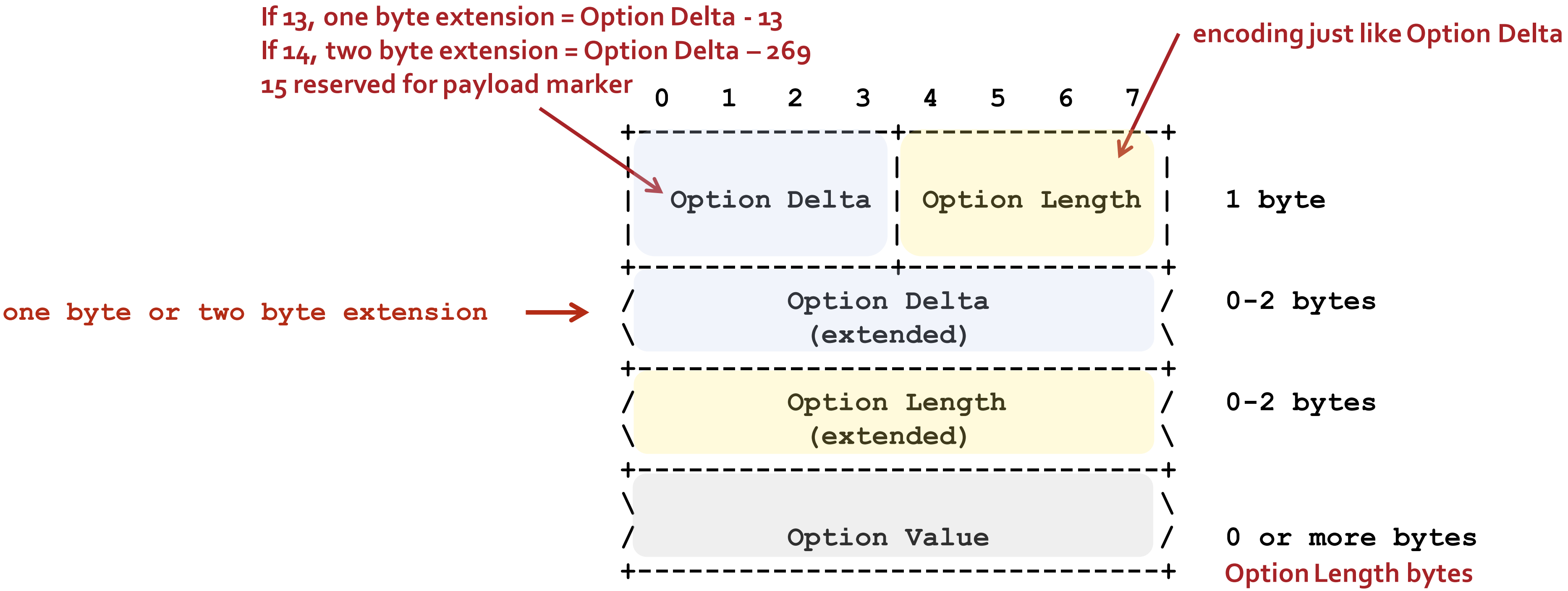
Figure 7: Message Format[RFC7252]

one option:

```
  0   1   2   3   4   5   6   7
+---------------+---------------+
|               |               |
|  Option Delta | Option Length |   1 byte
|               |               |
+---------------+---------------+
/         Option Delta          /   0-2 bytes
\          (extended)           \
+-------------------------------+
/         Option Length         /   0-2 bytes
\          (extended)           \
+-------------------------------+
\                               \
/         Option Value          /   0 or more bytes
+-------------------------------+
```

Figure 8: Option Format RFC[7252]

OPTION FORMAT: NUMBER + VALUE



If 13, one byte extension = Option Delta - 13
If 14, two byte extension = Option Delta – 269
15 reserved for payload marker

encoding just like Option Delta

one byte or two byte extension

```
  0   1   2   3   4   5   6   7
+---------------+---------------+
|               |               |
|  Option Delta | Option Length |   1 byte
|               |               |
+---------------+---------------+
/         Option Delta          /   0-2 bytes
\          (extended)           \
+-------------------------------+
/         Option Length         /   0-2 bytes
\          (extended)           \
+-------------------------------+
\                               \
/         Option Value          /   0 or more bytes
+-------------------------------+
```

