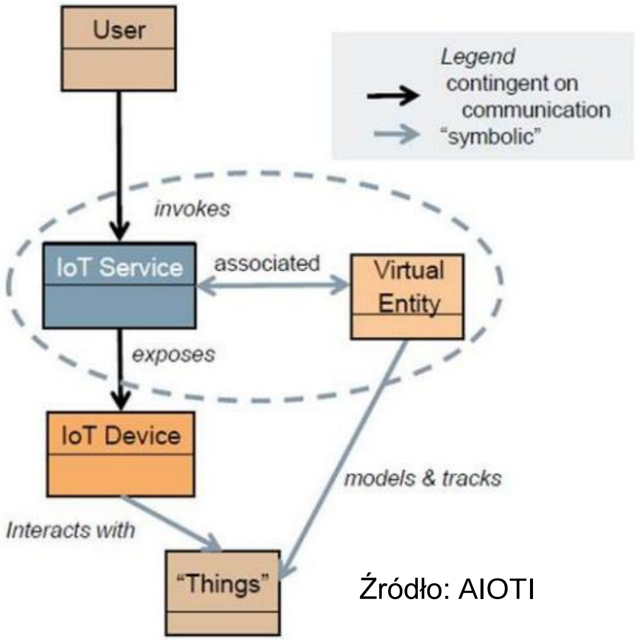
Obiekty Internetu Rzeczy

Wykład 3-4: Constrained Application Protocol (CoAP)

