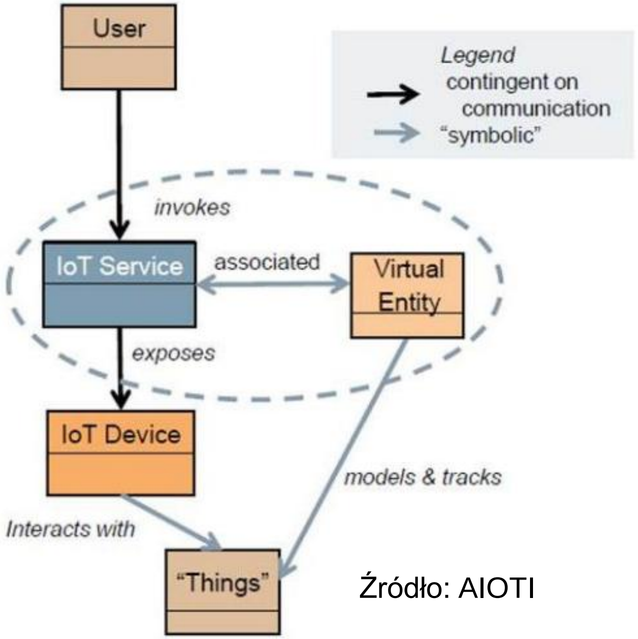
Obiekty Internetu Rzeczy

Wykład 3-4: Constrained Application Protocol (CoAP)

Jarosław Domaszewicz

Instytut Telekomunikacji Politechniki Warszawskiej

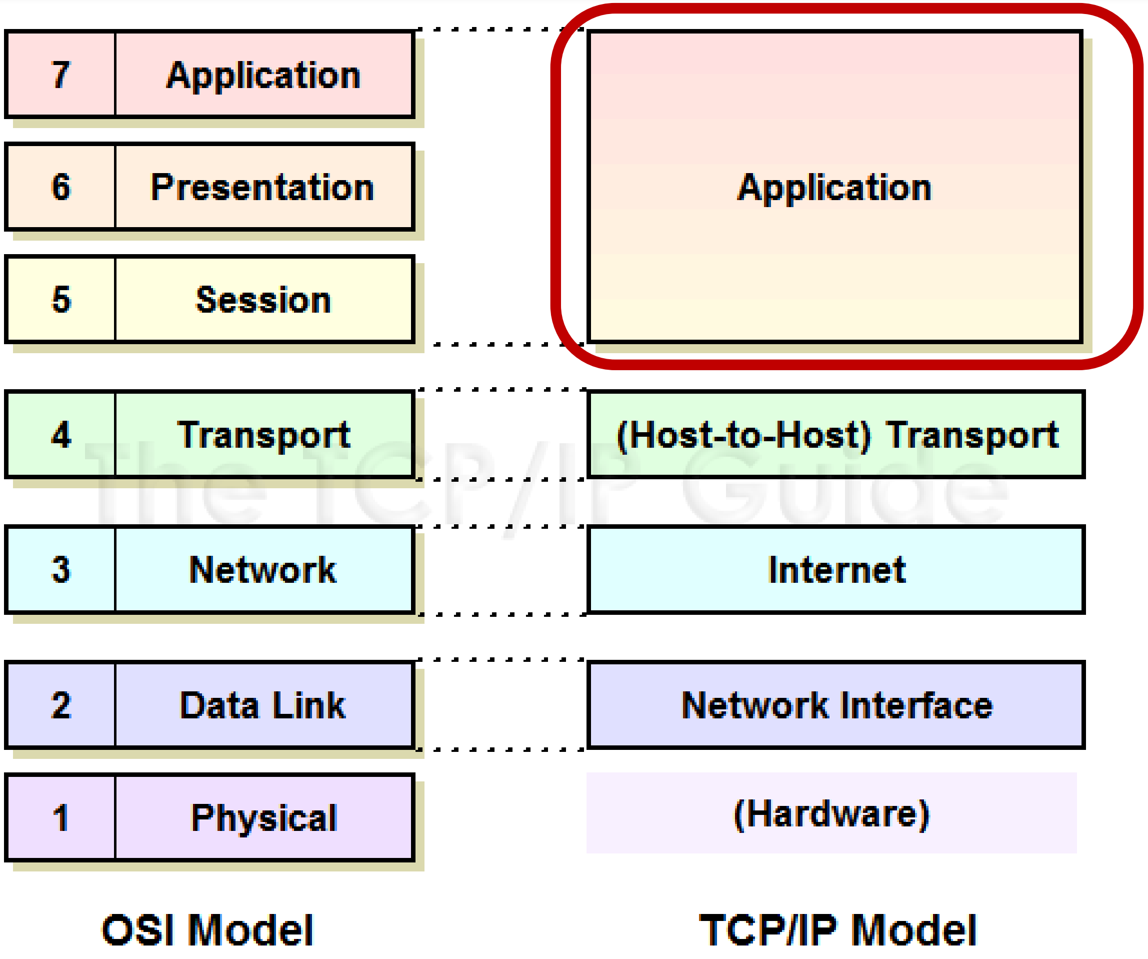


Źródło: AIOTI

Plan wykładu

- Wstęp
- Podstawy CoAPa
- Przykłady
- Cache i proxy
- Obserwowanie zasobów
- CoRE Link Format
- Transfery blokowe

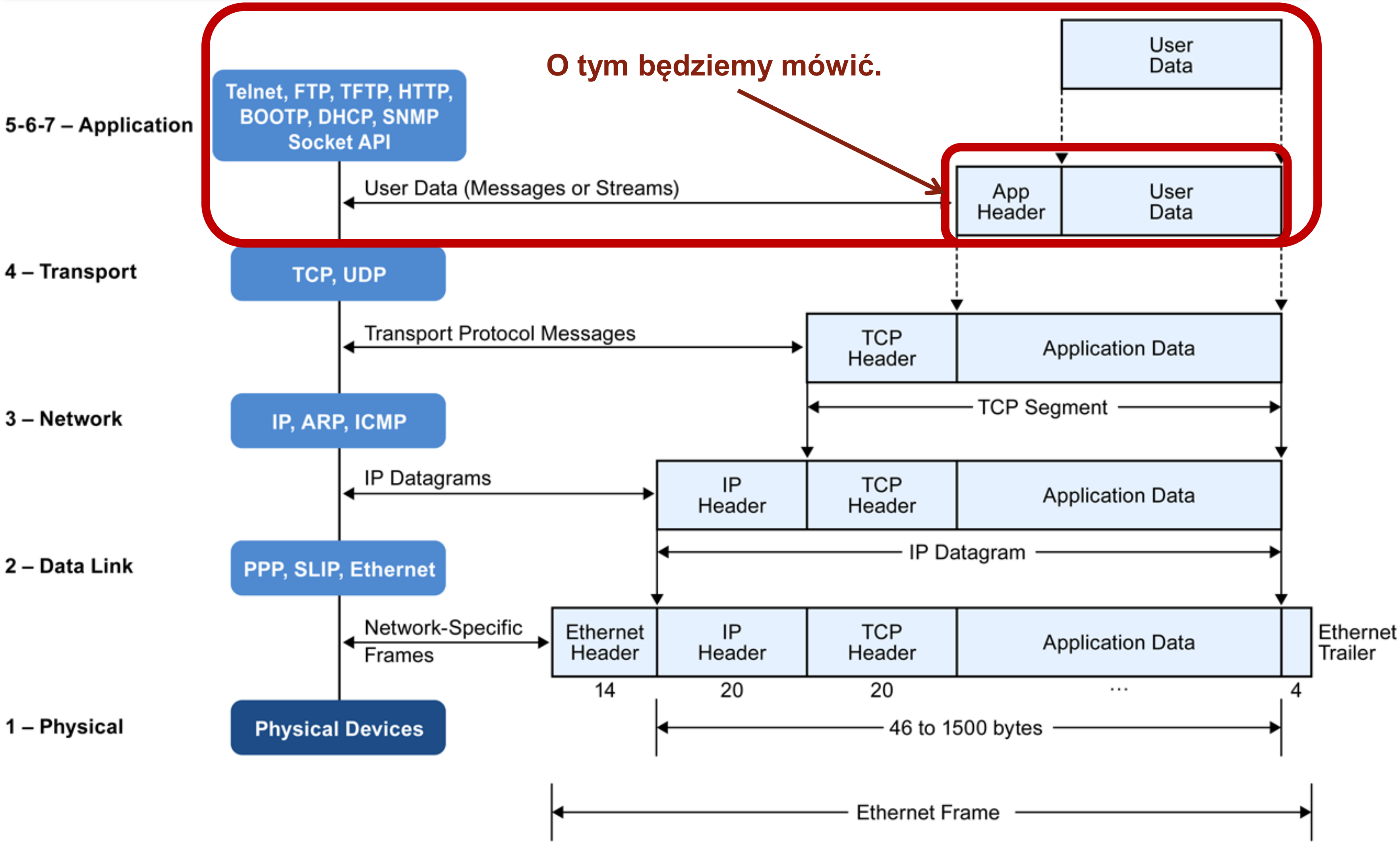
WARSTWA APLIKACJI (1/3)



Źródło: http://www.tcpipguide.com

Miejsce na innowacje ? W warstwie aplikacji!

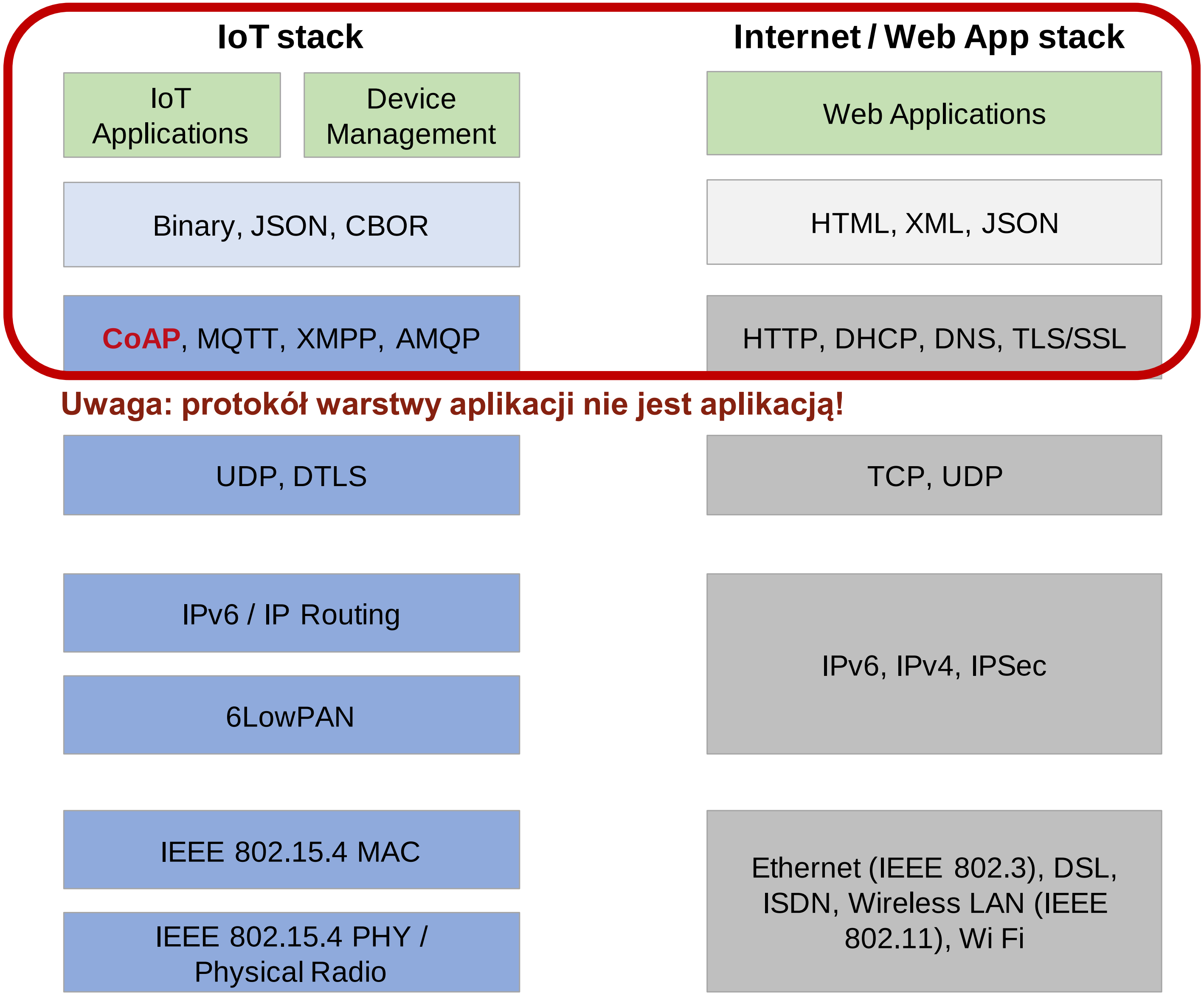
WARSTWA APLIKACJI (2/3)



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

Warstwa aplikacji (3/3)

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



IoT stack	Internet / Web App stack
IoT Applications Device Management	Web Applications
Binary, JSON, CBOR	HTML, XML, JSON
CoAP, MQTT, XMPP, AMQP	HTTP, DHCP, DNS, TLS/SSL
UDP, DTLS	TCP, UDP
IPv6 / IP Routing 6LowPAN	IPv6, IPv4, IPSec
IEEE 802.15.4 MAC IEEE 802.15.4 PHY / Physical Radio	Ethernet (IEEE 802.3), DSL, ISDN, Wireless LAN (IEEE 802.11), Wi Fi

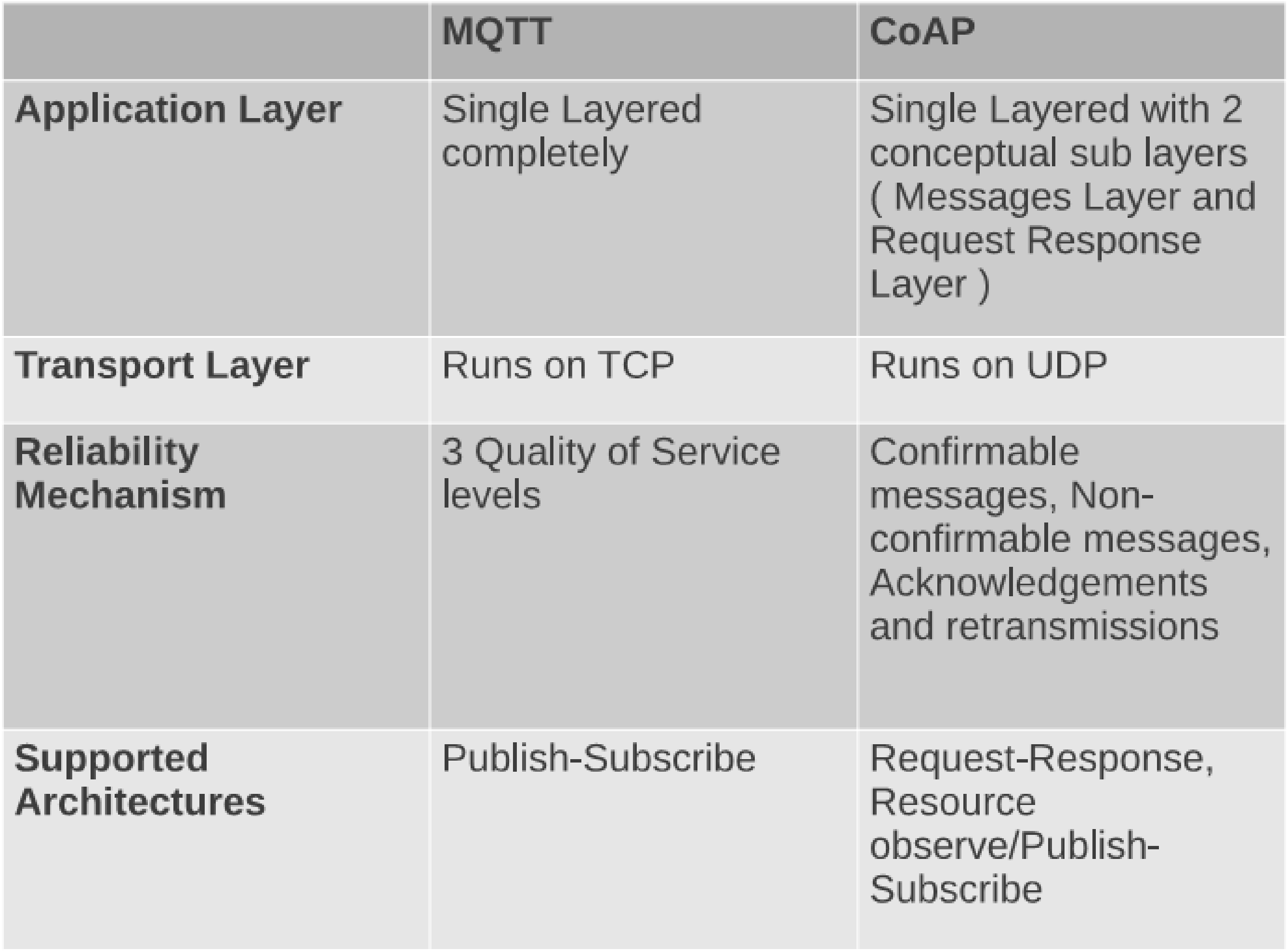
Uwaga: protokół warstwy aplikacji nie jest aplikacją!