Option Length bytes

Figure 8: Option Format RFC[7252]

- each option has a number
- a message may contain a sequence of options
- options are ordered according to their numbers (increasing order)
- Option Delta = no. of the current option – no. of the previous one
  - for the first option, Option Delta = no of the first option

OPTION FORMAT: DECODING OPTION DELTA

- let D = Option Delta (to be determined when parsing a message)

- let d = the Option Delta field in the first byte of the option
- let e0 = the first byte of the Option Delta extended (if present)
- let e1 = the second byte of the Option Delta extended (if present)

- if d <= 12
  - D=d, e0 missing, e1 missing
- if d == 13
  - D=13+e0, e1 missing
    (so 13 <= D <= 268)
- if d == 14
  - D=269+e0*256+e1
    (so D >= 269)

```
  0     1     2     3     4     5     6     7
+---------------+---------------+
|               |               |
|Option Delta d | Option Length |    1
|               |               |    byte
+---------------+---------------+
/   Option Delta (extended) e0  /    0-2
+-------------------------------+    bytes
\   Option Delta (extended) e1  \
+-------------------------------+

/               …               /
+-------------------------------+
```

network byte order: the first byte, e0, is more significant

Option format: encoding Option Delta

- let D = Option Delta (to be encoded when assembling a message)

- let d = the Option Delta field in the first byte of the option
- let e0 = the first byte of the Option Delta extended (if present)
- let e1 = the second byte of the Option Delta extended (if present)

- if D <= 12
  - d=D, e0 missing, e1 missing
- if 13 <= D <= 268
  - d=13, e0=D-13, e1 missing
- if D >= 269
  - d=14, e0=(D-269)/256, e1=(D-269)%256

```
   0   1   2   3   4   5   6   7
+---------------+---------------+
|               |               |
|Option Delta d | Option Length |   1
|               |               |   byte
+---------------+---------------+
/   Option Delta (extended) e0  /   0-2
+-------------------------------+   bytes
\   Option Delta (extended) e1  \
+-------------------------------+
/               ...              /
+-------------------------------+
```

OPTION FORMAT: OPTION DELTA EXAMPLES

D	d	e0	e1
7	7	-	-
13	13	0	-
17	13	4	-
268	13	255	-
269	14	0	0
270	14	0	1
524	14	0	255
525	14	1	0

```
  0   1   2   3   4   5   6   7
+---------------+---------------+
|               |               |
|Option Delta d | Option Length |   1
|               |               |   byte
+---------------+---------------+
/   Option Delta (extended) e0  /   0-2
+-------------------------------+   bytes
\   Option Delta (extended) e1  \
+-------------------------------+

/               ...             /
+-------------------------------+
```

OPTIONS: ASSORTED CAVEATS

- If the option value format is **`uint`**, leading zero bytes should not be included in the value.
  - in particular, if the option value is zero, the value should be empty (no bytes, Option Length equal to zero)

- ”each Uri-Path Option specifies one segment of the absolute path to the resource” RFC[7252]
  - one Uri-Path option per segment of the path (not the entire path)
  - for a path that includes multiple segments, a CoAP request will include multiple Uri-Path options (in the ”path order”)
  - the Option Delta for the second and following Uri-Path options is zero
  - we say that the Uri-Path option is **repeatable** (RFC[7262])