Jarosław Domaszewicz
Instytut Telekomunikacji Politechniki Warszawskiej

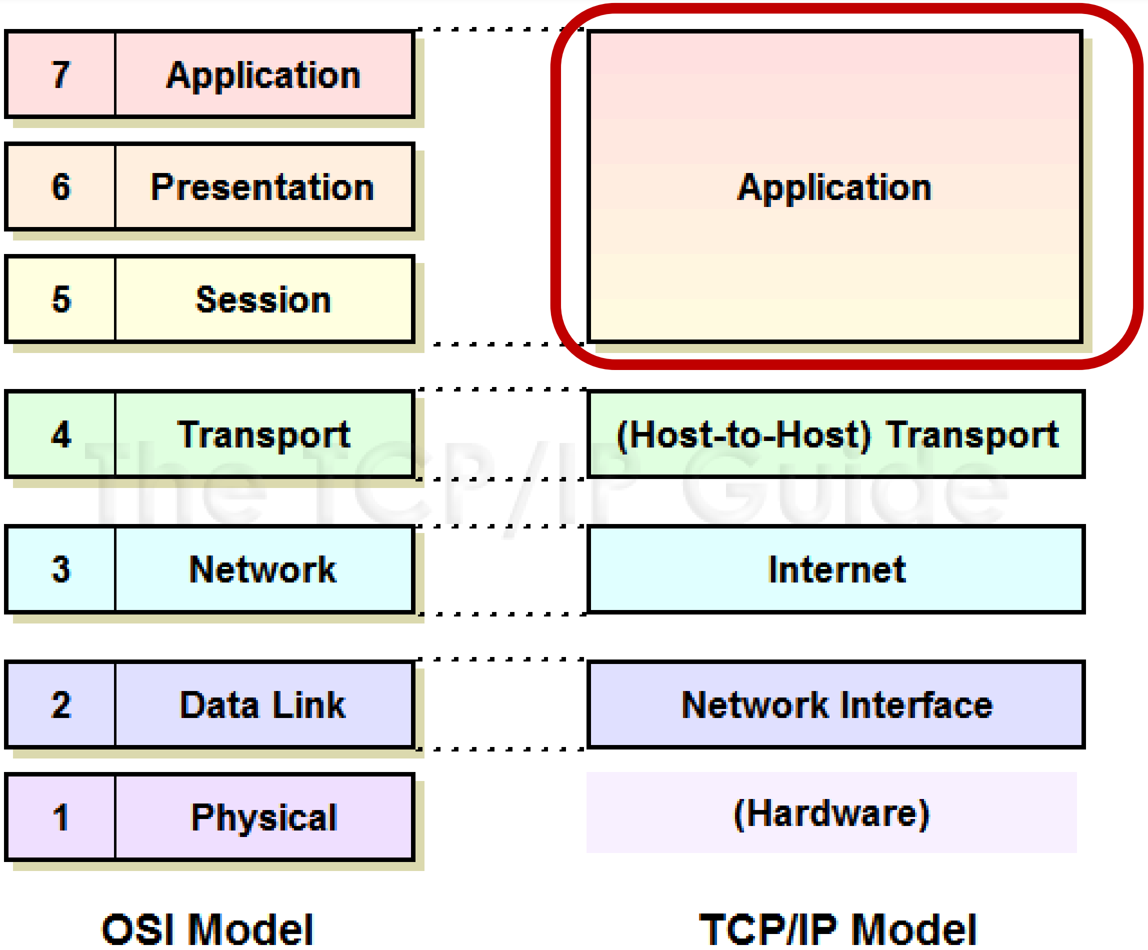


Źródło: AIOTI

Plan wykładu

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

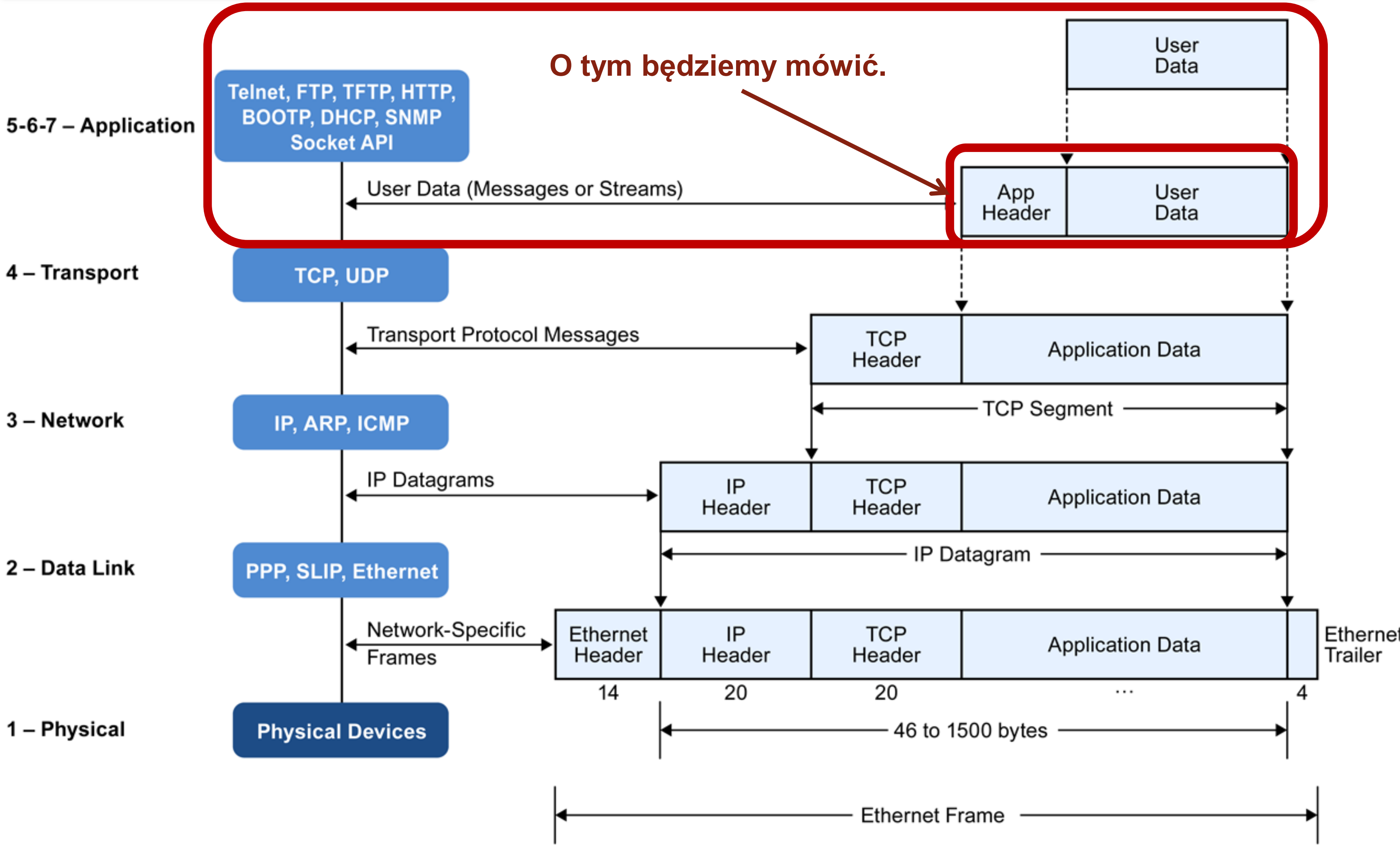
Warstwa aplikacji (1/3)



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	IPv6, IPv4, IPSec
6LowPAN	