Konkurenci w warstwie aplikacji IoT (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

Konkurenci w warstwie aplikacji IoT (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

Źródło: *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]

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

CoRE (Constrained RESTful E.) *RESTful?*

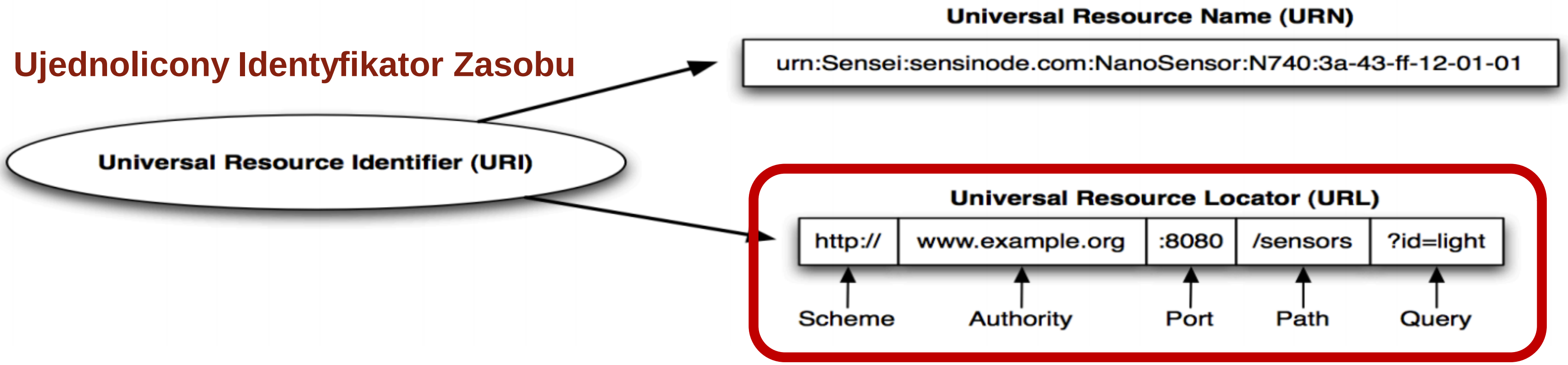
- note: the description below is (somewhat) simplified

- 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)
  - the state is kept only on clients

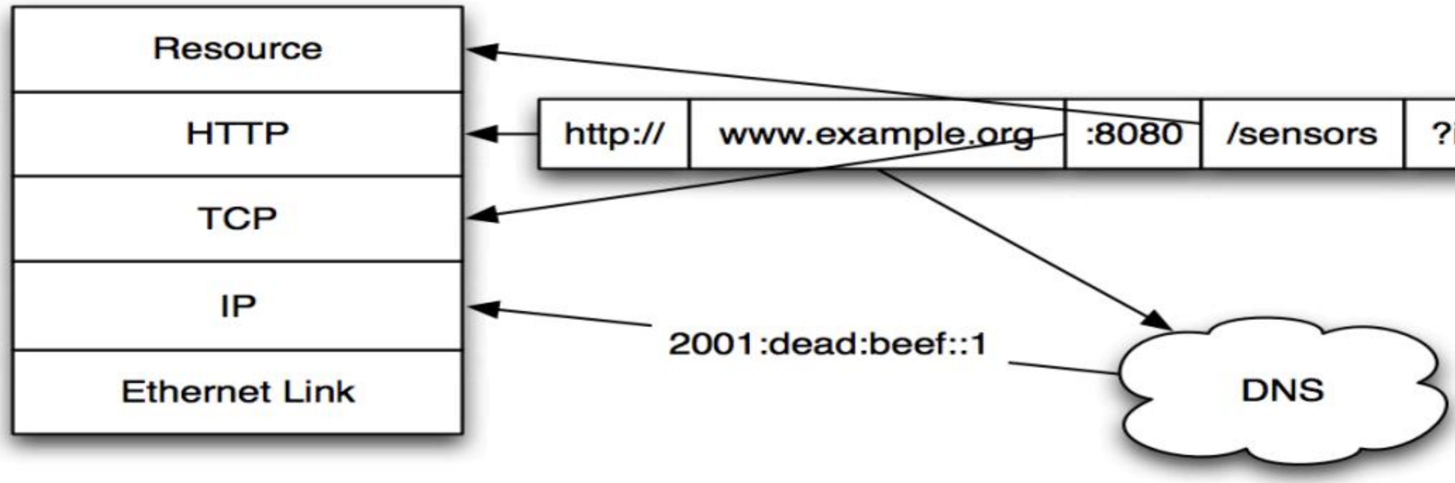
IDENTYFIKATORY URI

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

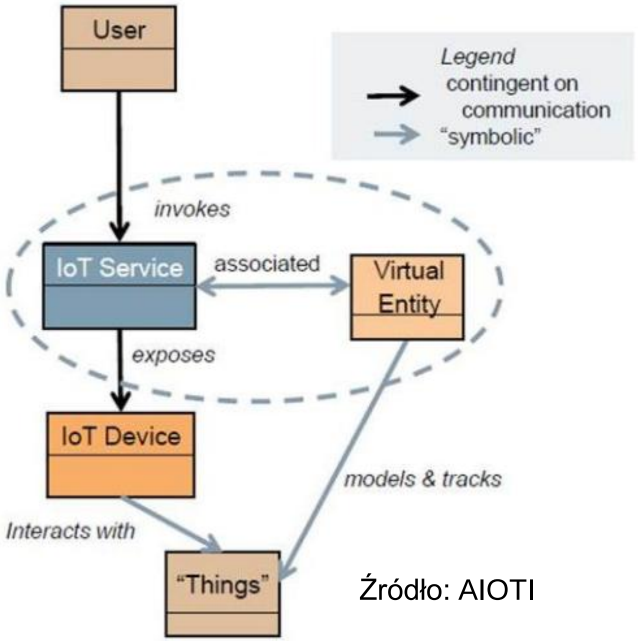
Ujednolicony Identyfikator Zasobu


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

schemat, podmiot (host), port, ścieżka, zapytanie



CoAP – podstawy



Źródło: AIOTI

COAP: GŁÓWNE DOKUMENTY RFC

- [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

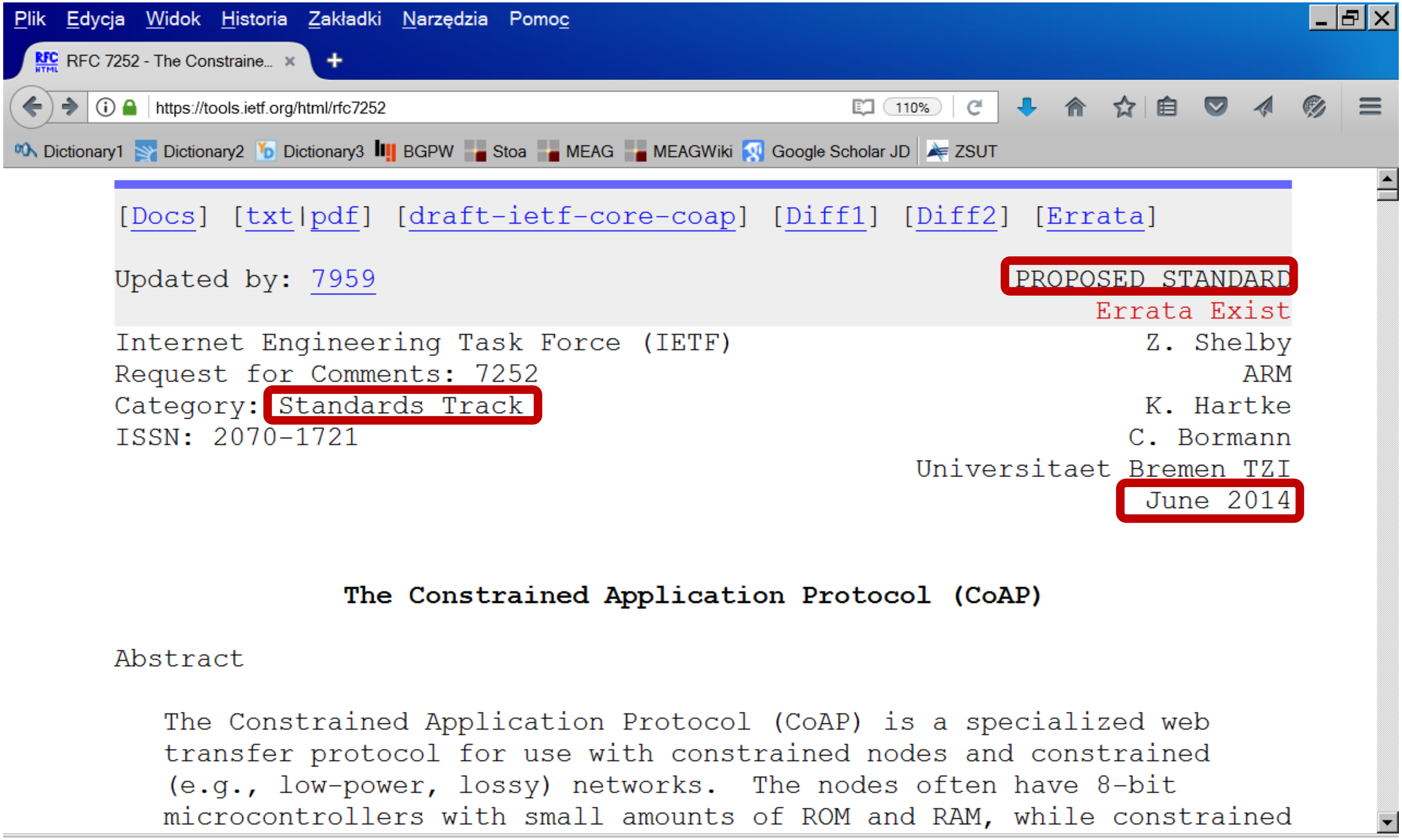
how to transfer big resource representations

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

how to discover resources hosted by a server

RFC 7252 (1/2)

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

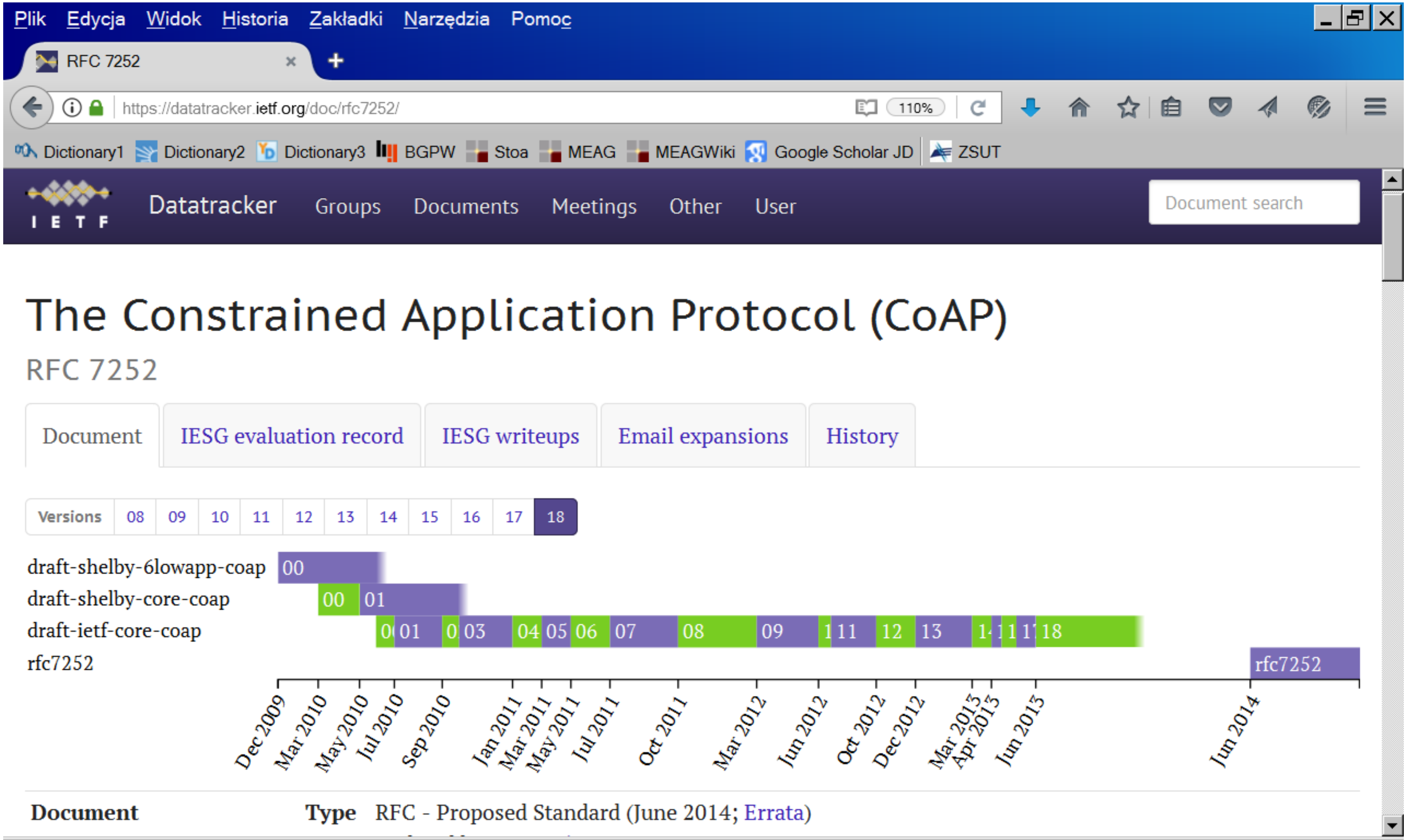


Plik Edycja Widok Historia Zakładki Narzędzia Pomoc
RFC 7252 - The Constraine...
https://tools.ietf.org/html/rfc7252
110%
Dictionary1 Dictionary2 Dictionary3 BGPW Stoa MEAG MEAGWiki Google Scholar JD ZSUT

```
[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)

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

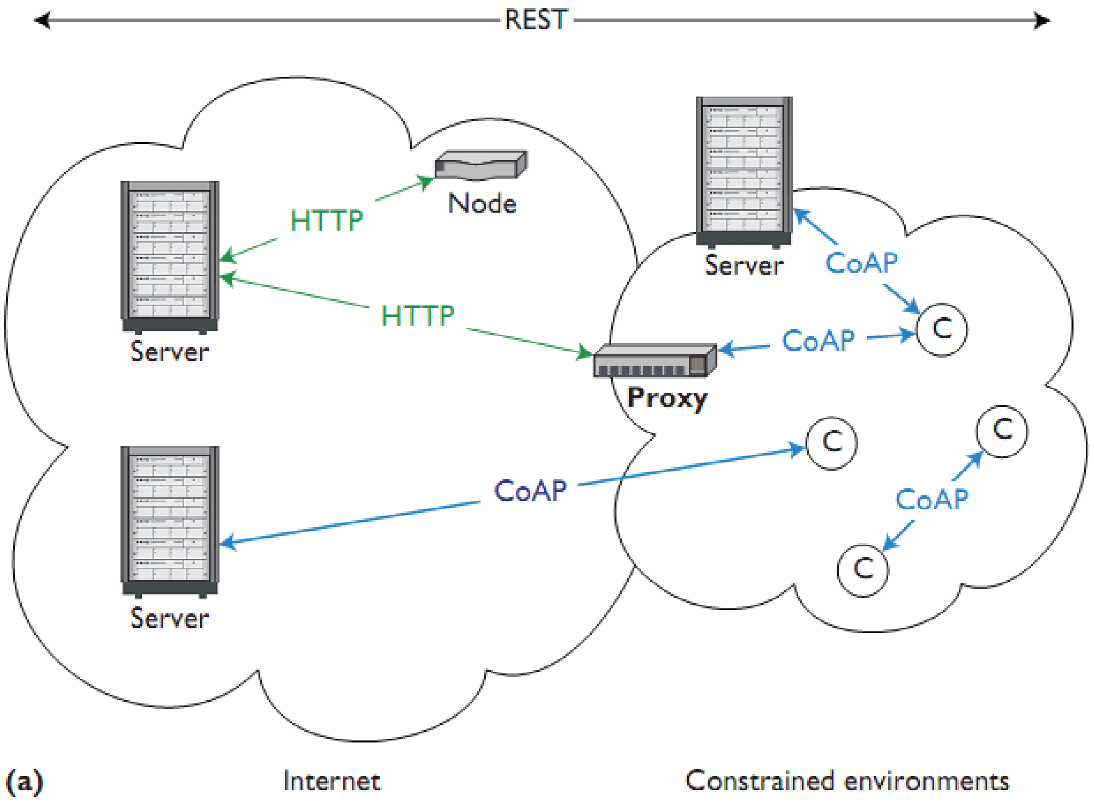


The Constrained Application Protocol (CoAP)

RFC 7252

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

Architektura systemu CoAP



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

(b)

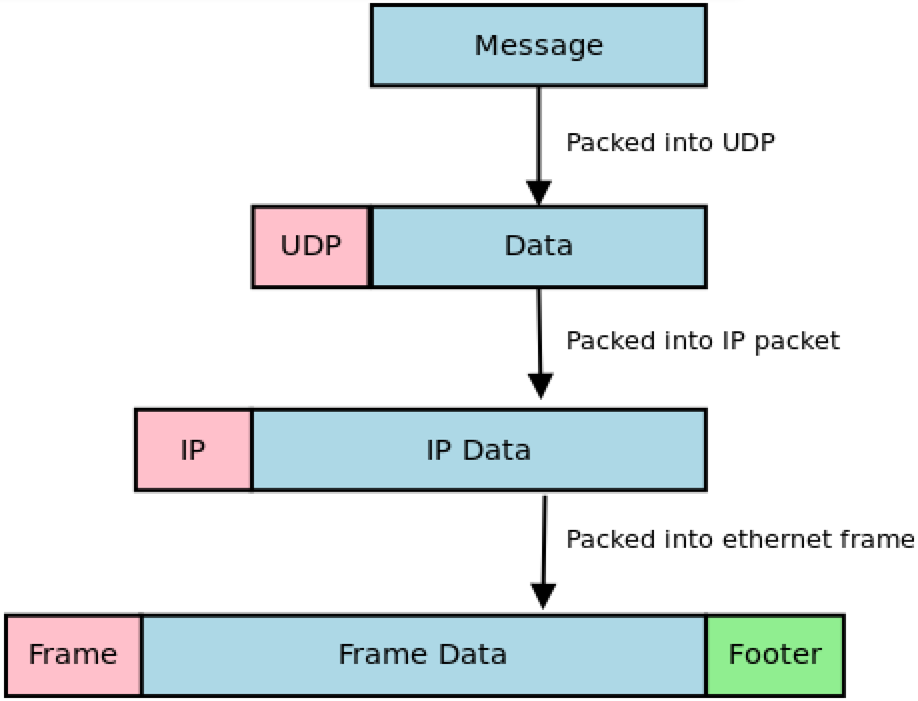
Uwaga: CoAP świadczy usługi aplikacji. CoAP nie jest aplikacją!

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

PODSTAWY UDP

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

 User Datagram Header Format [RFC768]
```

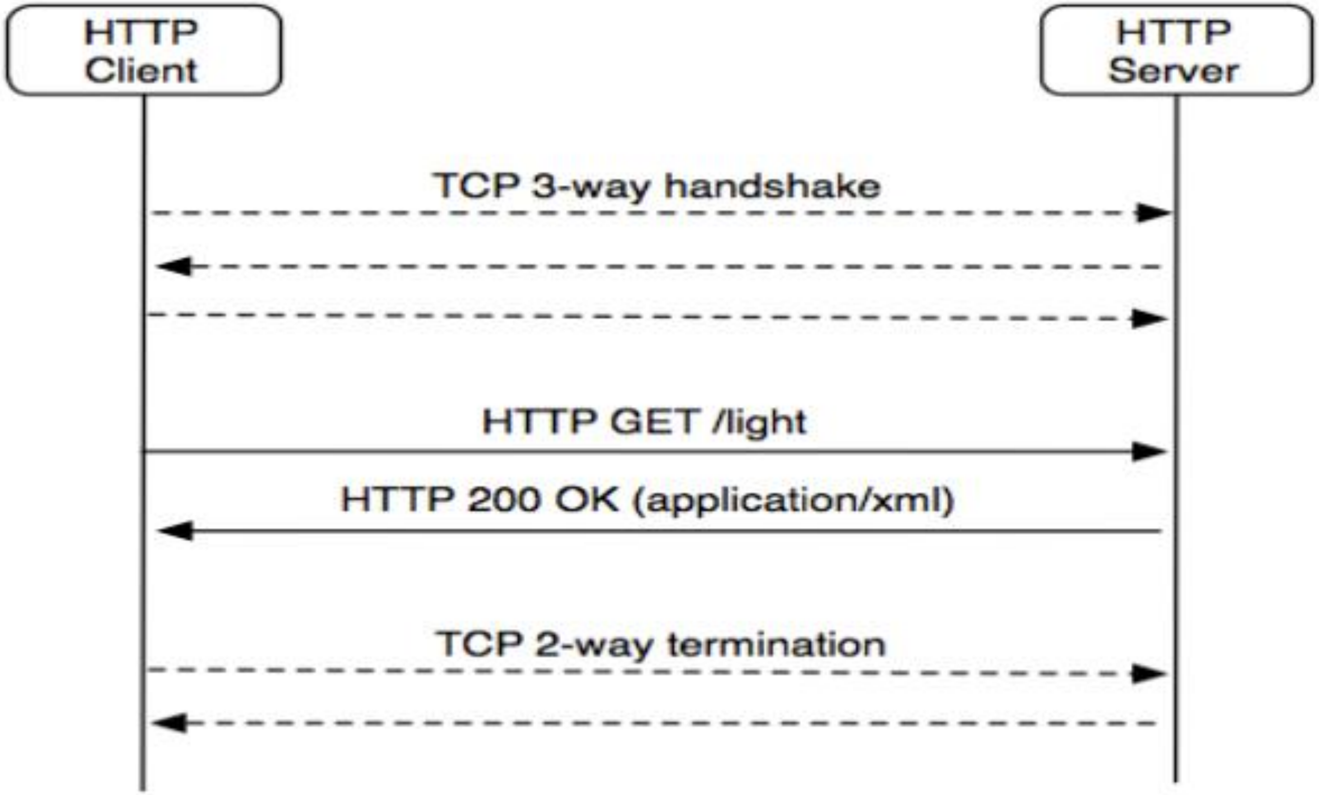
3 strony



http://jamesslocum.com/post/77759061182

- connectionless **bezpołączeniowy**
- 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?

DLACZEGO NIE TCP?