PAYLOAD

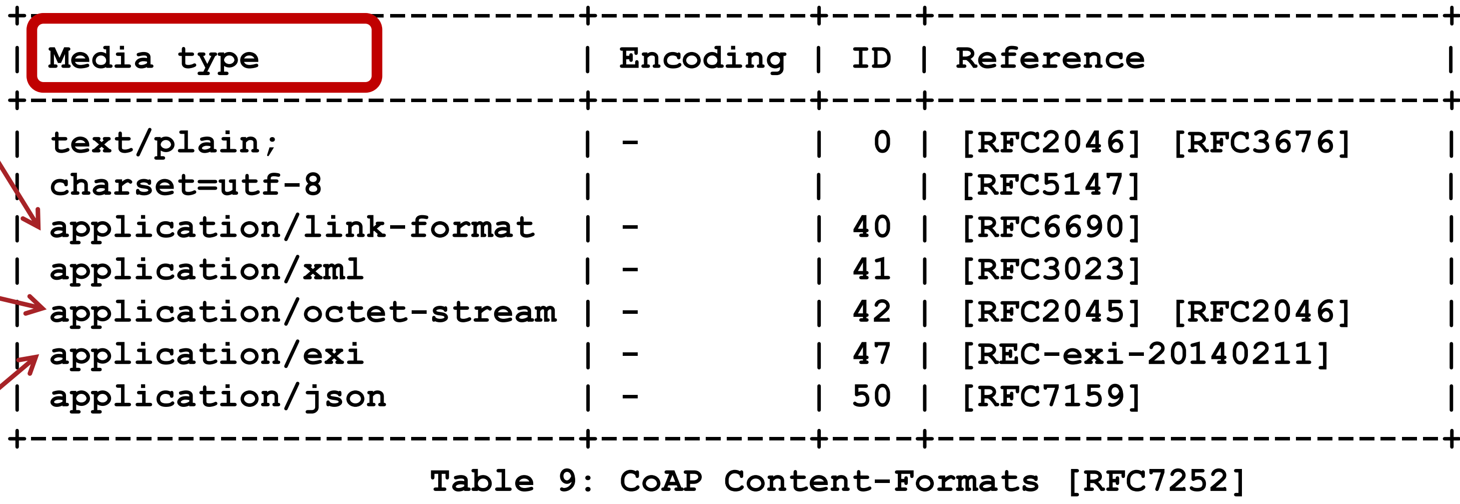
- possible payloads:
  - a resource representation
  - diagnostic payload (in case of error)

- resource representation
  - format is specified by the Internet media type given by the **`Content-Format`** Option

- diagnostic payload (when no **`Content-Format`** option is given)
  - the payload of responses indicating a client or server error is a brief human-readable diagnostic message, explaining the error situation

CONTENT FORMATS (CONTENT-FORMAT OPTION)

used for CoAP resource discovery



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 [RFC7252]

arbitrary binary data

Efficient XML Interchange (binary)

Concise Binary Object Representation →

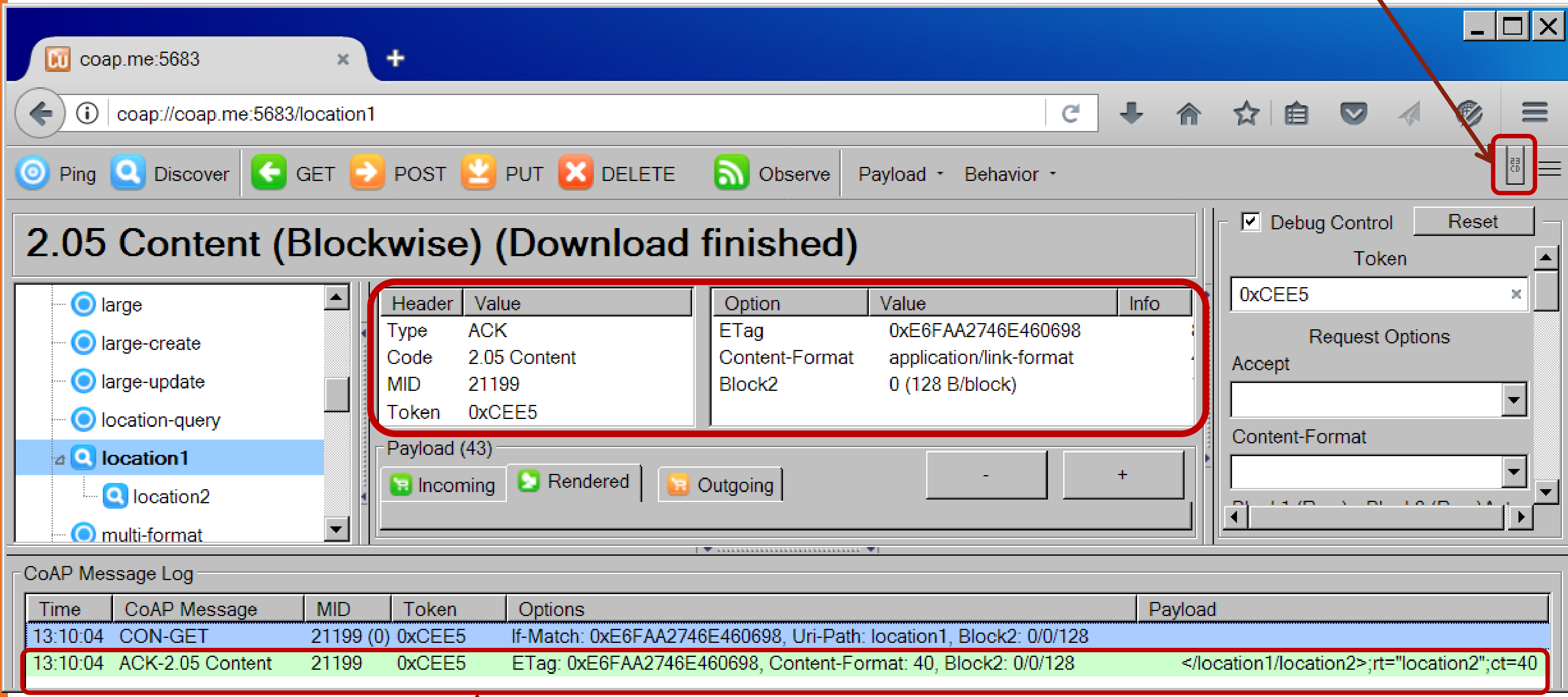
7.4. CoAP Content-Format
Media Type: application/cbor
Id: 60