IEEE 802.15.4 MAC	Ethernet (IEEE 802.3), DSL, ISDN, Wireless LAN (IEEE 802.11), Wi Fi
IEEE 802.15.4 PHY / Physical Radio	

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

Message Layer (reliability)

Observe option

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?*

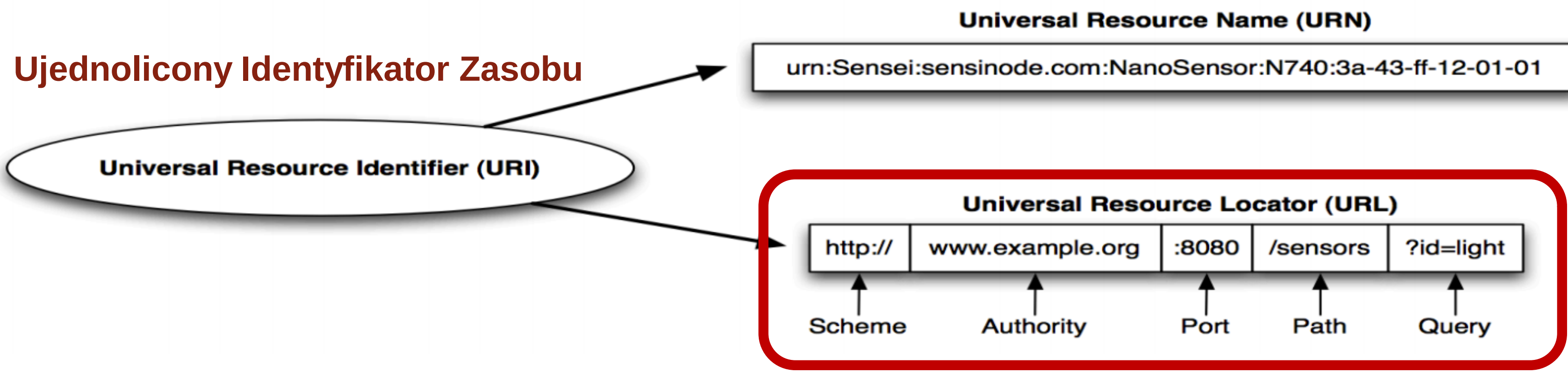
- note: the description below is (somewhat) simplified