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

CoAP w stosie protokołów

```
                          +----------------------+
                          |     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 SMS is also possible
- CoAP is not the application itself (the application logic is up to you!)

Wiadomości CoAP

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

klient/serwer

- 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)

zapytania/odpowiedzi

- CON (confirmable)/NON (non-confirmable)/ACK/RST
  - CON+ACK: lightweight reliability

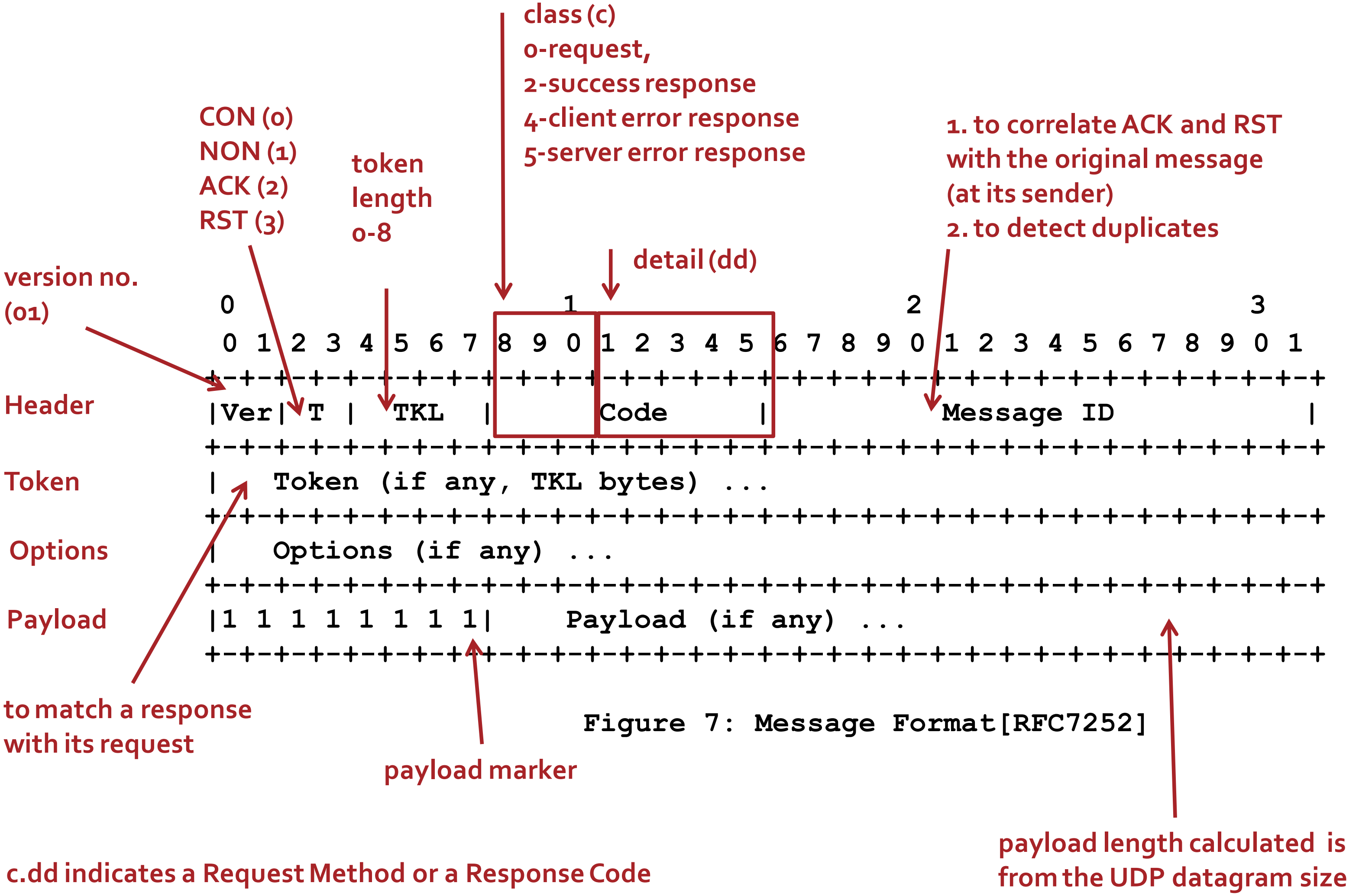
FORMAT WIADOMOŚCI COAP

```
 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]

Najkrótsza wiadomość CoAP: 4B

Format wiadomości CoAP



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)

```
        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
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Header |Ver| T |  TKL  |      Code     |          Message ID           |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Token  |   Token (if any, TKL bytes) ...
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Options|   Options (if any) ...
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Payload|1 1 1 1 1 1 1 1|    Payload (if any) ...
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

to match a response with its request

payload marker

Figure 7: Message Format[RFC7252]

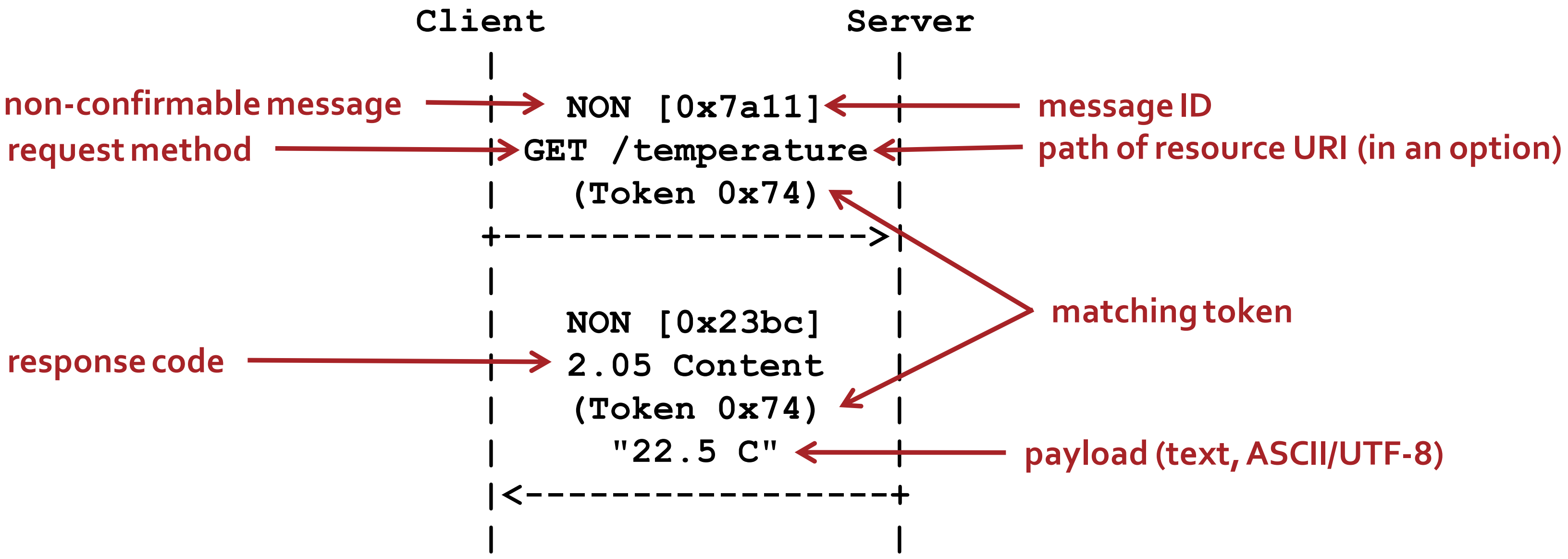
payload length calculated is from the UDP datagram size

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

BEZ NIEZAWODNOŚCI: WIAD. NON-CONFIRMABLE



```
        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)

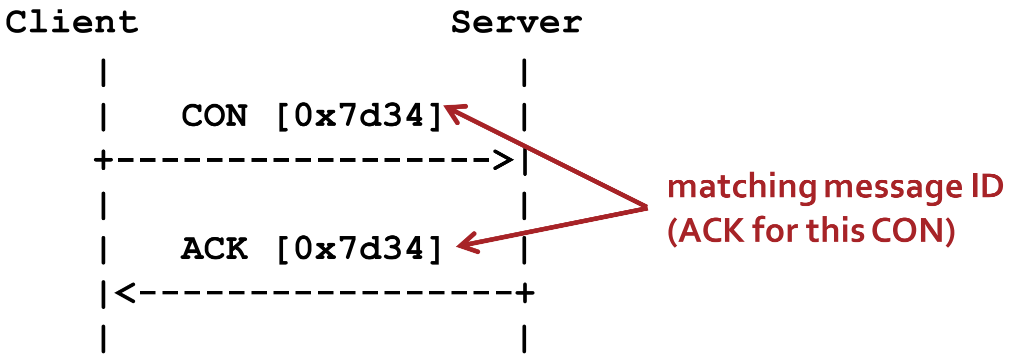
matching token

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

Z NIEZAWODNOŚCIĄ: WIAD. CONFIRMABLE



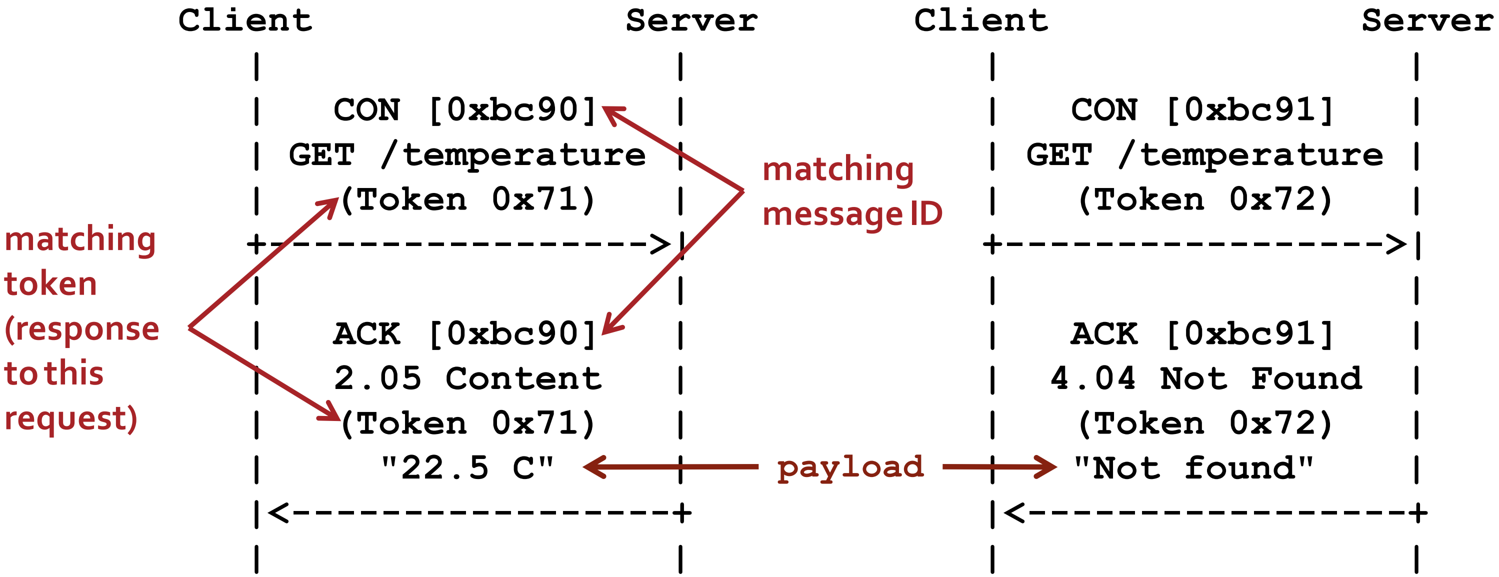
```
Client              Server
   |                  |
   |   CON [0x7d34]   |
   +----------------->|
   |                  |
   |   ACK [0x7d34]   |
   |<-----------------+
   |                  |
```

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 not 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"    |
   |<-----------------+             |<-----------------+
   |                  |             |                  |
```

matching token (response to this request)

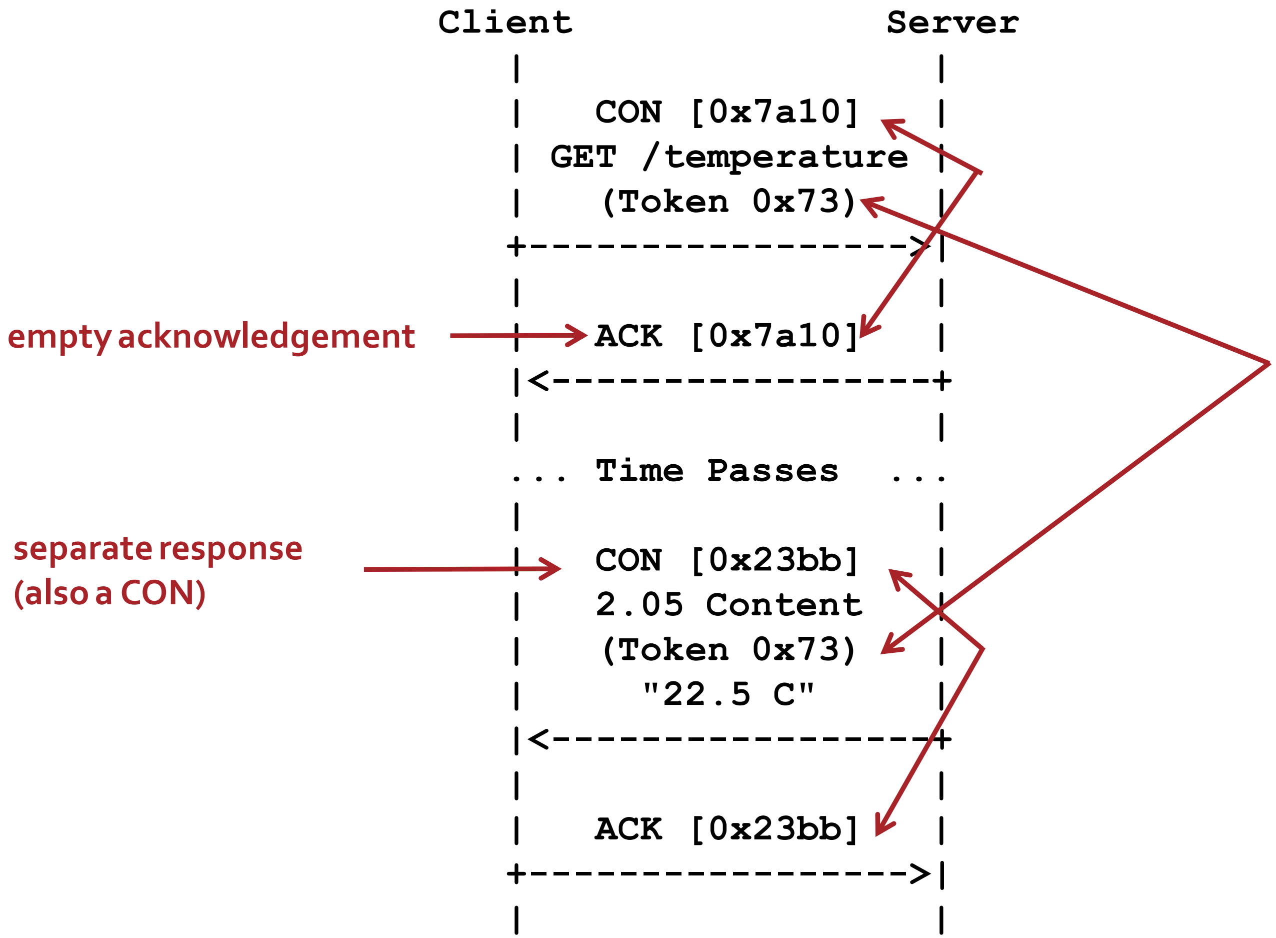
matching message ID

payload

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

- the response carried in ACK (if available immediately) pusta ACK oddzielna odpowiedź

EMPTY ACK AND SEPARATE RESPONSE

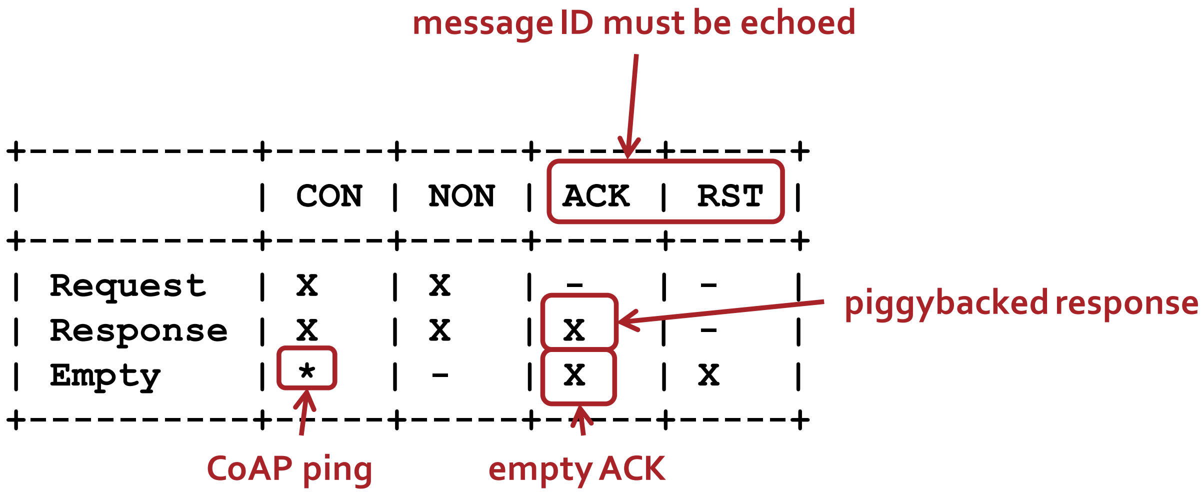


```
      Client              Server
        |                   |
        |   CON [0x7a10]    |
        | GET /temperature  |
        |   (Token 0x73)    |
        +------------------>|
        |                   |
        |   ACK [0x7a10]    |
        |<------------------+
        |              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

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

- if the response not available immediately

UŻYCIE WIADOMOŚCI RÓŻNYCH TYPÓW



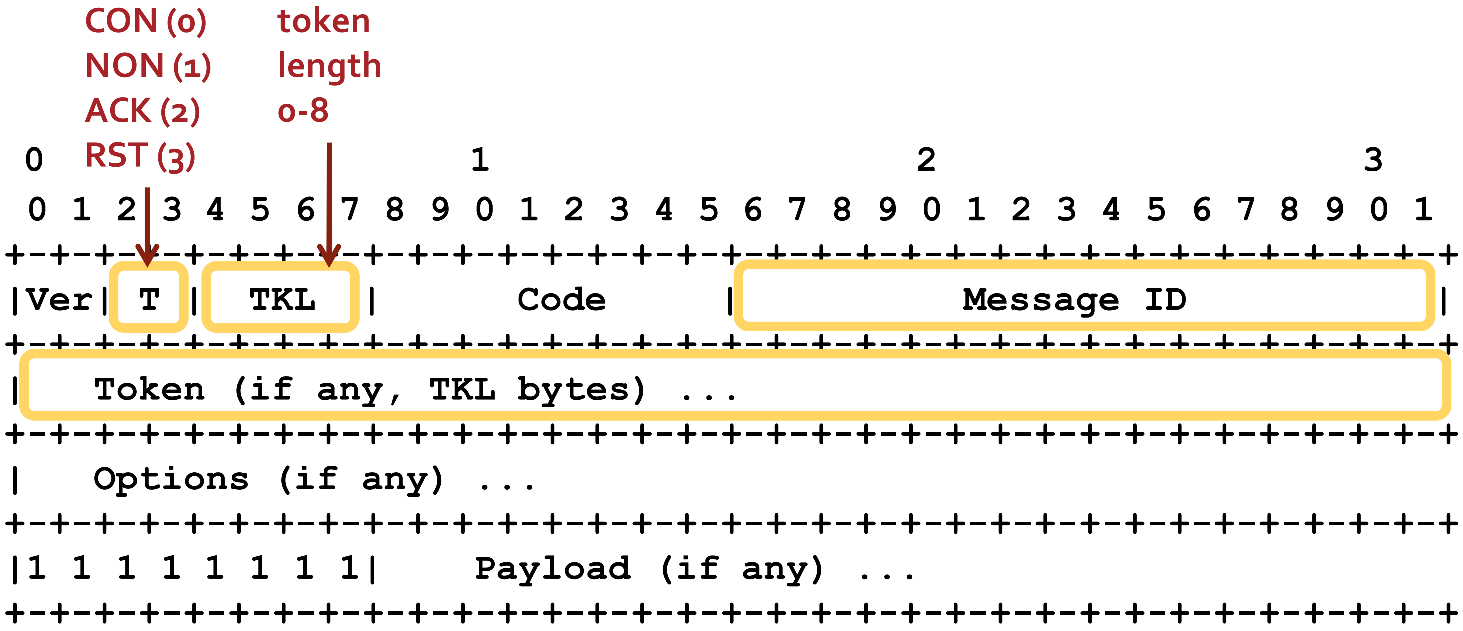
	CON	NON	ACK	RST
Request	X	X	-	-
Response	X	X	X	-
Empty	*	-	X	X

message ID must be echoed
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]