Source: *Concise Binary ObjectRepresentation (CBOR)* , [RFC7049]

Parsing example: message

coap://coap.me:5683/location1

press here to see the log



2.05 Content (Blockwise) (Download finished)

Header	Value
Type	ACK
Code	2.05 Content
MID	21199
Token	0xCEE5

Option	Value	Info
ETag	0xE6FAA2746E460698	
Content-Format	application/link-format	
Block2	0 (128 B/block)	

☑ Debug Control   Reset

Token: 0xCEE5

Request Options

Accept:

Content-Format:

CoAP Message Log

Time	CoAP Message	MID	Token	Options	Payload
13:10:04	CON-GET	21199 (0)	0xCEE5	If-Match: 0xE6FAA2746E460698, Uri-Path: location1, Block2: 0/0/128	
13:10:04	ACK-2.05 Content	21199	0xCEE5	ETag: 0xE6FAA2746E460698, Content-Format: 40, Block2: 0/0/128	</location1/location2>;rt="location2";ct=40

this is the message we are going to parse (it's a piggybacked response)

PARSING EXAMPLE: WHAT LOG SAYS

```
UDP: Received 63 bytes
PACKET (hex):
62,45,52,CF,CE,E5,48,E6,FA,A2,74,6E,46,6,98,81,28,B1,3,FF,
3C,2F,6C,6F,63,61,74,69,6F,6E,31,2F,6C,6F,63,61,74,69,6F,6
E,32,3E,3B,72,74,3D,22,6C,6F,63,61,74,69,6F,6E,32,22,3B,63
,74,3D,34,30
PARSE: Token length = 2
PARSE: Token = 0xCEE5
PARSE: Option ETag = 230,250,162,116,110,70,6,152
PARSE: Option Content-Format = 40
PARSE: Option Block2 = 3
```

PARSING EXAMPLE: HEADER, TOKEN, PAYLOAD

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

MID=
5x16^3+ //5
2x16^2+ //2
12x16+ //C
15= //F
21199

UDP: Received 63 bytes

Ver=1 T=2 (ACK) TKL=2 response code= 2.05 (Content)

PACKET (hex):

4B+ 62,45,52,CF, **header** 01 10 0010, 010 0 0101, 0101 0010, 1100 1111

2B+ CE,E5, **token**

13B+ 48,E6,FA,A2,74,6E,46,6,98,81,28,B1,3, **options (next slide)**

1B+ FF, **payload marker**

43B 3C,2F,6C,6F,63,61,74,69,6F,6E,31,2F,6C,6F,63,61,
= 74,69,6F,6E,32,3E,3B,72,74,3D,22,6C,6F,63,61,74,
63B 69,6F,6E,32,22,3B,63,74,3D,34,30