- 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<u>(s)</u>
  - 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 <u>resource and application neutral</u>
- 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

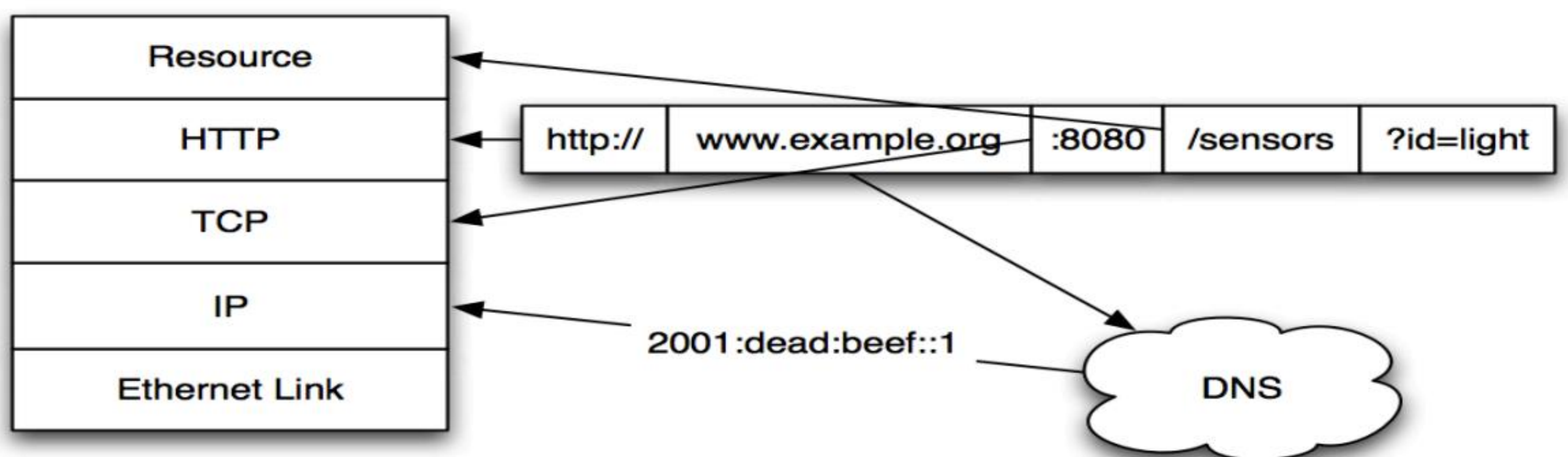
IDENTYFIKATORY URI

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

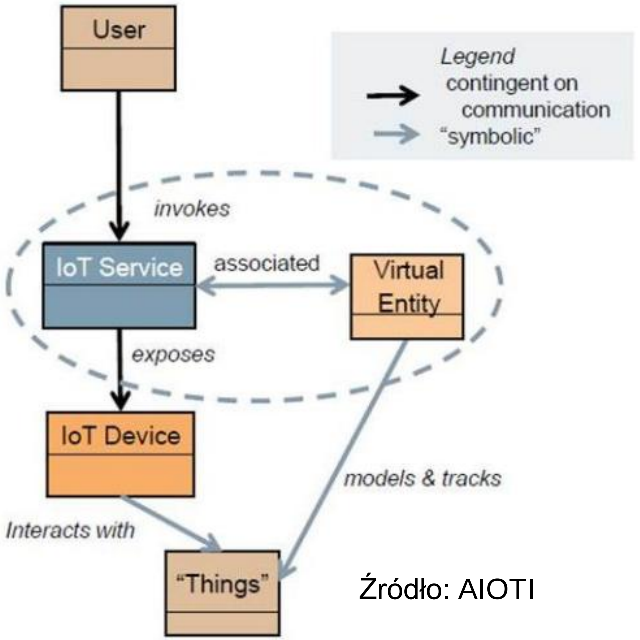
Ujednolicony Identyfikator Zasobu



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

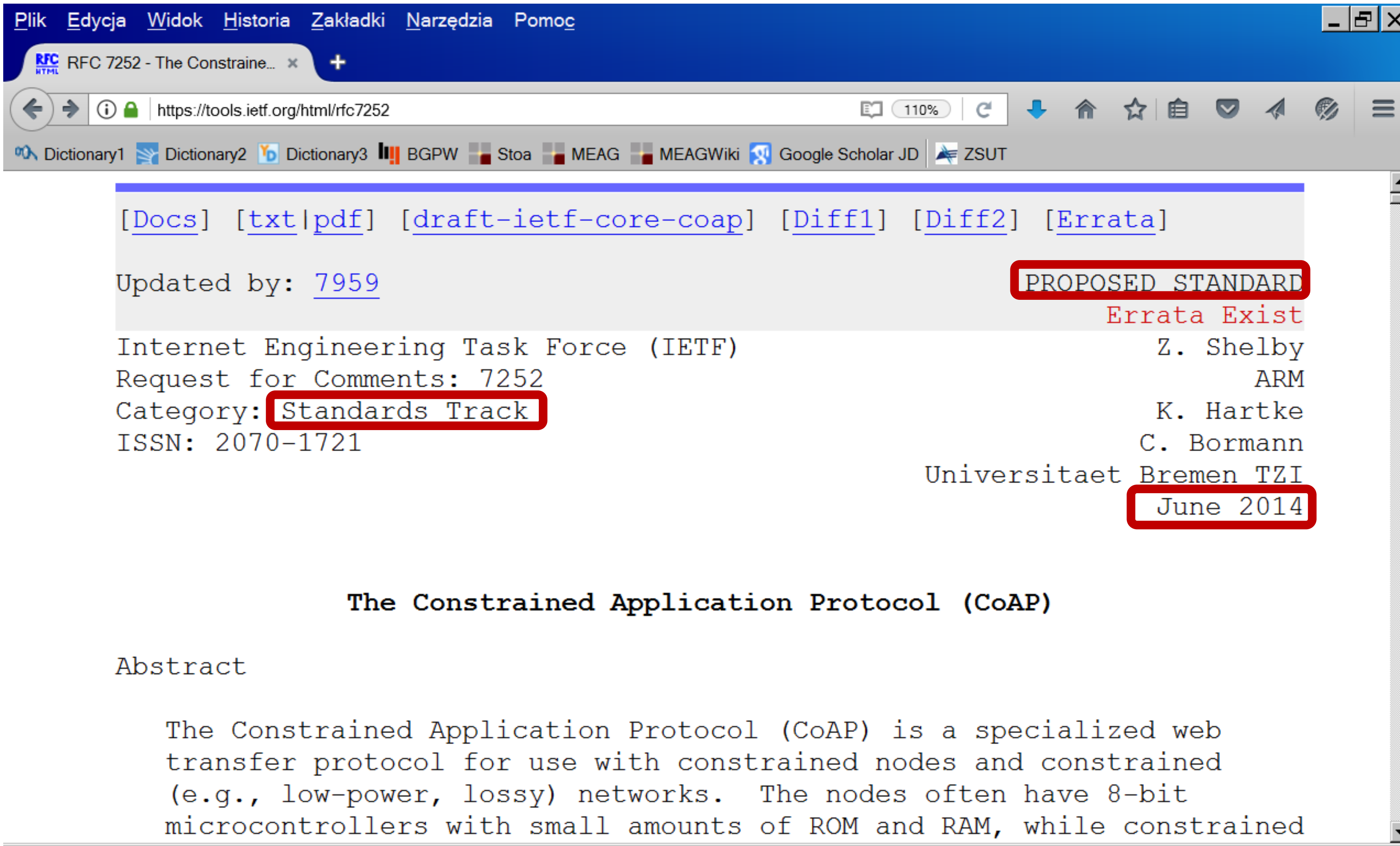
  how to transfer big resource representations

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

  how to discover resources hosted by a server

RFC 7252 (1/2)