METODY (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]

ODPOWIEDZI

- 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

```
  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]

SUCCESS RESPONSE CODES 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

CLIENT ERROR RESPONSE CODES (SELECTED) 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

Server error response codes (selected) 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

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]

OPCJE COAPA

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]

WYBRANE OPCJE (1/2)

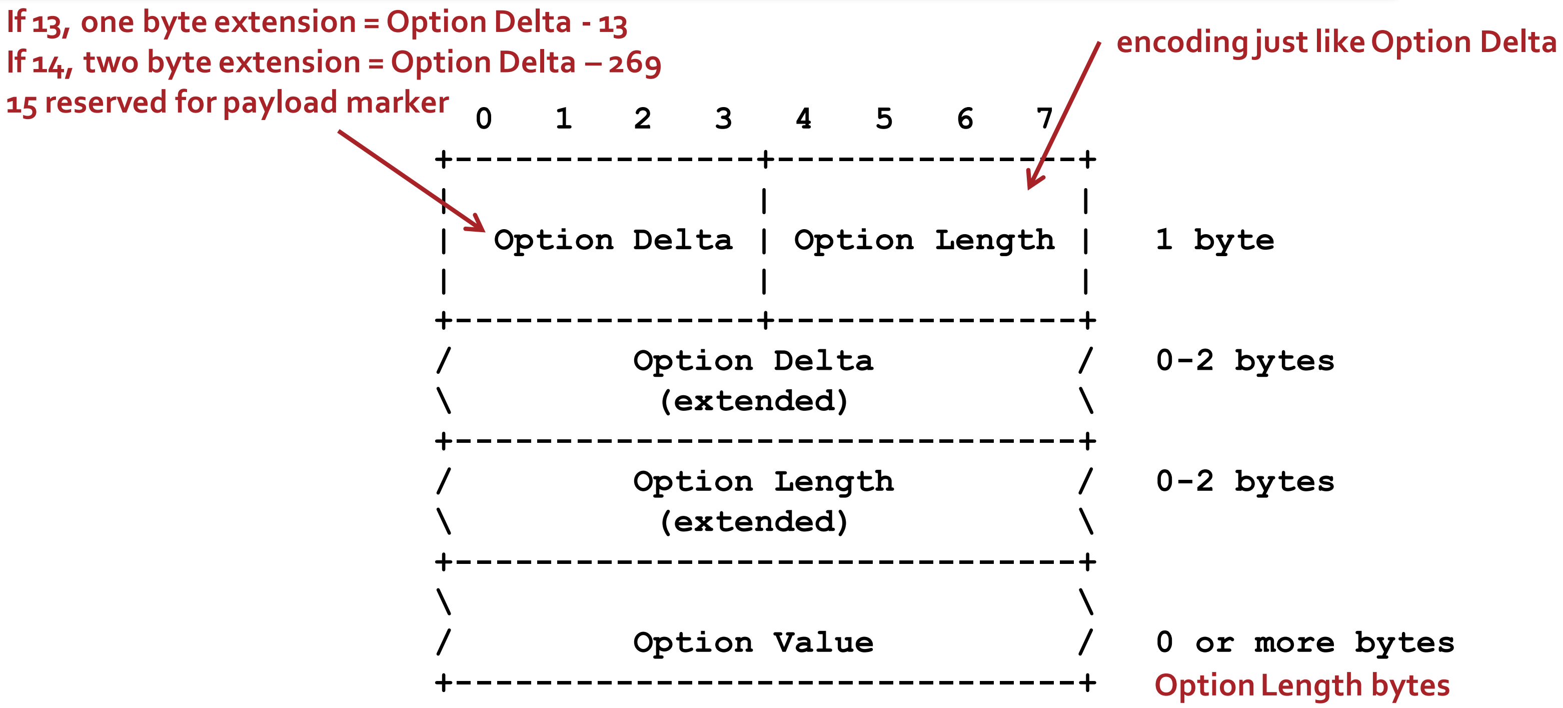
- **`Content-Format`**
  - the representation format of the payload
- **`Etag`**
  - an entity-tag is intended for use as a resource-local identifier for a specific representation of a resource; generated by the server providing the resource; used for validation
- **`Max-Age`**
  - the maximum time a 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

Wybrane opcje (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`**

FORMAT OPCJI



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

```
   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

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

arbitrary binary data

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]

Efficient XML Interchange (binary)

Concise Binary Object Representation →

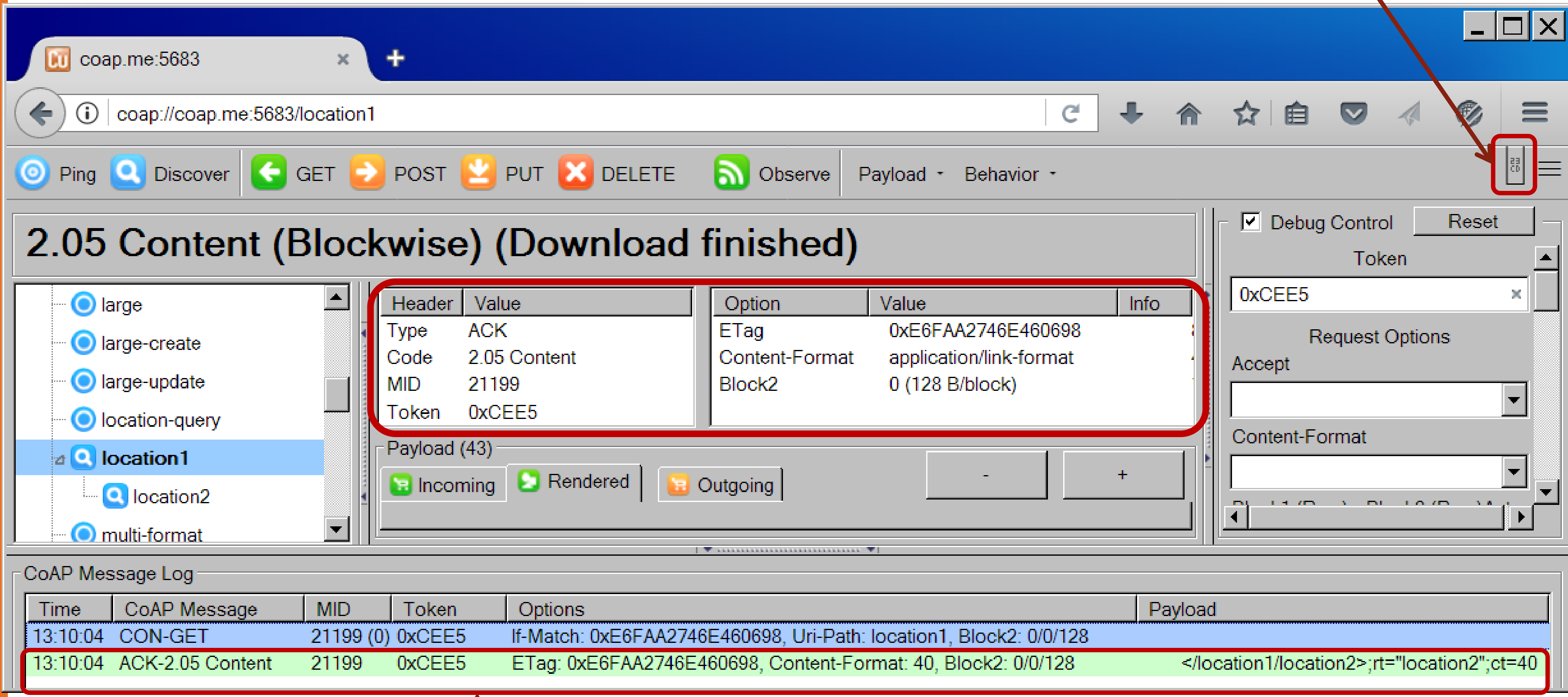
7.4. CoAP Content-Format
Media Type: application/cbor
Id: 60

Źródło: *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)	

Payload (43)

Token: 0xCEE5

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) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

UDP: Received 63 bytes

Ver=1

T=2 (ACK)

TKL=2

response code= 2.05 (Content)

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

PACKET (hex):

4B+ 62,45,52,CF, **header** 0110 0010, 0100 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 = 63B

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,
69,6F,6E,32,22,3B,63,74,3D,34,30

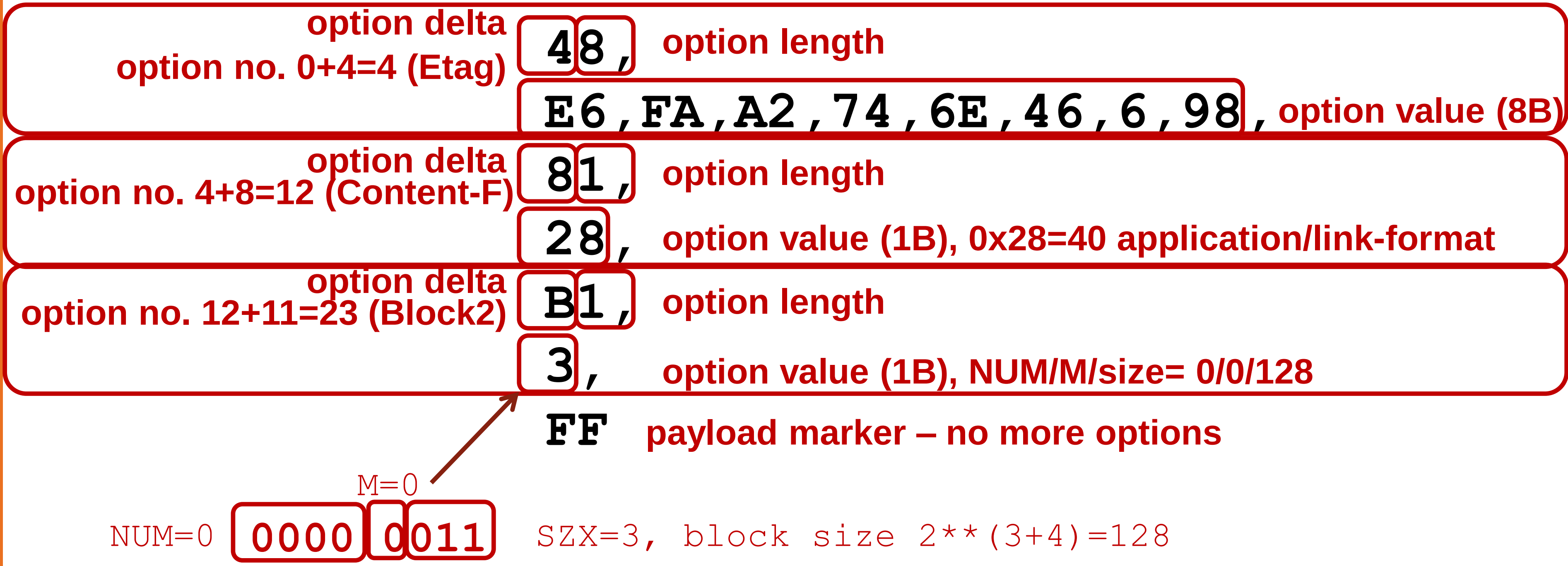
payload:
</location1/location2>;rt="location2";ct=40

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

option delta
option no. 4+8=12 (Content-F)
81, option length
28, option value (1B), 0x28=40 application/link-format

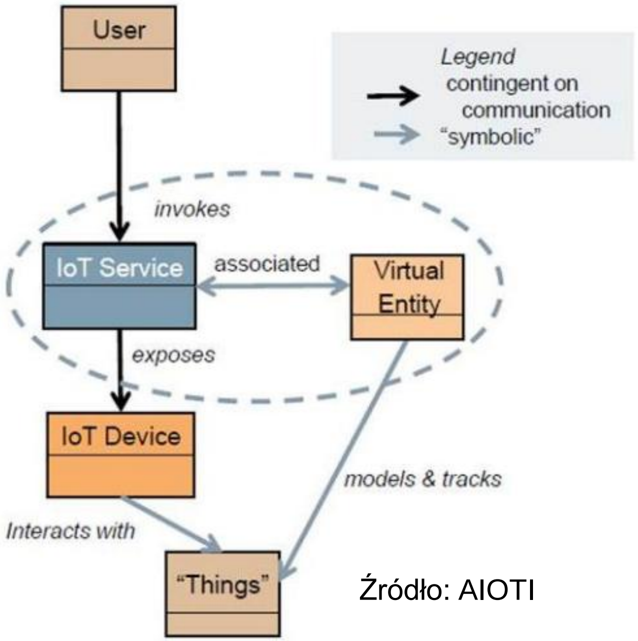
option delta
option no. 12+11=23 (Block2)
B1, option length
3, option value (1B), NUM/M/size= 0/0/128

FF payload marker – no more options

M=0
NUM=0 0000 0011 SZX=3, block size 2**(3+4)=128

Note: the Block2 option is covered below.

Przykłady



Źródło: AIOTI

CON REQUEST; PIGGYBACKED RESPONSE

```
Client  Server
   |      |
   |      |
   +----->|     Header: GET (T=CON, Code=0.01, MID=0x7d34)
   | GET  |   Uri-Path: "temperature"
   |      |
   |      |
   |<-----+     Header: 2.05 Content (T=ACK, Code=2.05, MID=0x7d34)
   | 2.05 |    Payload: "22.3 C"
   |      |
```

```
             0   CON empty token   1 GET                 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
            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  16 bytes
            | 1 | 0 |   0   |     GET=1     |          MID=0x7d34           |
            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 0+11=11
 Uri-Path   |  11   |  11   |      "temperature" (11 B) ...
            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                     Option Length      Option Value

                 ACK empty token     Content
            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            | 1 | 2 |   0   |    2.05=69    |          MID=0x7d34           |  11 bytes
            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            |1 1 1 1 1 1 1 1|      "22.3 C" (6 B) ...
            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              payload marker         payload
```

Figure 16: Confirmable Request; Piggybacked Response [RFC7252]

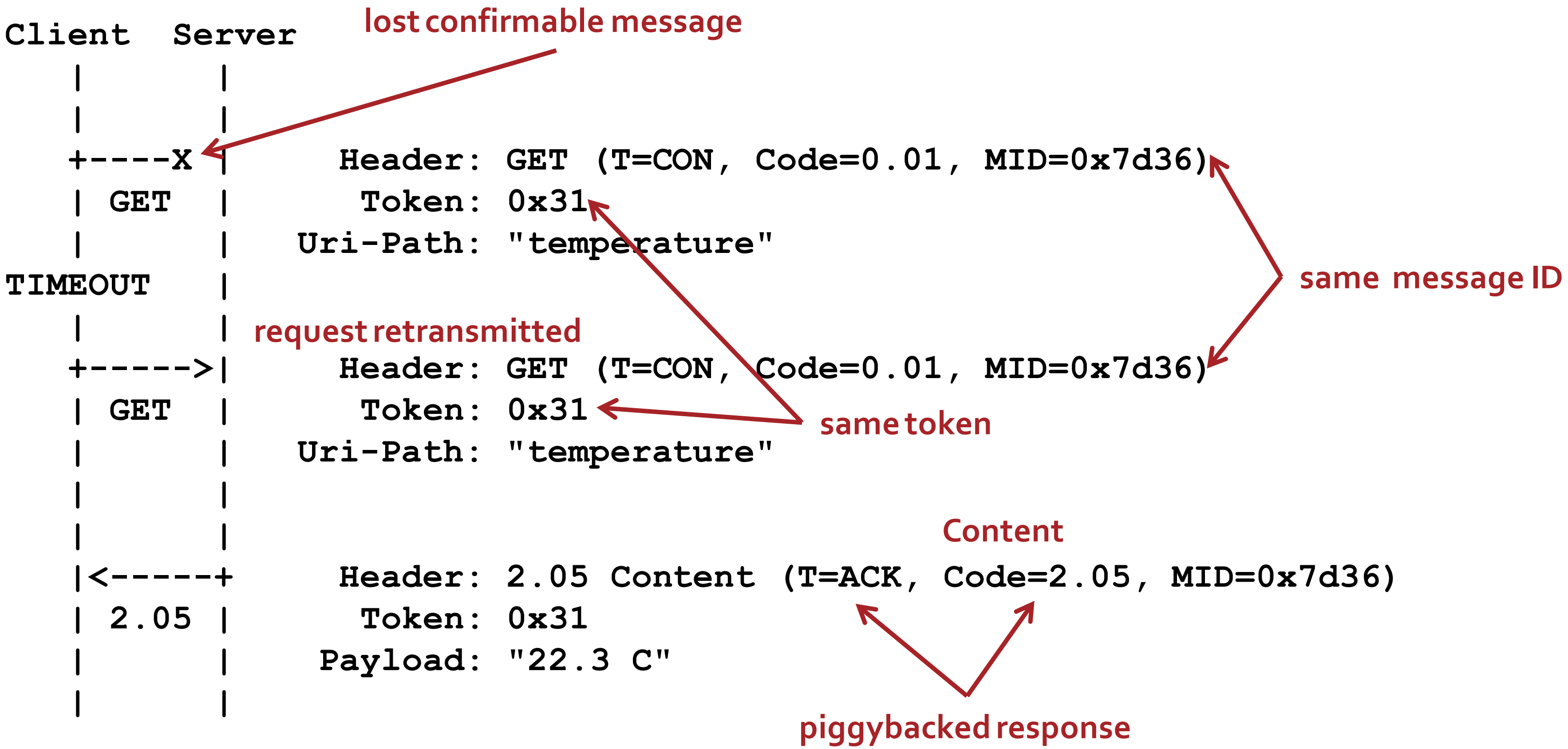
CON REQUEST; PIGGYBACKED RESPONSE, WITH TOKEN

```
Client  Server
   |      |
   +----->|     Header: GET (T=CON, Code=0.01, MID=0x7d35)
   | GET  |      Token: 0x20
   |      |   Uri-Path: "temperature"
   |      |
   |<-----+     Header: 2.05 Content (T=ACK, Code=2.05, MID=0x7d35)
   | 2.05 |      Token: 0x20
   |      |    Payload: "22.3 C"
   |      |
```

```
 0      token length 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 | 0 |   1   |     GET=1     |          MID=0x7d35           |  17 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x20 token|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  11   |  11   |      "temperature" (11 B) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

        token length
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 | 2 |   1   |    2.05=69    |          MID=0x7d35           |  12 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x20 token|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1| "22.3 C" (6 B) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 17: Confirmable Request; Piggybacked Response, with token [RFC7252]

CON REQ. RETRANSMITTED; PIGGYBACKED RESPONSE



```
Client  Server
   |      |
   |      |
   +----X |     Header: GET (T=CON, Code=0.01, MID=0x7d36)
   | GET  |      Token: 0x31
   |      |   Uri-Path: "temperature"
TIMEOUT   |
   |      |
   +----->|     Header: GET (T=CON, Code=0.01, MID=0x7d36)
   | GET  |      Token: 0x31
   |      |   Uri-Path: "temperature"
   |      |
   |      |
   |<-----+     Header: 2.05 Content (T=ACK, Code=2.05, MID=0x7d36)
   | 2.05 |      Token: 0x31
   |      |    Payload: "22.3 C"
   |      |
```

Figure 18: Confirmable Request (Retransmitted); Piggybacked Response [RFC7252]

CON REQ.; PIGGYBACKED RESPONSE RETRANSMITTED