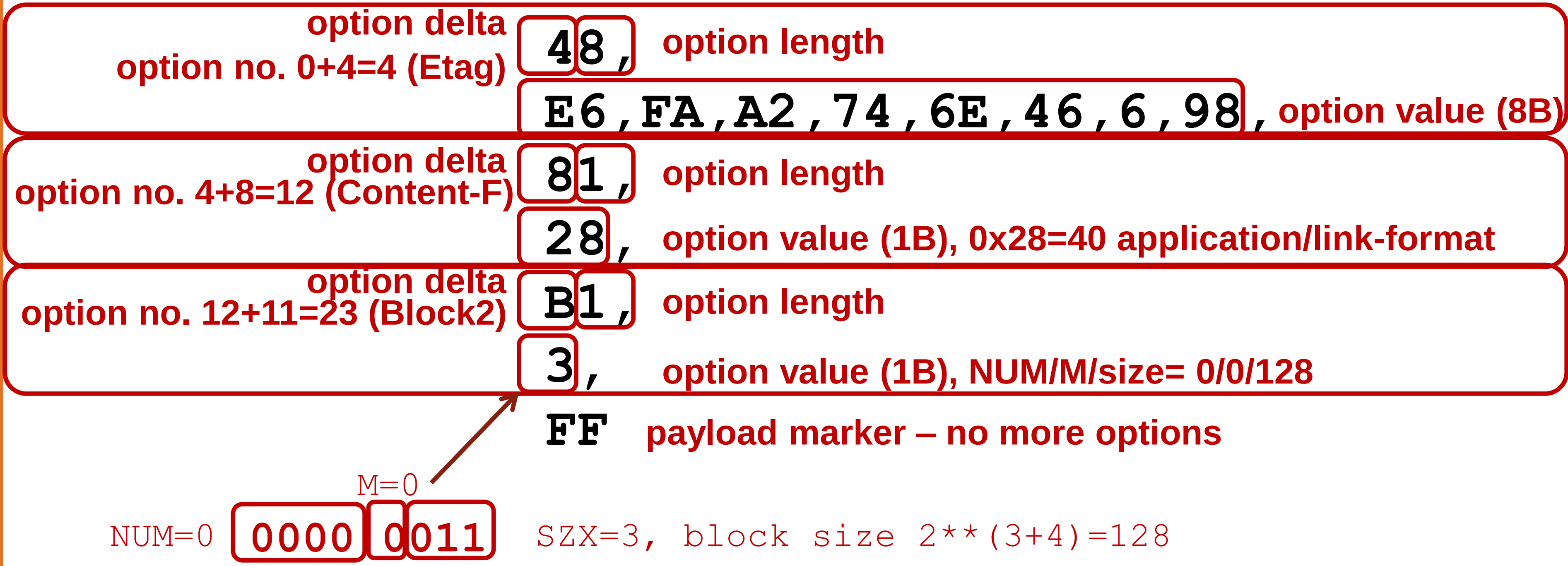
payload:
</location1/location2>;rt="location2";ct=40

0x30-ASCII ’0’

0x3C-ASCII ’<’

PARSING EXAMPLE: OPTIONS

No.	Name	Format	Length	Default
4	ETag	opaque	1-8	(none)
	. . .			
12	Content-Format	uint	0-2	(none)
	. . .			
23	Block2	uint	0-3	(none)



option delta

option no. 0+4=4 (Etag)

4 8, option length

E6,FA,A2,74,6E,46,6,98, option value (8B)

option delta

option no. 4+8=12 (Content-F)

8 1, option length

28, option value (1B), 0x28=40 application/link-format

option delta

option no. 12+11=23 (Block2)

B 1, option length

3, option value (1B), NUM/M/size= 0/0/128

FF payload marker – no more options

M=0

NUM=0 0000 0 011 SZX=3, block size 2**(3+4)=128

Note: the Block2 option is covered below.