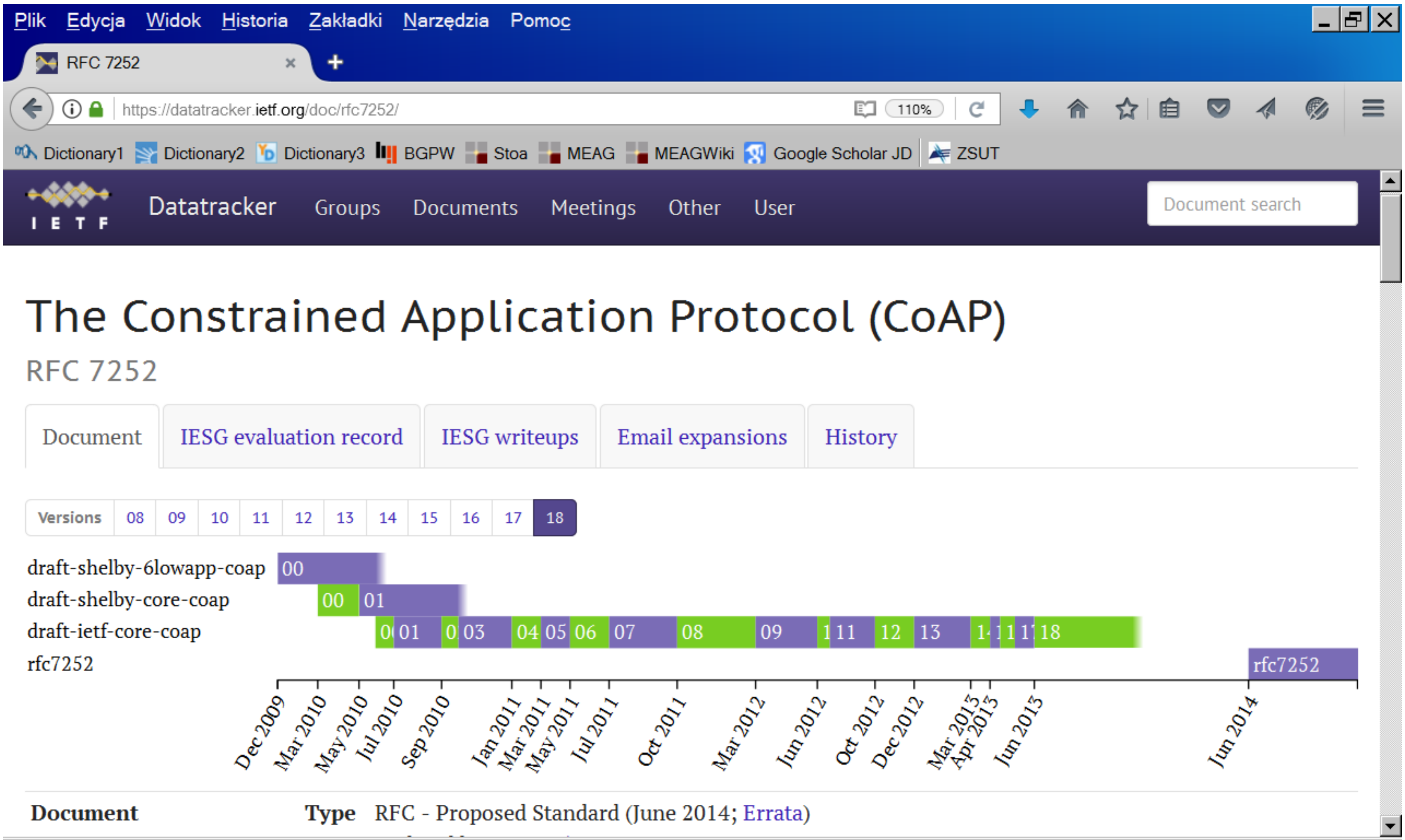
- Dokument: https://tools.ietf.org/html/rfc7252



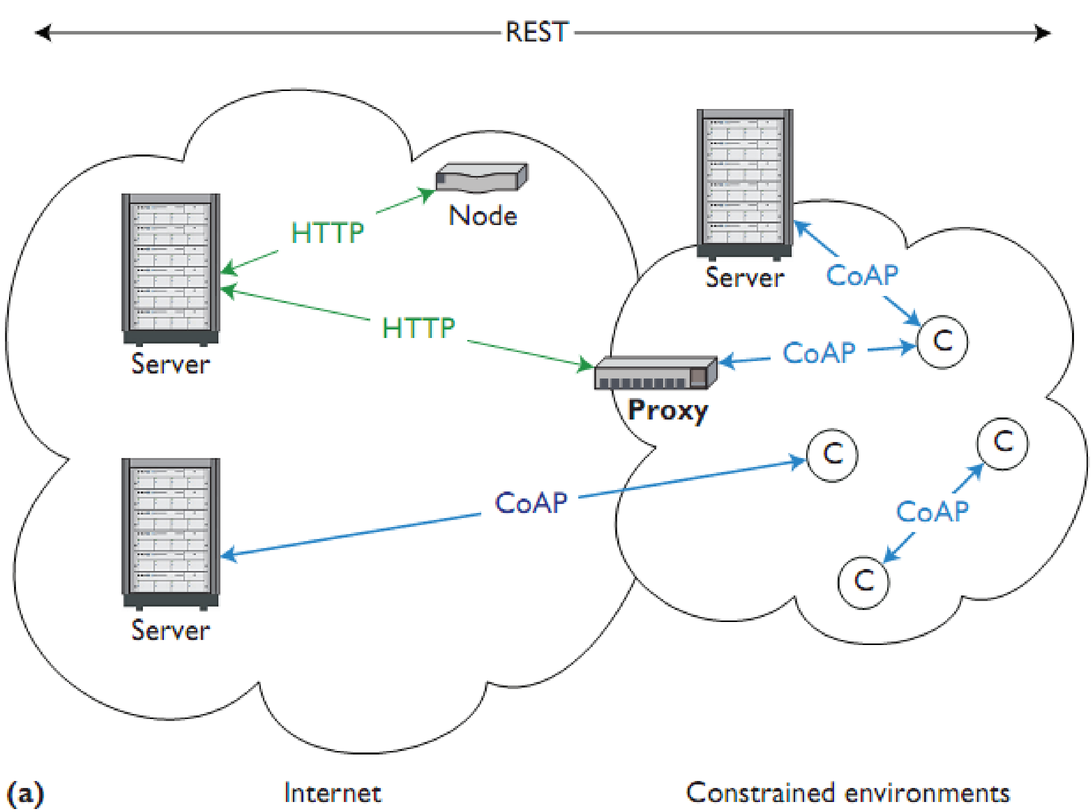

```
[Docs] [txt|pdf] [draft-ietf-core-coap] [Diff1] [Diff2] [Errata]

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
```