```
Client  Server
   |      |
   |      |
   +----->|     Header: GET (T=CON, Code=0.01, MID=0x7d37)
   | GET  |      Token: 0x42
   |      |   Uri-Path: "temperature"
   |      |
   |      |
   | X----+     Header: 2.05 Content (T=ACK, Code=2.05, MID=0x7d37)
   | 2.05 |      Token: 0x42
   |      |    Payload: "22.3 C"
TIMEOUT   |
   |      |
   +----->|     Header: GET (T=CON, Code=0.01, MID=0x7d37)
   | GET  |      Token: 0x42
   |      |   Uri-Path: "temperature"
   |      |
   |      |
   |<-----+     Header: 2.05 Content (T=ACK, Code=2.05, MID=0x7d37)
   | 2.05 |      Token: 0x42
   |      |    Payload: "22.3 C"
   |      |
```

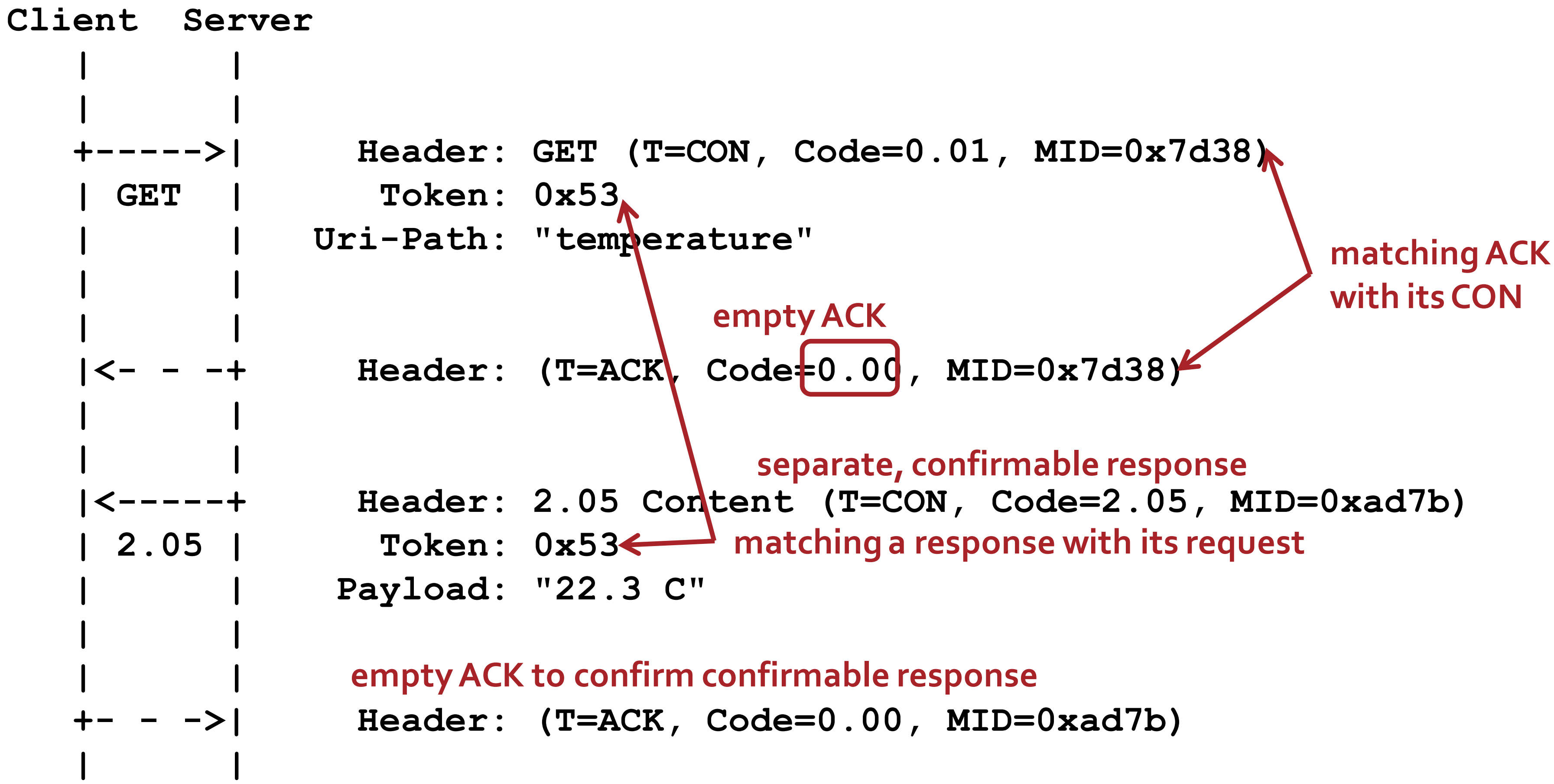
lost piggybacked response

request retransmitted

all messages: same message ID, same token

Figure 19: Confirmable Request; Piggybacked Response (Retransmitted) RFC[7252]

CON REQUEST; SEPARATE RESPONSE



```
Client   Server
   |       |
   |       |
   +----->|      Header: GET (T=CON, Code=0.01, MID=0x7d38)
   | GET  |       Token: 0x53
   |       |    Uri-Path: "temperature"
   |       |
   |       |
   |<- - -+      Header: (T=ACK, Code=0.00, MID=0x7d38)
   |       |
   |       |
   |<-----+      Header: 2.05 Content (T=CON, Code=2.05, MID=0xad7b)
   | 2.05 |       Token: 0x53
   |       |     Payload: "22.3 C"
   |       |
   |       |
   +- - ->|      Header: (T=ACK, Code=0.00, MID=0xad7b)
   |       |
```

matching ACK with its CON

empty ACK

separate, confirmable response

matching a response with its request

empty ACK to confirm confirmable response

Figure 20: Confirmable Request; Separate Response [RFC7252]

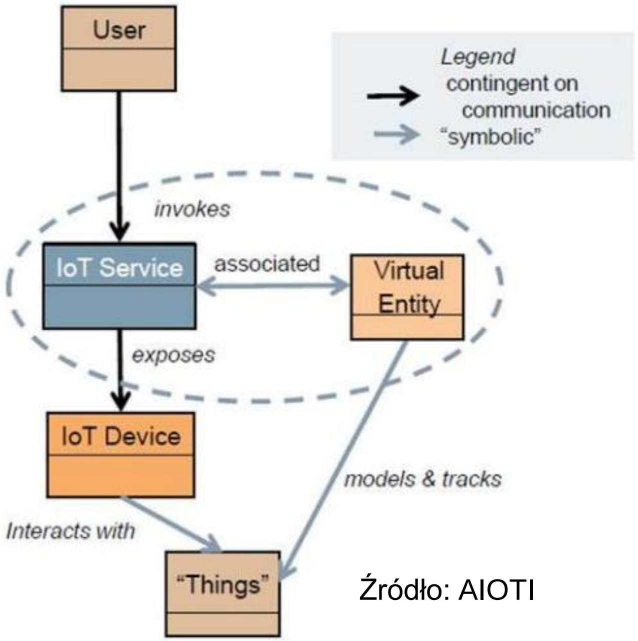
the response to CON request could also be NON

CON REQUEST; SEPARATE RESPONSE (UNEXPECTED)

```
               Client  Server
                  |       |
                  |       |
                  +----->|      Header: GET (T=CON, Code=0.01, MID=0x7d39)
                  | GET  |      Token: 0x64
crash and         |       |      Uri-Path: "temperature"
reboot:        CRASH      |
                                 unexpected ACK silently ignored
loss of state     |       |
                  |<- - -+      Header: (T=ACK, Code=0.00, MID=0x7d39)
                  |       |
                                 unexpected separate response, confirmable
                  |       |
                  |<-----+      Header: 2.05 Content (T=CON, Code=2.05, MID=0xad7c)
                  | 2.05 |      Token: 0x64
                  |       |      Payload: "22.3 C"
                  |       |
                                 RST: rejecting the confirmable response
                  |       |
                  +- - ->|      Header: (T=RST, Code=0.00, MID=0xad7c)
                  |       |
```

Figure 21: Confirmable Request; Separate Response (Unexpected) [RFC7252]

Caching and proxying



Źródło: AIOTI

CACHING

- CoAP endpoints may cache responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests

- reuse a prior response message (a **stored response**)

- two mechanisms: freshness and validation

CACHING: FRESHNESS

- a stored response is reused without contacting the server
  - the **`Max-Age`** Option indicates how long the response is fresh
  - default: 60s

CACHING: VALIDATION

- a new request is required, but it is possible to reuse the payload of a stored response
  - need to validate
  - responses are tagged by the server, with the **`ETag`** Option
  - the client can inquire if a stored response is valid by sending its ETag

PROXYING

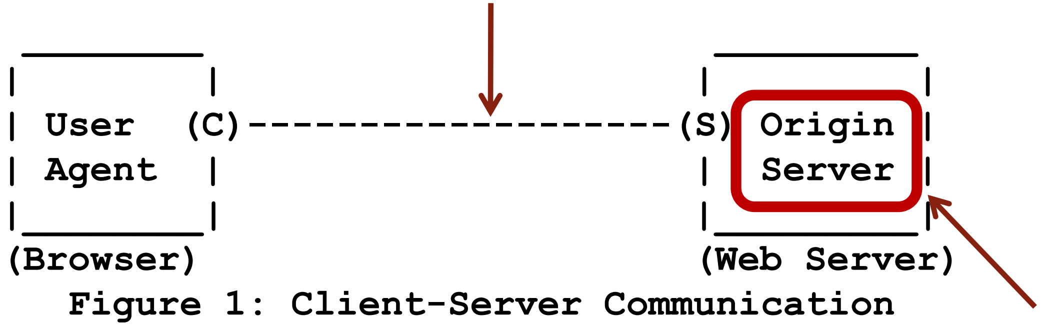
- a proxy is tasked by clients to perform requests on their behalf

- proxy classification 1:
  - **forward proxy**: explicitly selected by clients (as a proxy)
  - **reverse proxy**: the client is not aware that it talks to a proxy

- proxy classification 2:
  - **CoAP-to-CoAP** proxy
  - **cross proxy**: translates from or to a different protocol

NO PROXY – JUST ORIGIN SERVER

URI split into the `Uri-Host` (has a default), `Uri-Port` (has a default), `Uri-Path`, and `Uri-Query` Options



```
 _________                         _________
|         |                       |         |
| User   (C)---------------------(S) Origin |
| Agent   |                       |  Server |
|_________|                       |_________|
(Browser)                         (Web Server)
```

Figure 1: Client-Server Communication

that's where the resource really is

Źródło: *RESTful Design for Internet of Things Systems*
A. Keranen, M. Kovatsch, K. Hartke,
Internet -Draft, draft-keranen-t2trg-rest-iot-03, 2016

REVERSE PROXY

```
 __________                         ___________
|          |                       |           |
| User    (C)--------------------(S) Origin    |
| Agent    |                       |  Server   |
|__________|                       |___________|
(Browser)                          (Web Server)
   Figure 1: Client-Server Communication
```

Źródło: *RESTful Design for Internet of Things Systems*
A. Keranen, M. Kovatsch, K. Hartke,
Internet -Draft, draft-keranen-t2trg-rest-iot-03, 2016

the client talks as if to an origin server

```
 __________                           __________        ___________
|          |                         |          |      |           |
| User    (C)----------------------(S) Inter-  (x)---(x) Origin    |
| Agent    |                         |  mediary |      |  Server   |
|__________|                         |__________|      |___________|
(Browser)                          (Reverse Proxy)    (Legacy System)
          Figure 3: Communication with Reverse Proxy
```

need an URI mapping for the proxy

FORWARD PROXY

Źródło: *RESTful Design for Internet of Things Systems*
A. Keranen, M. Kovatsch, K. Hartke,
Internet -Draft, draft-keranen-t2trg-rest-iot-03, 2016

```
 ________                       _________
|        |                     |         |
| User  (C)-------------------(S) Origin |
| Agent  |                     |  Server |
|________|                     |_________|
(Browser)                      (Web Server)
```

Figure 1: Client-Server Communication

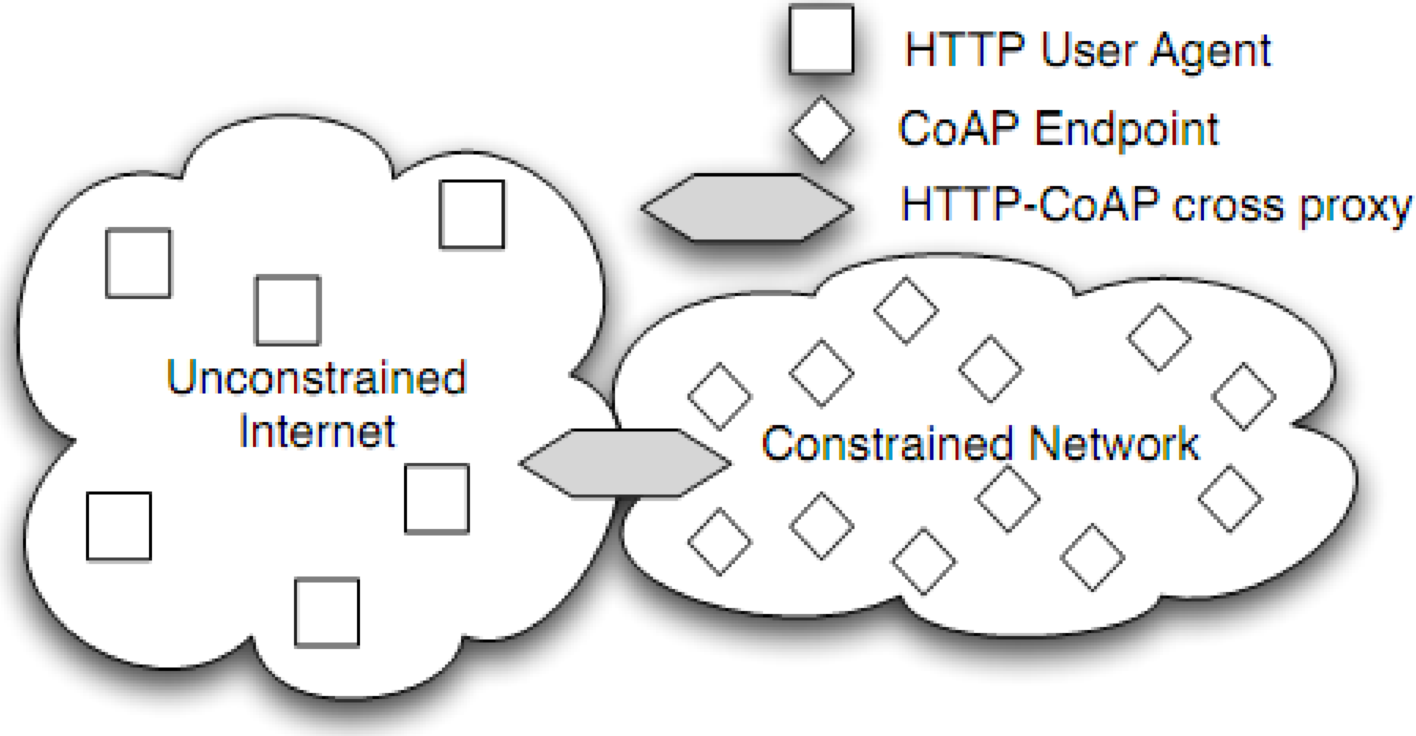
the request URI in a proxy request is in the `Proxy-Uri` Option

```
 ________       __________                        _________
|        |     |          |                      |         |
| User  (C)---(S) Inter- (C)--------------------(S) Origin |
| Agent  |     |  mediary |                      |  Server |
|________|     |__________|                      |_________|
(Browser)     (Forward Proxy)                    (Web Server)
```

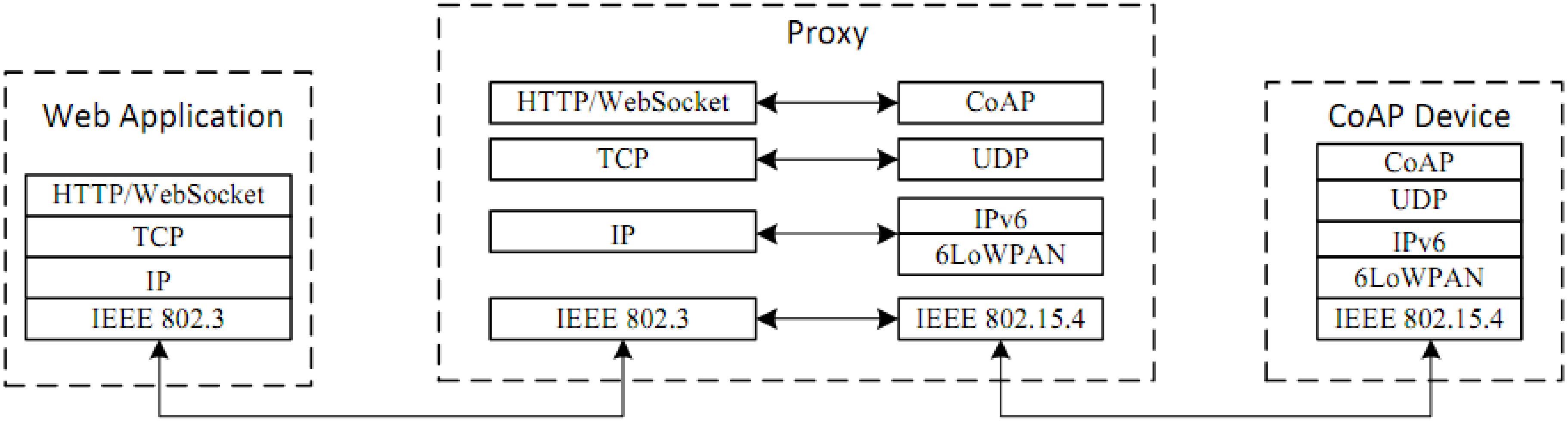
Figure 2: Communication with Forward Proxy

HTTP-TO-COAP PROXY (1/2)

Źródło:
HTTP-CoAP Cross Protocol Proxy: An Implementation Viewpoint
A. P. Castellani, Th. Fossati, S. Loreto
MASS 2012



HTTP-TO-COAP PROXY (2/2)



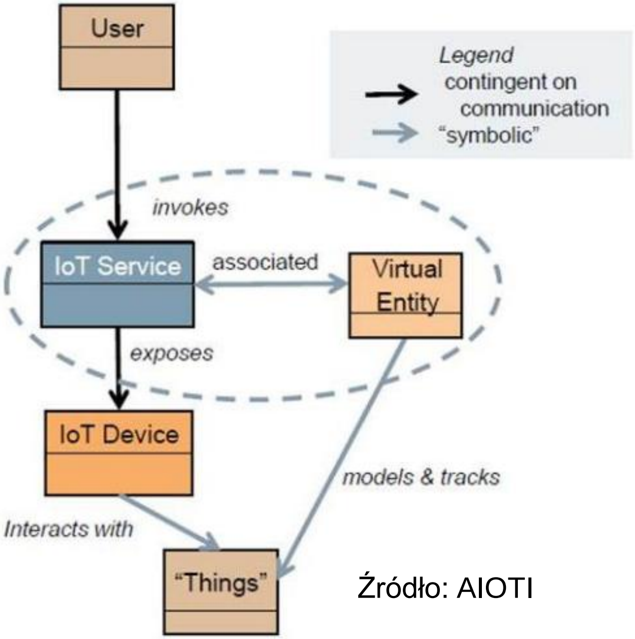
Source:
A Proxy Design to Leverage the Interconnection of CoAP Wireless Sensor Networks with Web Applications
A. Ludovici, A. Calveras
Sensors, 2015

- CoAP is designed with HTTP-to-CoAP proxies in mind

„Obserwowanie” zasobów

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

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



Źródło: AIOTI

W CZYM PROBLEM?

- the client/server model does not work well when a client wants to have a current representation of a resource over a period of time.
- HTTP: polling, long polling
- aside: polling vs. interrupts
- pull vs. push

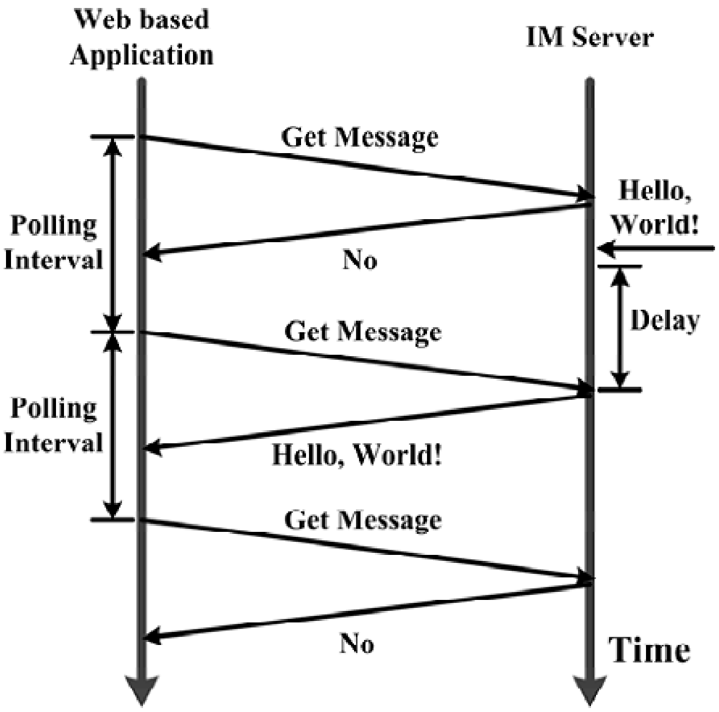
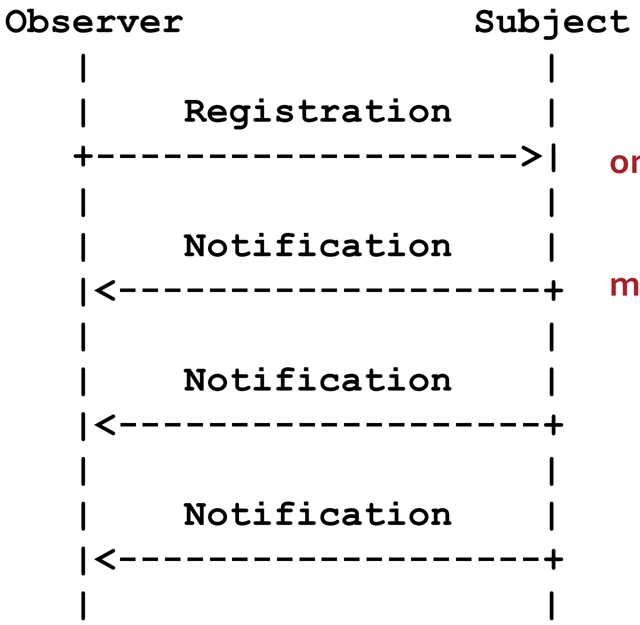


Figure 1. Workflow of HTTP Polling

Źródło:
Research on Server Push Methods in Web Browser based Instant Messaging Applications
Kai Shuang, Feng Kai
Journal of Software, Vol 8, No 10 (2013)

WZORZEC PROJEKTOWY „OBSERWATOR”



```
Observer                 Subject
   |                        |
   |     Registration       |
   +----------------------->|
   |                        |
   |     Notification       |
   |<-----------------------+
   |                        |
   |     Notification       |
   |<-----------------------+
   |                        |
   |     Notification       |
   |<-----------------------+
   |                        |
```

one registration

multiple notifications

Figure 1: The Observer Design Pattern [RFC7641]

Opcja Observe

No.	Name	Format	Length	Default
6	Observe	uint	0-3 B	(none)

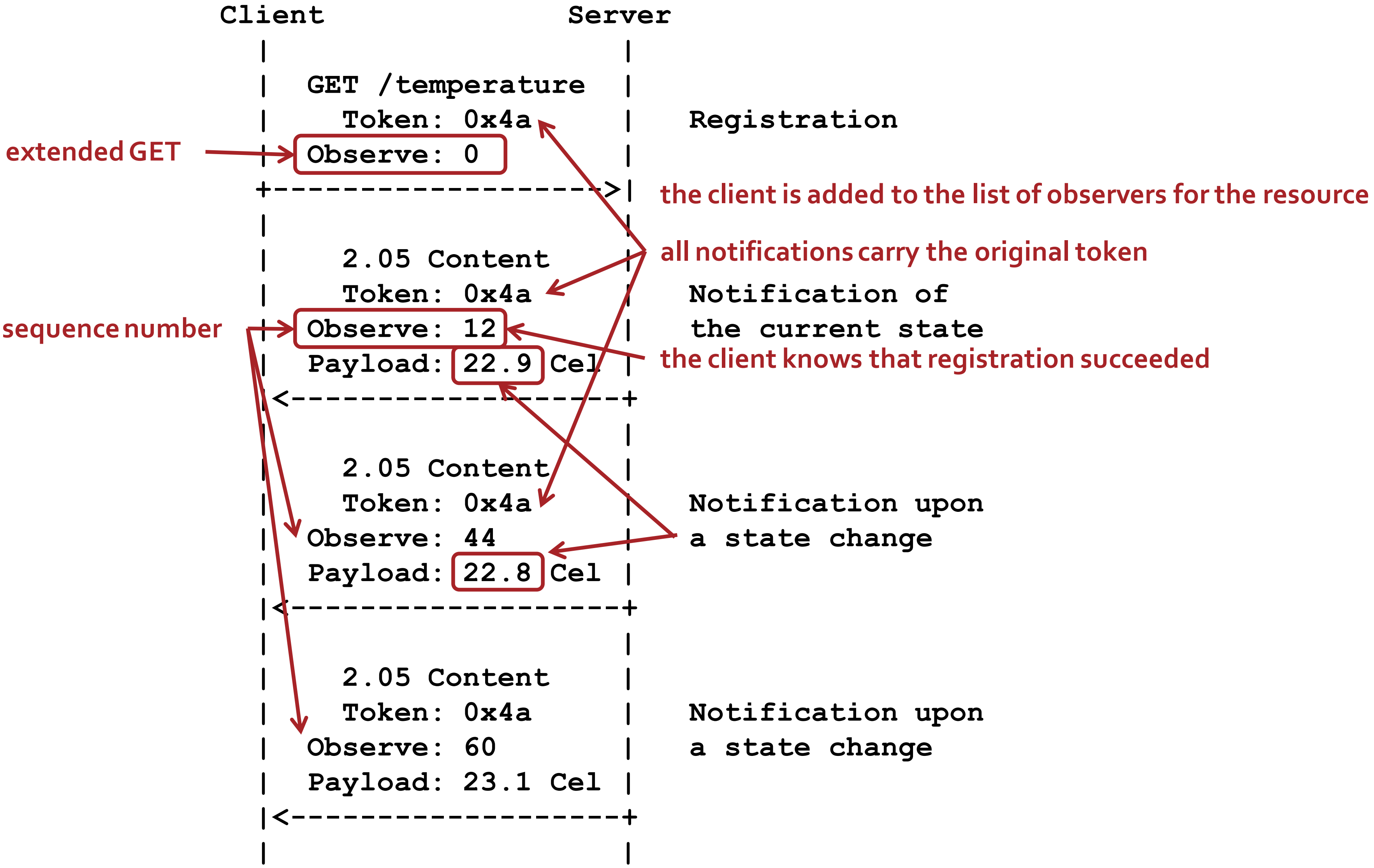
Table 1: The Observe Option [RFC7641]

value in GET:
0 add to the list of observers
1 remove from the list of observers
value in response/notification:
sequence number

- a GET request with the Observe Option:
  - retrieves a current representation, but also ... lista obserwatorów
  - requests the server to add/remove an entry in the **list of observers** of the resource
  - 0 (register) adds the entry to the list, if not present
  - 1 (deregister) removes the entry from the list, if present

- a response with the Observe Option
  - the original response and each subsequent notification
  - the option value is a sequence number for reordering detection
  - in every notification the token is as in the original request

OBSERWOWANIE ZASOBU: PRZYKŁAD



```
Client                  Server
   |                       |
   |  GET /temperature     |
   |    Token: 0x4a        |   Registration
   |  Observe: 0           |
   +---------------------->|
   |                       |
   |    2.05 Content       |
   |    Token: 0x4a        |   Notification of
   |  Observe: 12          |   the current state
   |  Payload: 22.9 Cel    |
   |<----------------------+
   |                       |
   |    2.05 Content       |
   |    Token: 0x4a        |   Notification upon
   |  Observe: 44          |   a state change
   |  Payload: 22.8 Cel    |
   |<----------------------+
   |                       |
   |    2.05 Content       |
   |    Token: 0x4a        |   Notification upon
   |  Observe: 60          |   a state change
   |  Payload: 23.1 Cel    |
   |<----------------------+
   |                       |
```

extended GET

the client is added to the list of observers for the resource

all notifications carry the original token

sequence number

the client knows that registration succeeded

Figure 2: Observing a resource in CoAP [RFC7641]

LISTA OBSERWATORÓW

- created for a given resource by the server when it receives a GET request with an Observe Option set to 0 (register)
- the list entry consists of the client endpoint and the token specified by the client in the request
- how long a client remains on the list?
  - the server can send a confirmable notification; if it does not receive ACK, it will assume that the client is no longer interested
  - the client may send RST in reaction to a notification
  - the client may deregister with Observe Option set to 1

Consistency model: eventual consistency

- the goal is to keep the client in sync with the changes in the state of the resource

- sometimes, however, the client gets out-of-sync
  - the server may skip some notifications if changes occur too often
  - notification latency
  - lost notifications
  - the server may decide to drop the client from the list of observers

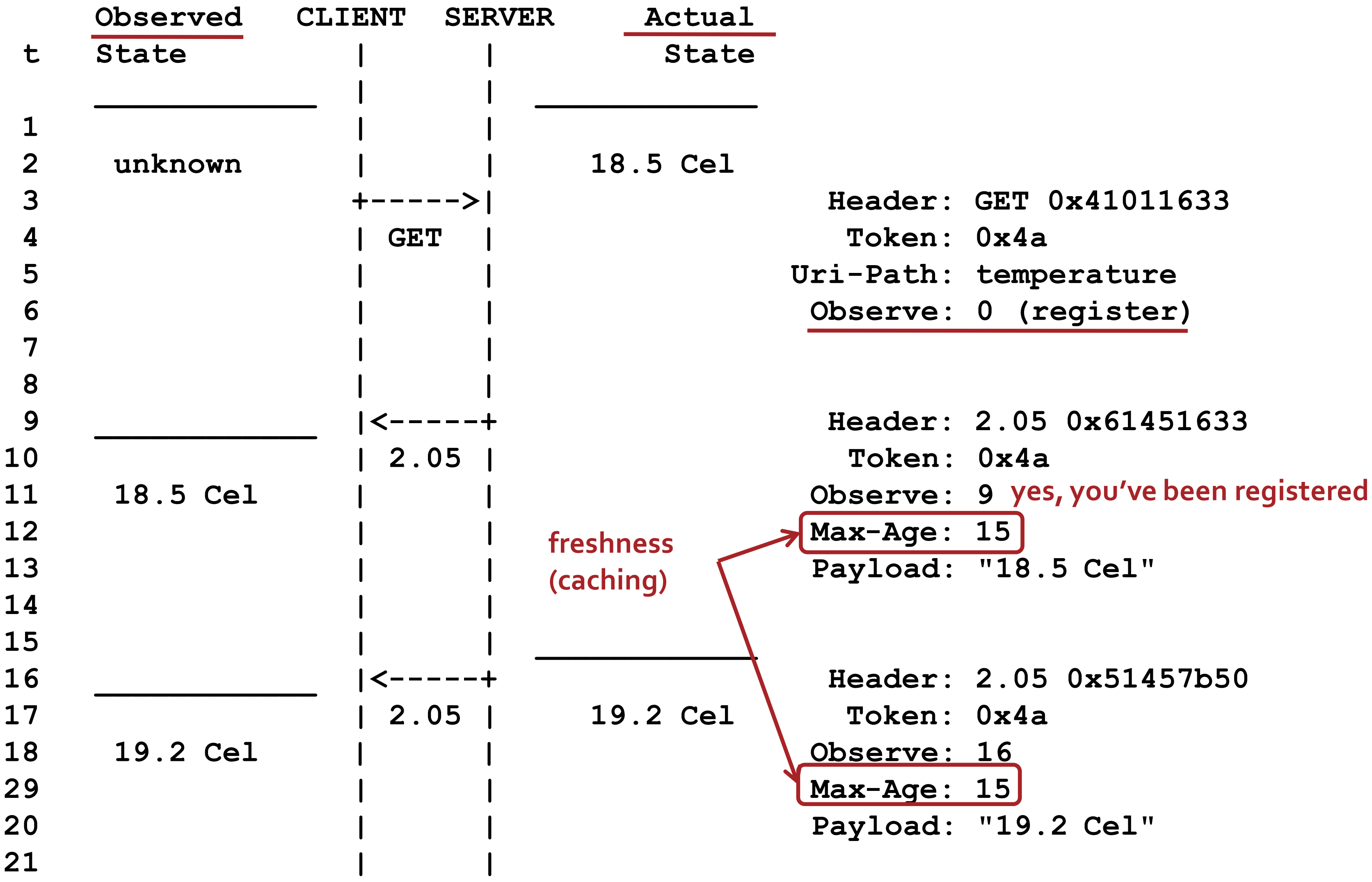
- the approach in CoAP
  - best effort
  - notifications are labeled with maximum duration
  - the **eventual consistency** model

Co to znaczy, że stan zasobu się zmienił?

- the server decides what it means for a resource to change its state (in other words, how to expose an observable resource in a useful way)

- consider temperature (a temperature sensor):
  - **<coap://server/temperature>**
    - changes its state every few seconds to a current reading of the sensor
  - **<coap://server/temperature/felt>**
    - changes its state to "COLD" or "WARM", depending on some thresholds
  - **<coap://server/temperature/critical?above=42>**
    - changes its state either every few seconds to the current temperature reading
      if the temperature exceeds the client-specified threshold, or to "OK" when the reading drops below

OBSERWOWANIE ZASOBU: PRZYKŁAD … (CDN.)



```
    Observed    CLIENT  SERVER     Actual
 t   State         |       |         State
                   |       |
     ____________  |       |  ____________
 1                 |       |
 2    unknown      |       |     18.5 Cel
 3                 +------>|                    Header: GET 0x41011633
 4                 | GET   |                     Token: 0x4a
 5                 |       |                  Uri-Path: temperature
 6                 |       |                   Observe: 0 (register)
 7                 |       |
 8                 |       |
 9   ____________  |<------+                    Header: 2.05 0x61451633
10                 | 2.05  |                     Token: 0x4a
11    18.5 Cel     |       |                   Observe: 9  yes, you’ve been registered
12                 |       |  freshness        Max-Age: 15
13                 |       |  (caching)        Payload: "18.5 Cel"
14                 |       |
15                 |       |  ____________
16   ____________  |<------+                    Header: 2.05 0x51457b50
17                 | 2.05  |     19.2 Cel        Token: 0x4a
18    19.2 Cel     |       |                   Observe: 16
29                 |       |                   Max-Age: 15
20                 |       |                   Payload: "19.2 Cel"
21                 |       |
```

Figure 3: A Client Registers and Receives One Notification of the Current State and One of a New State upon a State Change [RFC7641]

```
        Observed    CLIENT  SERVER     Actual
   t    State         |       |         State
                      |       |
        ____________  |       |  ____________
  22                  |       |
  23     19.2 Cel     |       |    19.2 Cel
  24                  |       |  ____________    a new state
  25     lost notification  | X----+            Header: 2.05 0x51457b51
  26                  | 2.05  |    19.7 Cel       Token: 0x4a
  27                  |       |                 Observe: 25
  28                  |       |                 Max-Age: 15
  29      Max-Age seconds |   |                 Payload: "19.7 Cel"
  30      have passed  |      |
  31      16+15 = 31   |      |
        ____________  |       |
  32                  |       |
  33     19.2 Cel     |       |
  34      (stale)     |       |
  35                  |       |
  36                  |       |
  37                  |       |
  38                  +------>|                  Header: GET 0x41011634
  39                  | GET   |                   Token: 0xb2
  40                  |       |                Uri-Path: temperature
  41                  |       |   re-registration  Observe: 0 (register)
  42                  |       |
  43                  |       |
  44     ____________ |<------+                  Header: 2.05 0x61451634
  45                  | 2.05  |                   Token: 0xb2
  46     19.7 Cel     |       |                 Observe: 44
  47                  |       |                 Max-Age: 15
  48                  |       |  tagging a response  ETag: 0x78797a7a79
  49                  |       |                 Payload: "19.7 Cel"
```

Figure 4: The Client Re-registers after Max-Age Ends [RFC7641]

```
     Observed   CLIENT  SERVER     Actual
 t   State         |       |         State
     ____________  |       |  ____________
51                 |       |
52    19.7 Cel     |       |    19.7 Cel
53                 |       |
54                 |       |  ____________
55                 |     crash
56                 |
57    Max-Age seconds |
58    have passed  |
59    44+15 = 59   |
     ____________
60                 |
61    19.7 Cel     |
62    (stale)      |
63                 |    reboot____________
```

Figure 5: The Client Re-registers and Gives the Server the Opportunity to Select a Stored Response (1/2) [RFC7641]