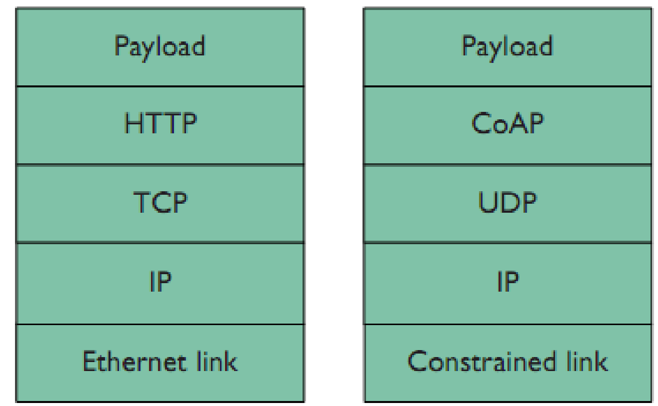
RFC 7252 (2/2)

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



ARCHITEKTURA SYSTEMU COAP





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

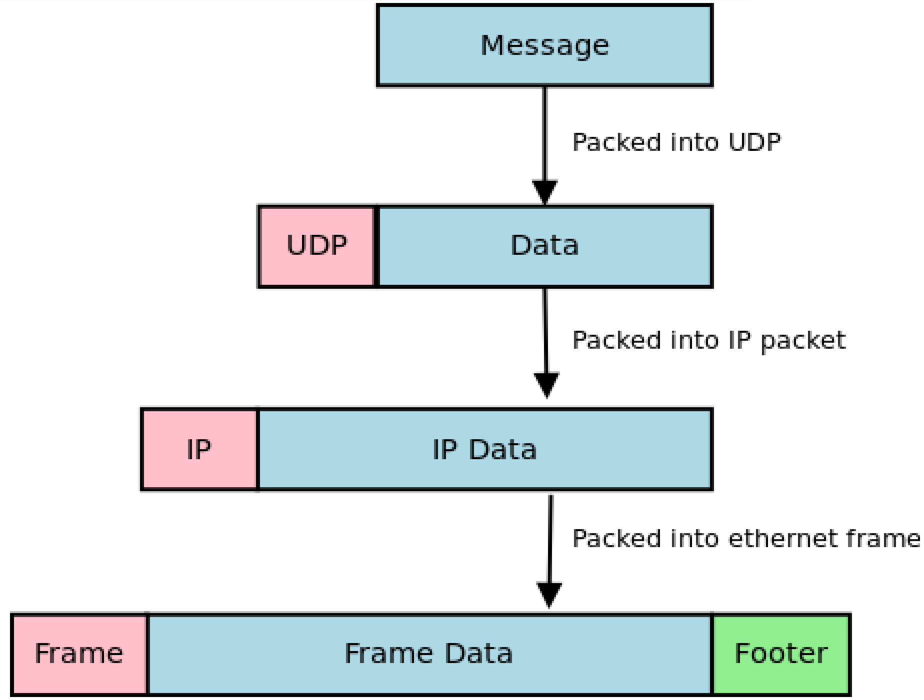
(b)