```
     Observed   CLIENT  SERVER     Actual
 t   State         |      |         State
                   |      |
63                 |    reboot____________
64                 |      |
65                 |      |     20.0 Cel
66                 |      |
67                 +----->|                  Header: GET 0x41011635
68                 | GET  |                   Token: 0xf9
69                 |      |                Uri-Path: temperature
70                 |      |                 Observe: 0 (register)
71                 |      |                    ETag: 0x78797a7a79
72                 |      |
73                 |      |
74     ____________|<-----+                  Header: 2.05 0x61451635
75                 | 2.05 |                   Token: 0xf9
76    20.0 Cel     |      |                 Observe: 74
77                 |      |                 Max-Age: 15
78                 |      |                 Payload: "20.0 Cel"
79                 |      |
80                 |      |   ____________
81     ____________|<-----+                  Header: 2.03 0x5143aa0c
82                 | 2.03 |     19.7 Cel      Token: 0xf9
83    19.7 Cel     |      |                 Observe: 81
84                 |      |                    ETag: 0x78797a7a79
85                 |      |                 Max-Age: 15
86 stored response |      |
```

re-registration is response with this Etag valid?

Content

no, need to send notification with payload

Valid

now it's valid again, can refer to it with ETag

no payload!

stored response

Figure 5: The Client Re-registers and Gives the Server the Opportunity to Select a Stored Response (2/2) [RFC7641]

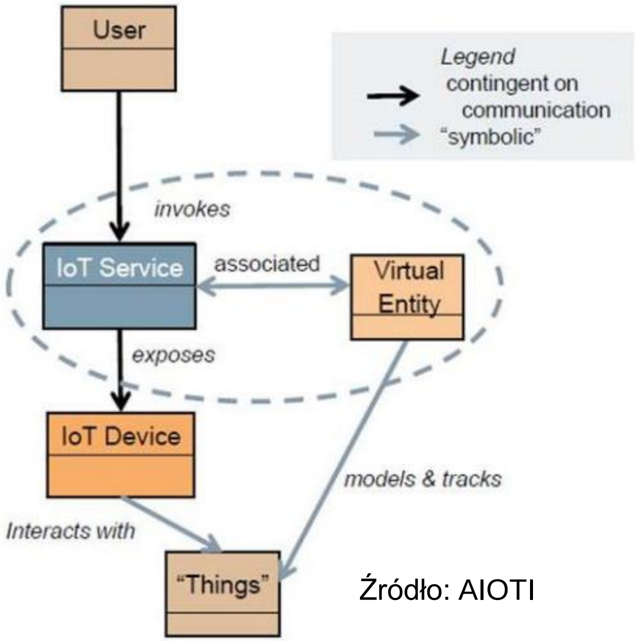
```
        Observed    CLIENT  SERVER       Actual
  t     State          |       |          State
                       |       |
     ____________      |       |    ____________
 87                    |       |
 88     19.7 Cel       |       |      19.7 Cel
 89                    |       |
 90                    |       |    ____________     regular notification
 91                    |<-----+                       Header: 2.05 0x4145aa0f
 92  ____________      | 2.05  |      19.3 Cel          Token: 0xf9
 93     19.3 Cel       |       |                     Observe: 91
 94                    |       |                     Max-Age: 15
 95                    |       |                     Payload: "19.3 Cel"
 96                    |       |   rejecting a notification and thus
 97                    |       |   canceling the observation
 98                    +- - ->|                       Header: 0x7000aa0f
 99                    |       |                      0111 0000 0000 0000
100                    |       |                      RST
101                    |       |                      client dropped from list  of observers
102                    |       |
103                    |       |    ____________
104                    |       |      19.0 Cel
105                    |       |
106                    |       |
107  ____________      |       |
108     19.3 Cel       |       |
109     (stale)        |       |
110                    |       |
```

Figure 6: The Client Rejects a Notification and Thereby Cancels the Observation [RFC7641]

CoRE Link Format

how to discover resources hosted by a server

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



Źródło: AIOTI

WEB LINKING

- a means of indicating the relationships between Web resources
- **link**, **typed link** = a typed connection between two resources

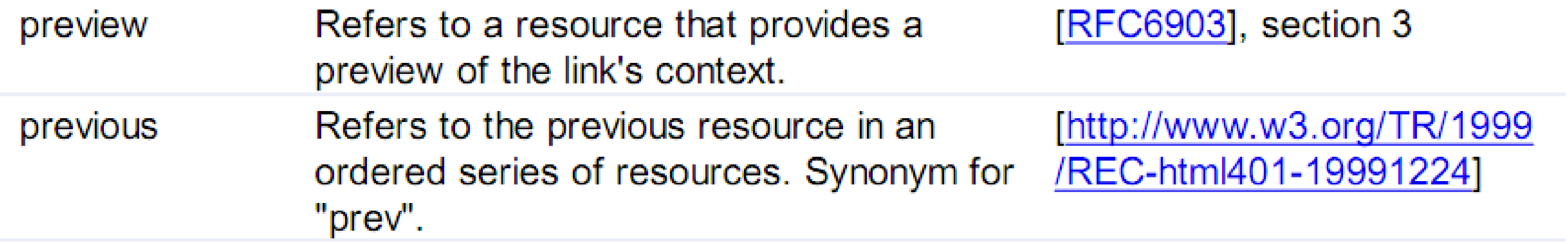
- a typed link consists of
  - a **context URI** (by default, the requested resource)
  - a **link relation type** (semantics of a link: how the two resources are related)
  - a **target URI**, and
  - optionally, **target attributes** (key/value pairs).

- a link is the following statement:
  "{context URI} has a {relation type} with resource at {target URI}, which has {target attributes}"

- HTTP Link header is a serialization of typed links

WEB LINKING: UŻYCIE „ZWYKŁE”

- link relation types registry
  - http://www.iana.org/assignments/link-relations/link-relations.xhtml
  - excerpt:



preview	Refers to a resource that provides a preview of the link's context.	[RFC6903], section 3
previous	Refers to the previous resource in an ordered series of resources. Synonym for "prev".	[http://www.w3.org/TR/1999/REC-html401-19991224]

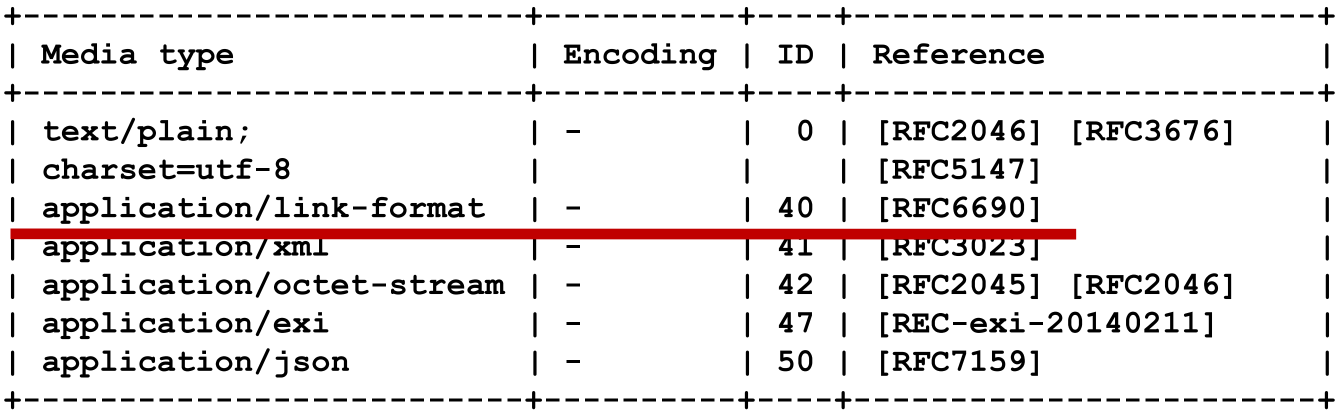
- more examples of link relations:
  - **`describedby`** refers to a resource providing information about the link's context
  - **`preview`** refers to a resource that provides a preview of the link's context
  - **`start`** refers to the first resource in a collection of resources
  - **`next`** indicates that the link's context is a part of a series, and that the next in the series is the link target
  - **`copyright`** refers to a copyright statement that applies to the link's context

WEB LINKING IN COAP: CORE RESOURCE DISCOVERY

- a default URI to discover resources hosted by a constrained server: **/.well-known/core**

- the resource at /.well-known/core contains typed links

- the resource at /.well-known/core is serialized using the **CoRE Link Format**
  - carried as payload (in HTTP this is a header)
  - assigned an Internet media type: **application/link-format**



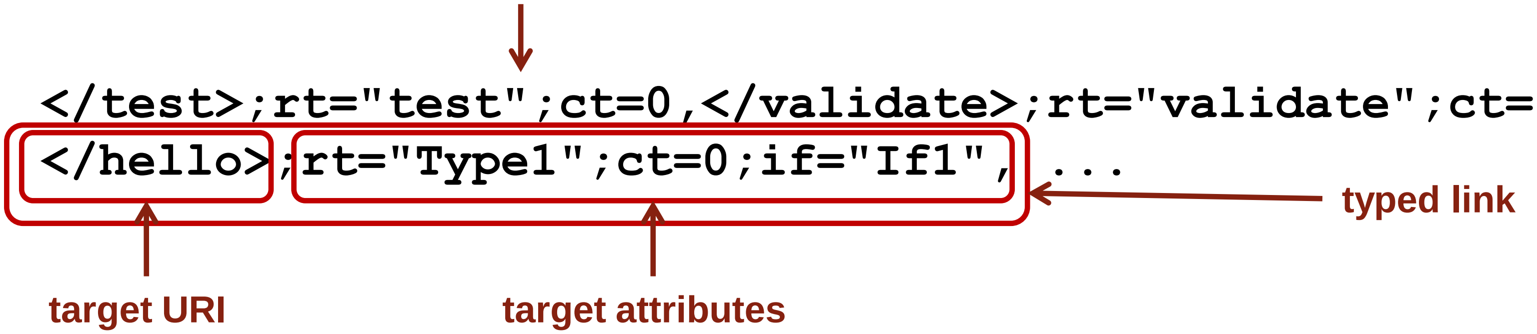
Media type	Encoding	ID	Reference
text/plain; charset=utf-8	-	0	[RFC2046] [RFC3676] [RFC5147]
application/link-format	-	40	[RFC6690]
application/xml	-	41	[RFC3023]
application/octet-stream	-	42	[RFC2045] [RFC2046]
application/exi	-	47	[REC-exi-20140211]
application/json	-	50	[RFC7159]

Table 9: CoAP Content-Formats [RFC7252]

Typed link in CoRE

- in CoRE, a typed link consists of
  - a context URI (by default: the constrained server)
  - a link relation type (by default: ”hosts”)
    - ”hosts” indicates that the target resource is hosted by the constrained server
  - a target URI (the URI of a resource hosted by the constrained server)
  - target attributes (key/value pairs)

a part of the payload received from coap://coap.me:5683 in response to `GET /.well_known/core` (three typed links shown)



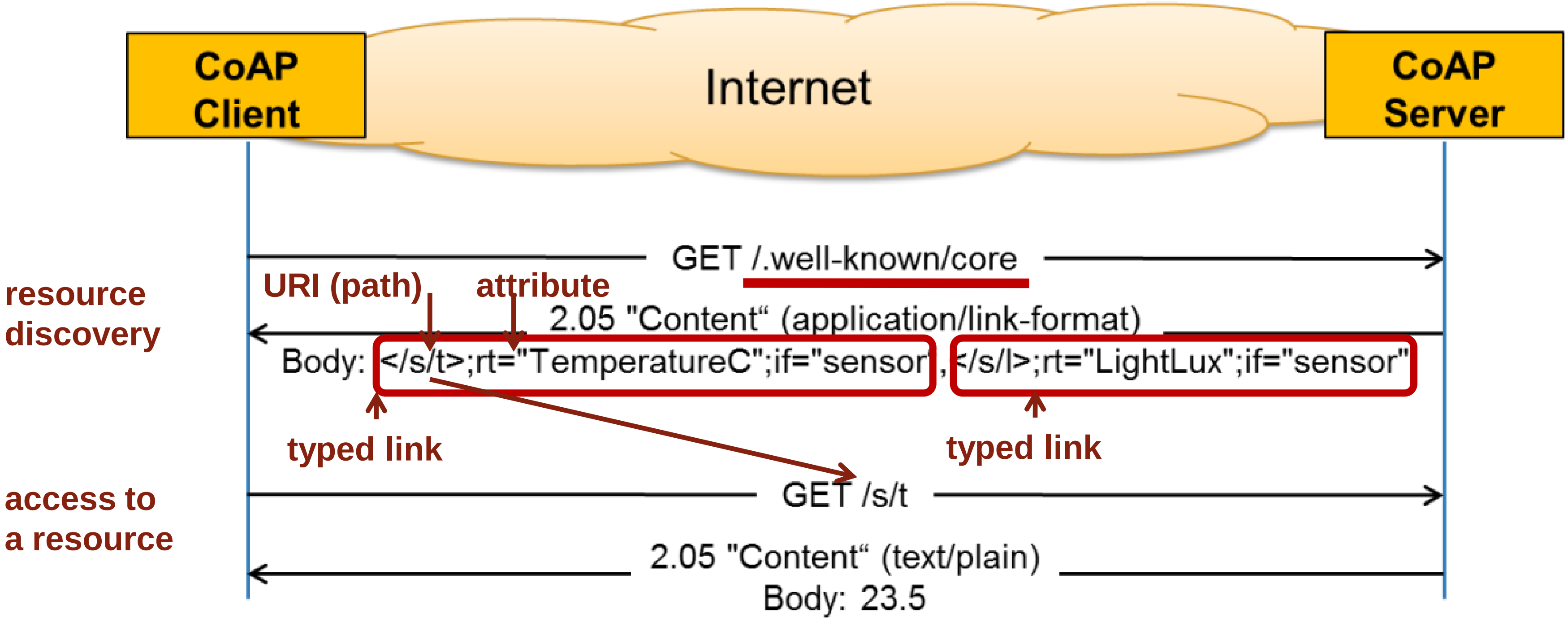
```
</test>;rt="test";ct=0,</validate>;rt="validate";ct=0,
</hello>;rt="Type1";ct=0;if="If1",...
```

**the typed link says: „This server hosts a resource with the URI path `/hello`.
The resource is characterized by the following values of the attributes rt, ct, and if.”**

TARGET ATTRIBUTES

- **Resource Type**, **rt**: application-specific semantic type of a resource
  - example: outdoor-temperature
  - example: http://sweet.jpl.nasa.gov/2.0/phys.owl#Temperature
- **Interface Description**, **if**: describes the REST interface to interact with a resource
  - example: sensor
  - example: http://www.example.org/myapp.wadl#sensor
- **Maximum Size Estimate**, **sz**: an indication of the maximum size of the resource representation returned by performing a GET
- **Content type**, **ct**: a hint about the Content-Format this resource returns
- **Observable**, **obs**: a hint indicating that the resource is useful for observation (not a promise that the Observe Option can be used)

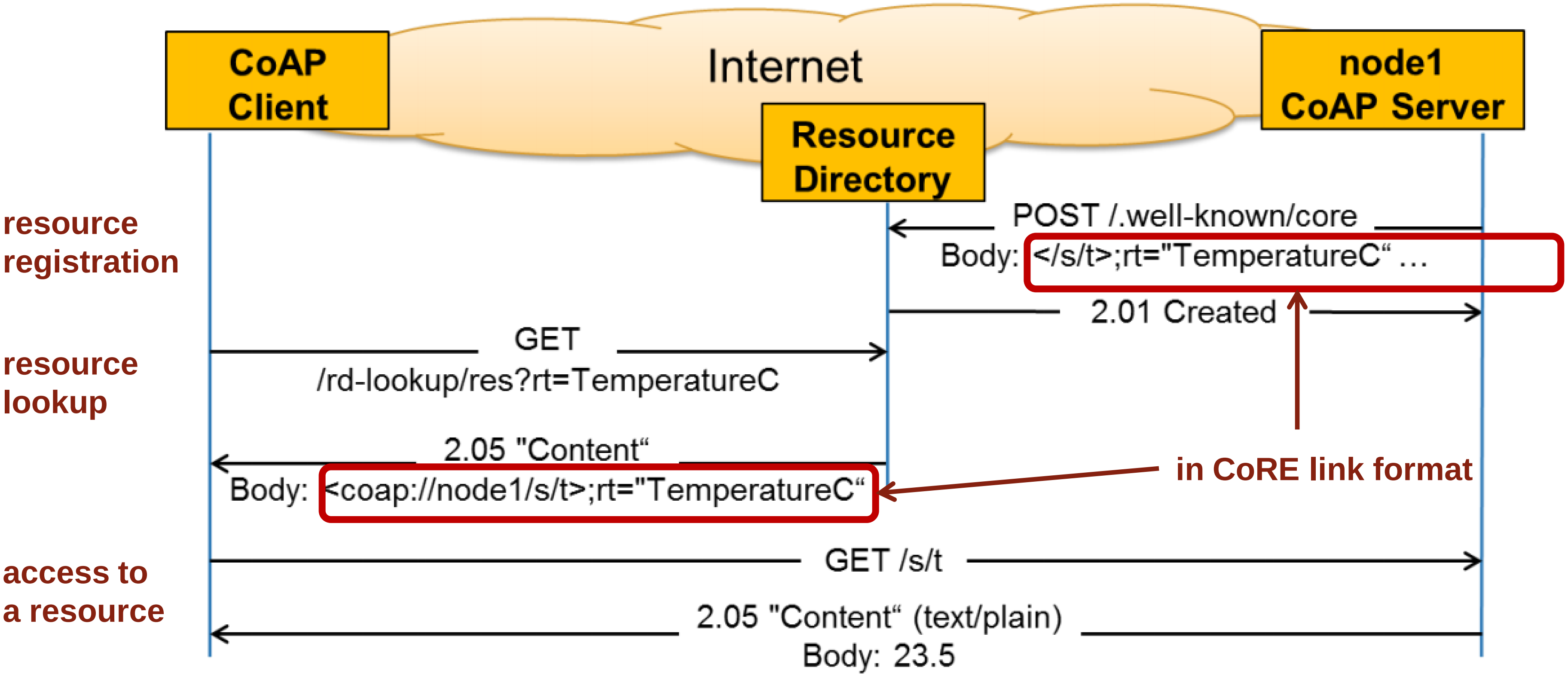
CoRE Resource Discovery: przykład



Źródło: *Flexible Unicast-Based Group Communication for CoAP-Enabled Devices*
I.Ishaq et al., Sensors, 2014, 14

CoRE Resource Directory

katalog zasobów



Źródło: *Flexible Unicast-Based Group Communication for CoAP-Enabled Devices*
I.Ishaq et al., Sensors, 2014, 14

- See [draft-ietf-core-resource-directory-07] Z. Shelby et al. "CoRE Resource Directory", March 2016

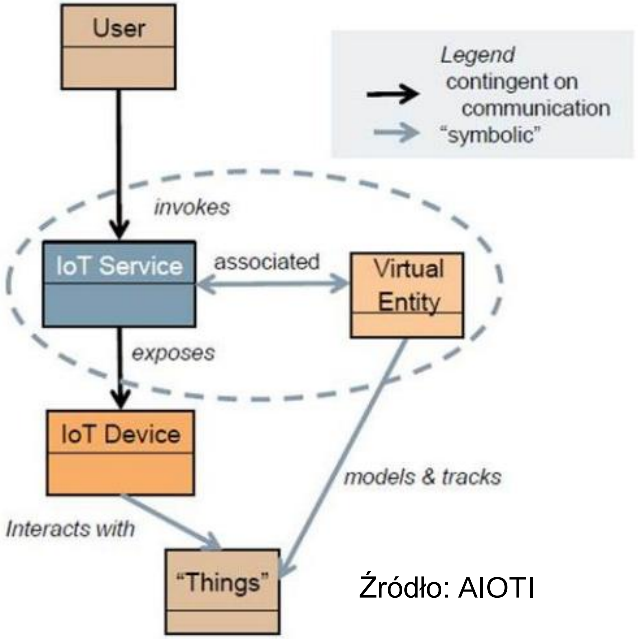
KILKA PYTAŃ O ODKRYWANIU ZASOBÓW

- how to identify resource types (rt=) and describe interfaces (if=)?
- how to ensure that the client "understands" these attributes?
- aside: what does it mean to "understand"?
- the resource discovery mechanism described so far may not be enough
- answer: **semantics**

Block-Wise Transfers in CoAP

how to transfer big resource representations

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

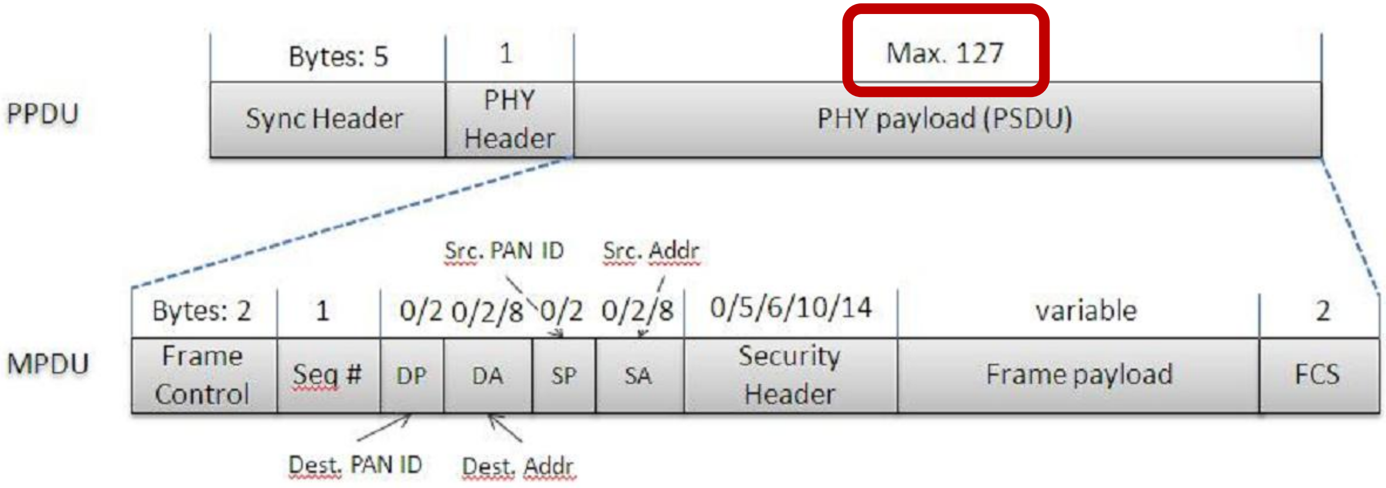


Źródło: AIOTI

TRANSFER BLOKOWY

- Problem?

Figure 3. IEEE 802.15.4 The *Physical Protocol Data Unit* (PPDU) and *MAC Protocol Data Unit* (MPDU) formats.



Źródło: *IETF Standardization in the Field of the Internet of Things (IoT): A Survey*
Isam Ishaq et al., J. Sens. Actuator Netw. 2013, 2, 235-287

- Cel: uniknąć fragmentacji w niższych warstwach.

Opcje Block1, Block2

	No.	C	U	N	R	Name	Format	Length	Default
GET (response payload)	23	C	U	-	-	Block2	uint	0-3	(none)
POST, PUT (request payload)	27	C	U	-	-	Block1	uint	0-3	(none)

Table 1: Block Option Numbers [RFC7959]

Opcje Block1, Block2: wartości opcji