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]
```

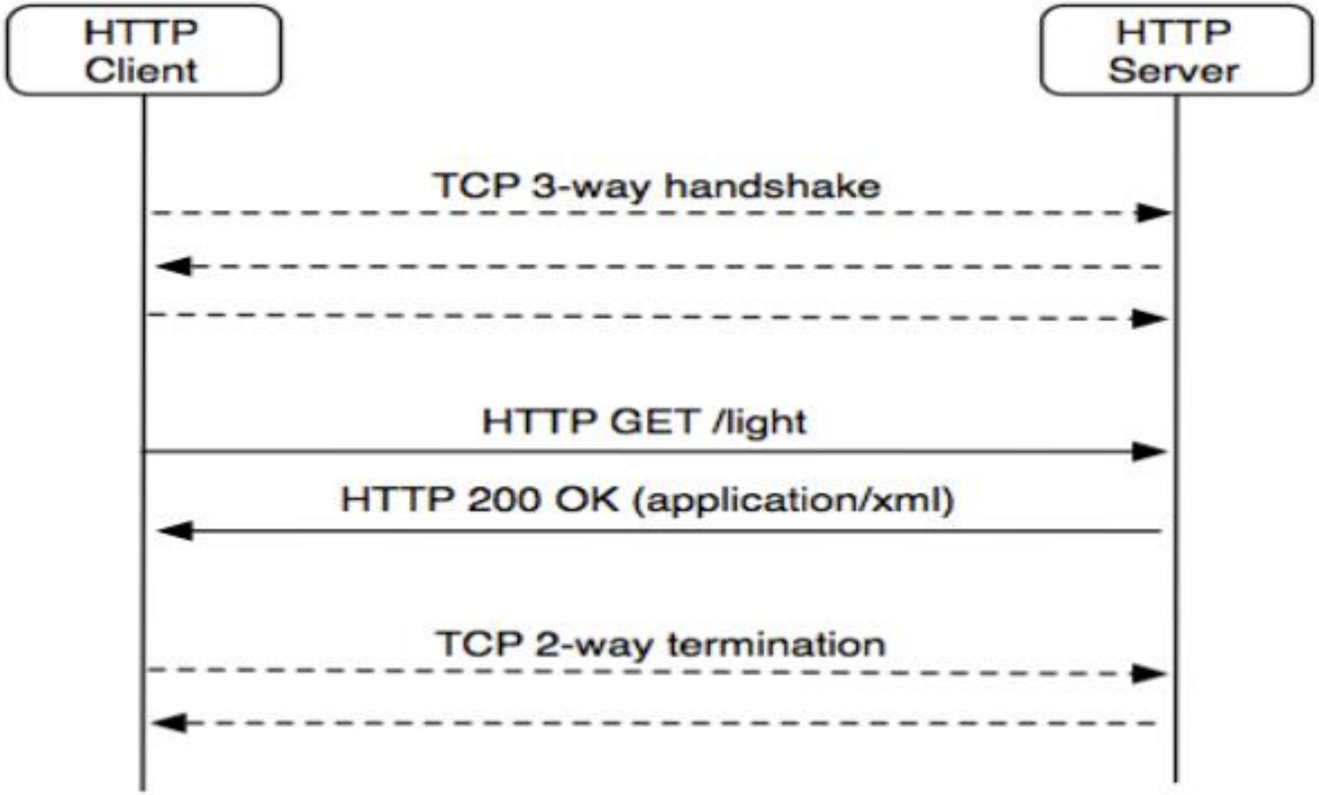
3 strony



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 TCP (RFC 8323) is also possible
- CoAP is <u>not</u> 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: **zapytania/odpowiedzi**
  - 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

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

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) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

to match the response with the request

URI (in the request) goes here

a resource representation (in the response)

Figure 7: Message Format[RFC7252]

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

Header

Token

Options

Payload

Figure 7: Message Format[RFC7252]

to match a response with its request

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

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

message ID

path of resource URI (in an option)

matching token

response code

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

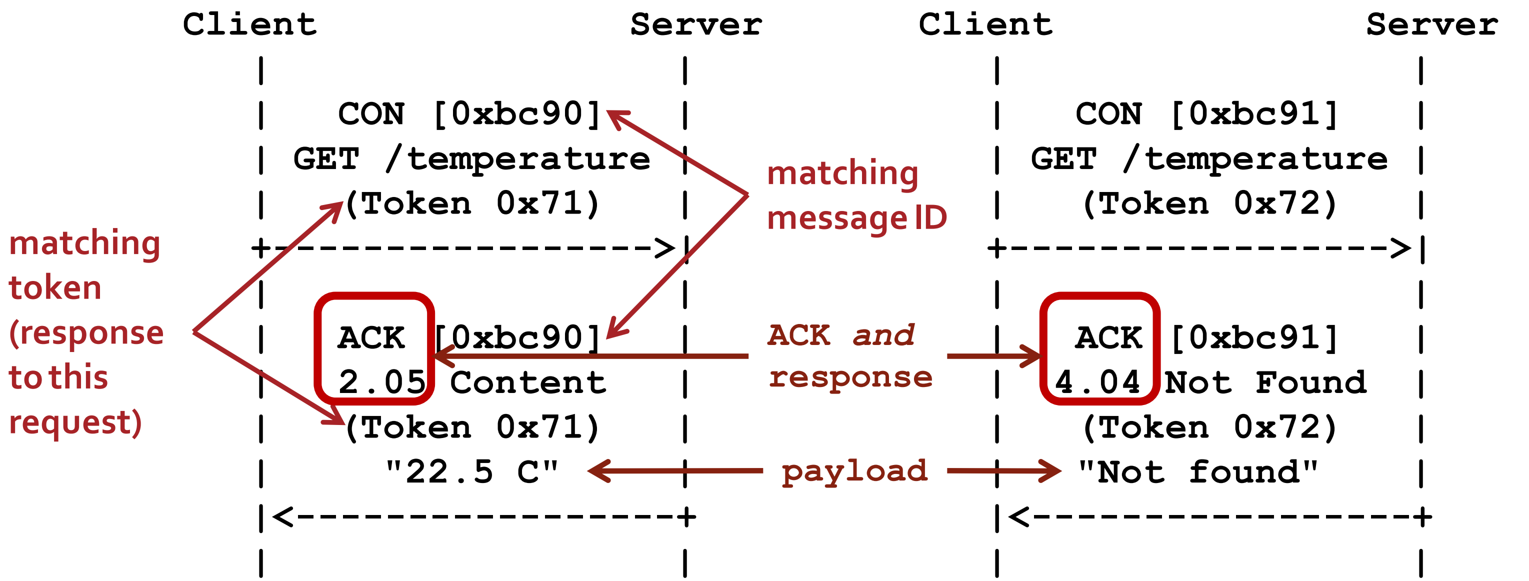


Figure 4: Two GET Requests with Piggybacked Responses[RFC7252]

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

EMPTY ACK AND SEPARATE RESPONSE

empty acknowledgement

separate response (also a CON)

```
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 (say, it takes some time to take a sensor reading)

UŻYCIE WIADOMOŚCI RÓŻNYCH TYPÓW

message ID must be echoed

T field

Code field

piggybacked response

```
+----------+-----+-----+-----+-----+
|          | CON | NON | ACK | RST |
+----------+-----+-----+-----+-----+
| Request  | X   | X   | -   | -   |
| Response | X   | X   | X   | -   |
| Empty    | *   | -   | X   | X   |
+----------+-----+-----+-----+-----+
```

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: 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]