```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|  NUM  |M| SZX |
+-+-+-+-+-+-+-+-+
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          NUM          |M| SZX |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 0                   1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   NUM                 |M| SZX |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     Figure 1: Block Option Value [RFC7959]
```

SZX "exponent" for the size of the block

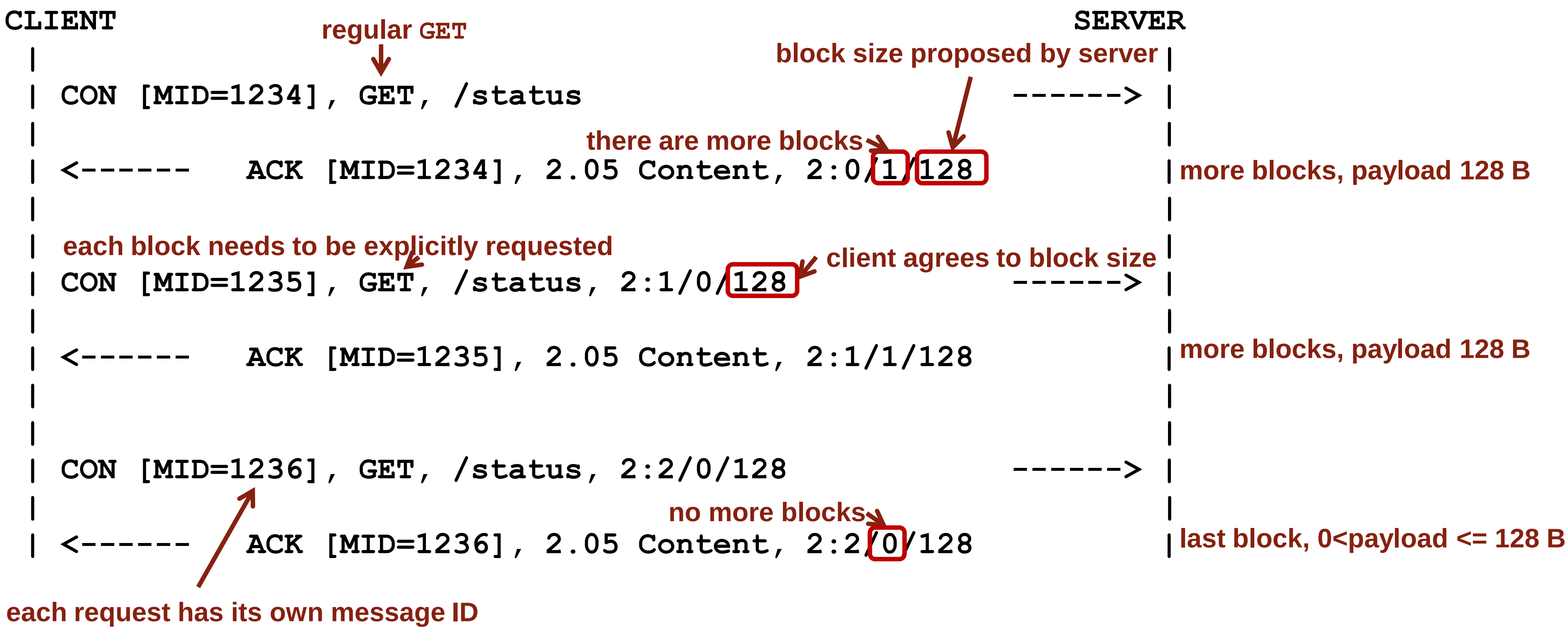
M are there more blocks?

NUM block sequence number (requested or provided)

- size of the block = 2**(SZX + 4)
  - SZX = 1, 2, …, 6 (7 is reserved)
  - size = 16, 32, …, 1024
- in examples, option **`Block<n>`** is shown as n: NUM/M/<size>

OPCJA BLOCK2: PRZYKŁAD (SIMPLE BLOCKWISE GET)

Block2 – has to do with response payload (transferred from the server to the client)



```
CLIENT                                                     SERVER
  |                                                          |
  | CON [MID=1234], GET, /status                     ------> |
  |                                                          |
  | <------   ACK [MID=1234], 2.05 Content, 2:0/1/128        |
  |                                                          |
  |                                                          |
  | CON [MID=1235], GET, /status, 2:1/0/128          ------> |
  |                                                          |
  | <------   ACK [MID=1235], 2.05 Content, 2:1/1/128        |
  |                                                          |
  |                                                          |
  | CON [MID=1236], GET, /status, 2:2/0/128          ------> |
  |                                                          |
  | <------   ACK [MID=1236], 2.05 Content, 2:2/0/128        |
```

regular GET
block size proposed by server
there are more blocks
more blocks, payload 128 B
each block needs to be explicitly requested
client agrees to block size
more blocks, payload 128 B
no more blocks
last block, 0<payload <= 128 B
each request has its own message ID

Figure 2: Simple Block-Wise GET [RFC7959]

OPCJA BLOCK2: PRZYKŁAD (EARLY NEGOTIATION)

client anticipates a blockwise transfer and sends a block size proposal

control usage of a block option: how should the transfer proceed

descriptive usage: info on the actual payload

```
CLIENT                                                     SERVER
  |                                                          |
  | CON [MID=1234], GET, /status, 2:0/0/64           ------> |
  |                                                          |
  | <------   ACK [MID=1234], 2.05 Content, 2:0/1/64         |
  |                                                          |
  | CON [MID=1235], GET, /status, 2:1/0/64           ------> |
  |                                                          |
  | <------   ACK [MID=1235], 2.05 Content, 2:1/1/64         |
  :                                                          :
  :                          ...                             :
  :                                                          :
  | CON [MID=1238], GET, /status, 2:4/0/64           ------> |
  |                                                          |
  | <------   ACK [MID=1238], 2.05 Content, 2:4/1/64         |
  |                                                          |
  | CON [MID=1239], GET, /status, 2:5/0/64           ------> |
  |                                                          |
  | <------   ACK [MID=1239], 2.05 Content, 2:5/0/64         |

   Figure 3: Block-Wise GET with Early Negotiation [RFC7959]
```

payload 64 B

payload 64 B

payload 64 B

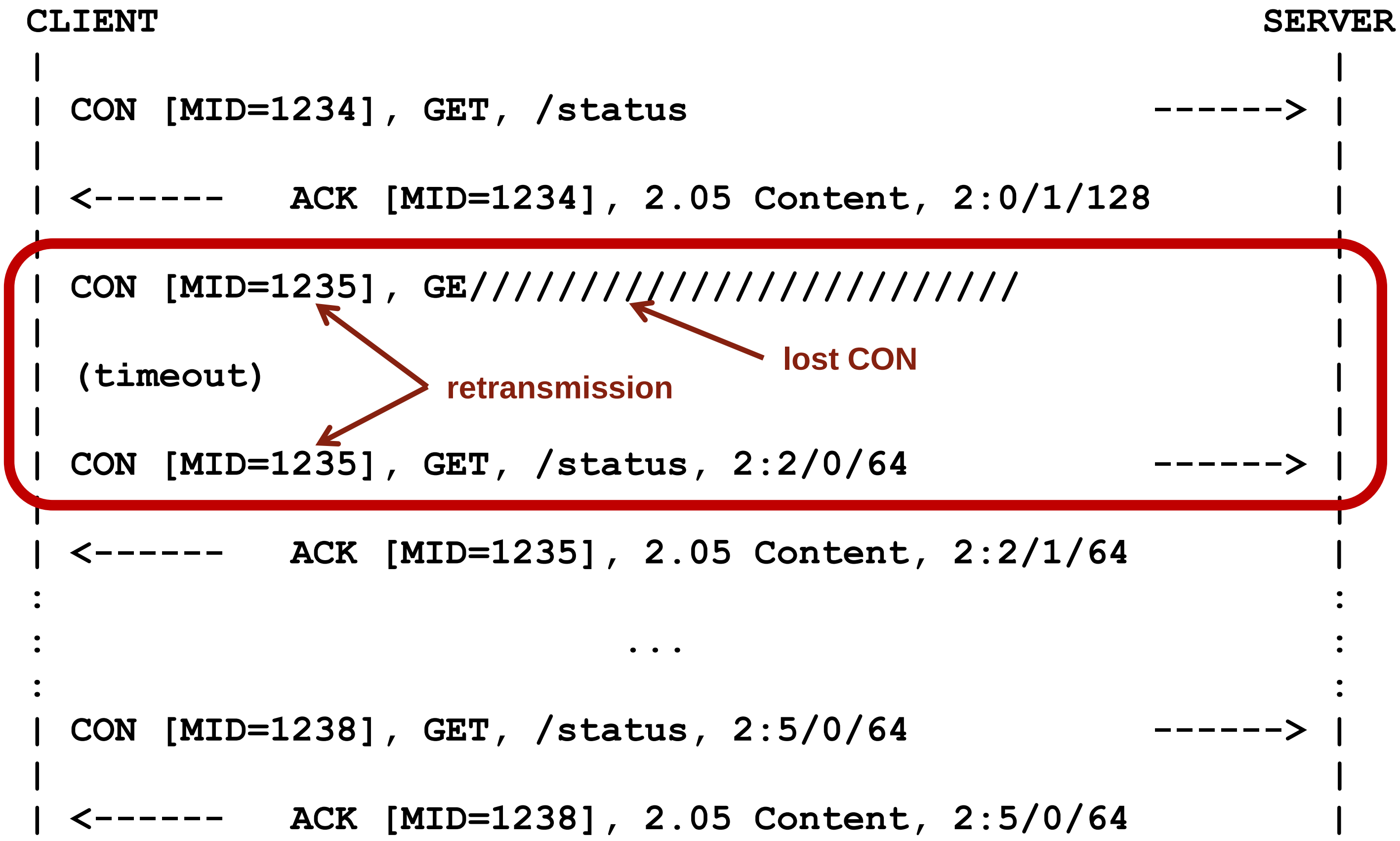
0 < payload <= 64 B

Opcja Block2: przykład (late negotiation)

```
CLIENT                                                       SERVER
  |                                                            |
  | CON [MID=1234], GET, /status                       ------> |
  |                                 block size proposed by server
  |                                                            |
  | <------   ACK [MID=1234], 2.05 Content, 2:0/1/128          |  payload 128 B
  | blocks 0 and 1 (of size 64) have already been transferred
  |                                                            |
  | CON [MID=1235], GET, /status, 2:2/0/64  client prefers less ------> |
  |                                                            |
  | <------   ACK [MID=1235], 2.05 Content, 2:2/1/64           |  payload 64 B
  |                                                            |
  | CON [MID=1236], GET, /status, 2:3/0/64             ------> |
  |                                                            |
  | <------   ACK [MID=1236], 2.05 Content, 2:3/1/64           |  payload 64 B
  |                                                            |
  | CON [MID=1237], GET, /status, 2:4/0/64             ------> |
  |                                                            |
  | <------   ACK [MID=1237], 2.05 Content, 2:4/1/64           |  payload 64 B
  |                                                            |
  | CON [MID=1238], GET, /status, 2:5/0/64             ------> |
  |                                                            |
  | <------   ACK [MID=1238], 2.05 Content, 2:5/0/64           |  0 < payload <= 64 B

   Figure 4: Block-Wise GET with Late Negotiation [RFC7959]
```

OPCJA BLOCK2: PRZYKŁAD (LOST CON)

- Retransmisje bez zmian

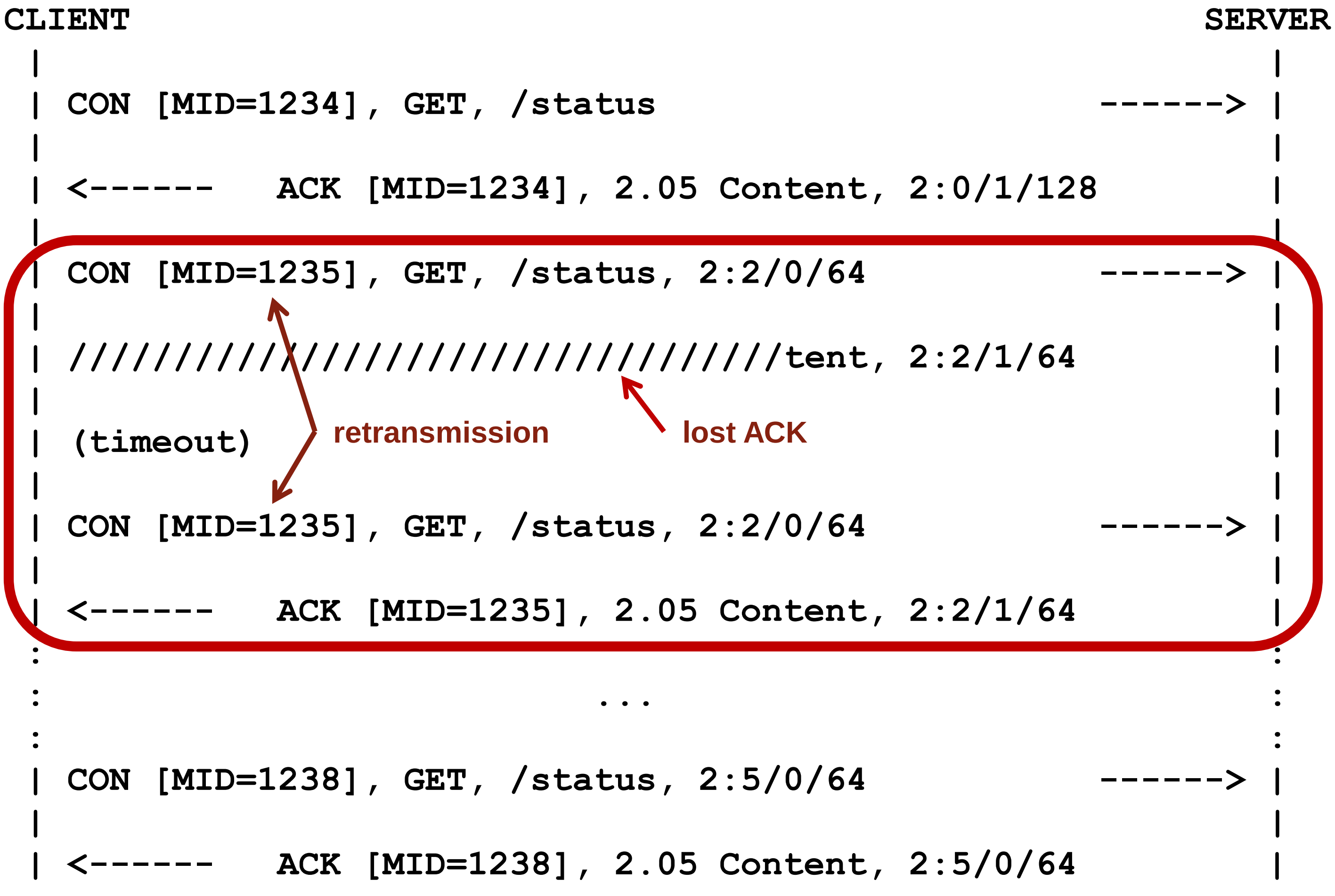


```
CLIENT                                                     SERVER
  |                                                          |
  | CON [MID=1234], GET, /status                     ------> |
  |                                                          |
  | <------   ACK [MID=1234], 2.05 Content, 2:0/1/128        |
  |                                                          |
  | CON [MID=1235], GE/////////////////////////              |
  |                                                          |
  | (timeout)                                                |
  |                                                          |
  | CON [MID=1235], GET, /status, 2:2/0/64           ------> |
  |                                                          |
  | <------   ACK [MID=1235], 2.05 Content, 2:2/1/64         |
  :                                                          :
  :                             ...                          :
  :                                                          :
  | CON [MID=1238], GET, /status, 2:5/0/64           ------> |
  |                                                          |
  | <------   ACK [MID=1238], 2.05 Content, 2:5/0/64         |
```

Figure 5: Block-Wise GET with Late Negotiation and Lost CON [RFC7959]

Opcja Block2: przykład (lost ACK)

- Retransmisje bez zmian



```
CLIENT                                                   SERVER
 |                                                          |
 | CON [MID=1234], GET, /status                     ------> |
 |                                                          |
 | <------   ACK [MID=1234], 2.05 Content, 2:0/1/128        |
 |                                                          |
 | CON [MID=1235], GET, /status, 2:2/0/64           ------> |
 |                                                          |
 | //////////////////////////////////tent, 2:2/1/64         |
 |                                                          |
 | (timeout)                                                |
 |                                                          |
 | CON [MID=1235], GET, /status, 2:2/0/64           ------> |
 |                                                          |
 | <------   ACK [MID=1235], 2.05 Content, 2:2/1/64         |
 :                                                          :
 :                             ...                          :
 :                                                          :
 | CON [MID=1238], GET, /status, 2:5/0/64           ------> |
 |                                                          |
 | <------   ACK [MID=1238], 2.05 Content, 2:5/0/64         |
```

Figure 6: Block-Wise GET with Late Negotiation and Lost ACK

OPCJA BLOCK1: PRZYKŁAD

```
CLIENT                                                     SERVER
|                                                            |
| CON [MID=1234], PUT, /options, 1:0/1/128    ------>        | payload 128 B
|          new response code                                 |
| <------   ACK [MID=1234], 2.31 Continue, 1:0/1/128         |
|                                                            |
| CON [MID=1235], PUT, /options, 1:1/1/128    ------>        | payload 128 B
|                                                            |
| <------   ACK [MID=1235], 2.31 Continue, 1:1/1/128         |
|                                                            |
| CON [MID=1236], PUT, /options, 1:2/0/128    ------>        | 0 < payload <= 128 B
|                                                            |
| <------   ACK [MID=1236], 2.04 Changed, 1:2/0/128          |
```

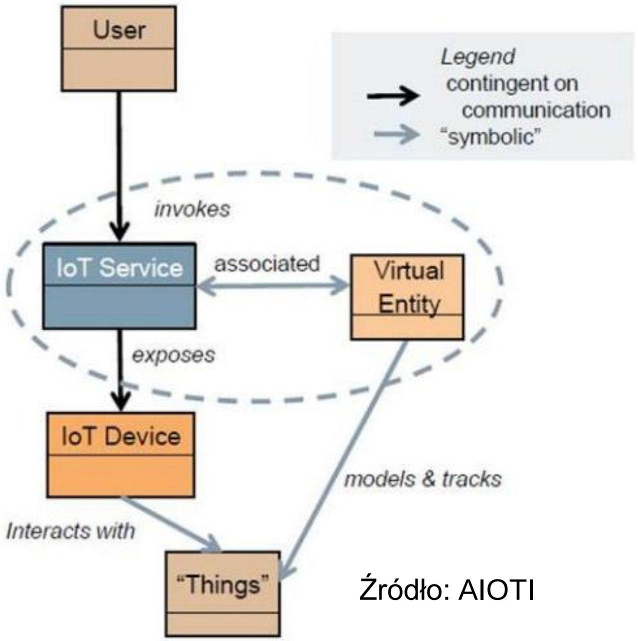
resource updated only after the last block has been received

Figure 7: Simple Atomic Block-Wise PUT [RFC7959]

OPCJE **SIZE1, SIZE2**

- Całkowity rozmiar reprezentacji zasobu
- **`Size1`** – gdy reprezentacja przesyłana w zapytaniach
- **`Size2`** – gdy reprezentacja przesyłana w odpowiedziach

Dziękujemy za uwagę!



Źródło: AIOTI