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

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]

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`**

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 current 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  \
+-------------------------------+

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

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

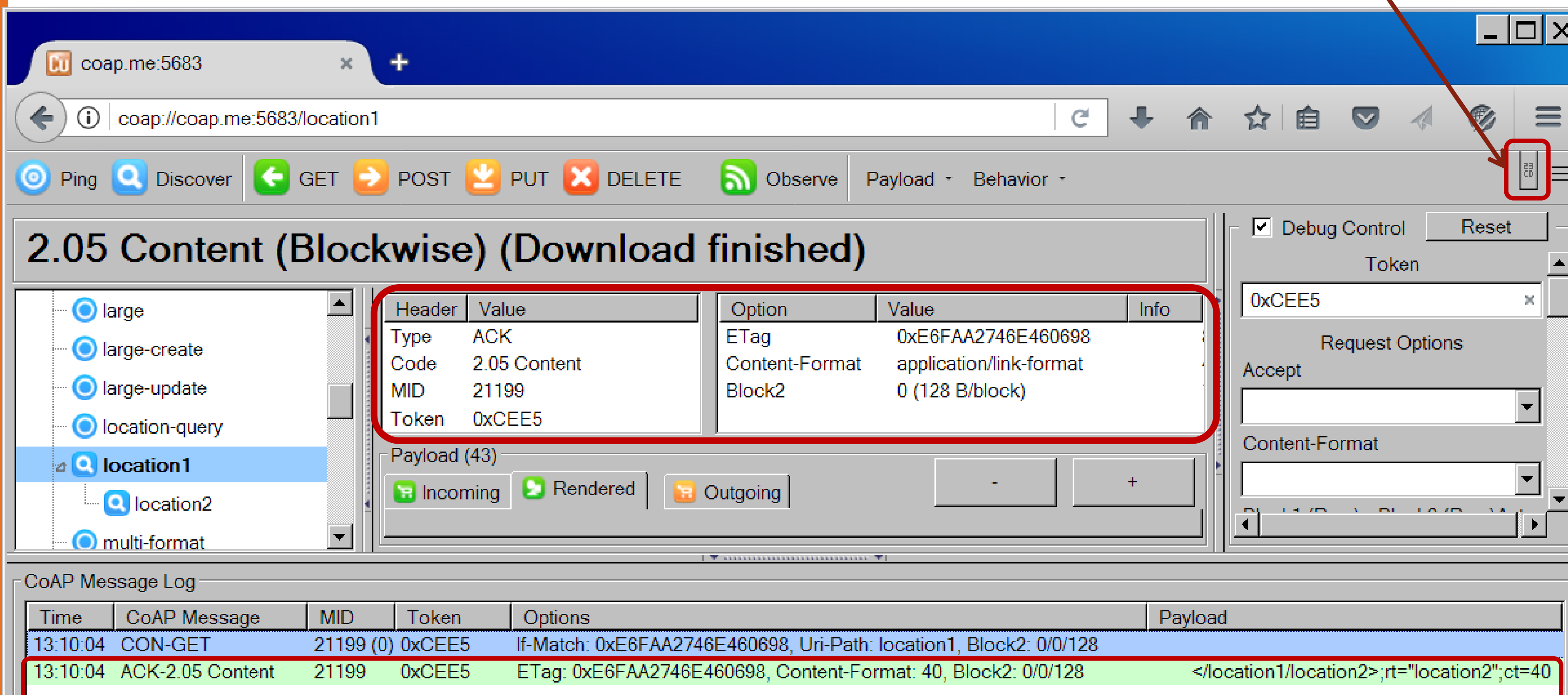
7.4. CoAP Content-Format
Media Type: application/cbor
Id: 60

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

PARSING EXAMPLE: MESSAGE

coap://coap.me:5683/location1

press here to see the log



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 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)
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

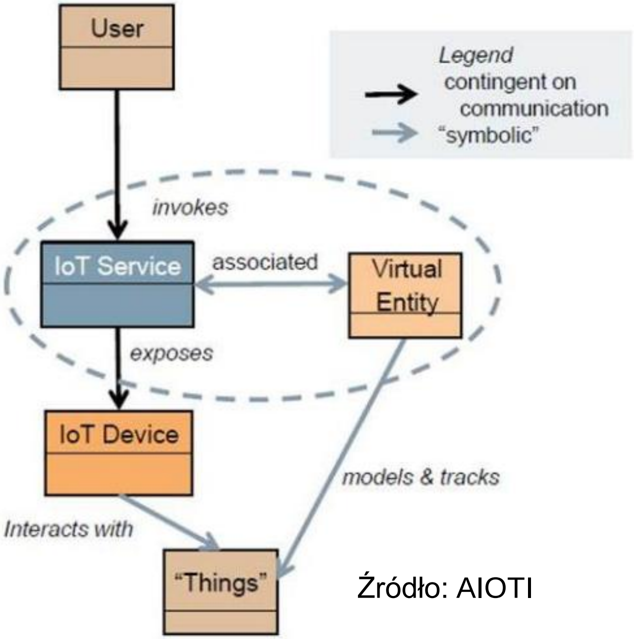
option delta
option no. 12+11=23 (Block2)
B1, option length
3, option value (1B), NUM/M/size= 0/0/128

FF payload marker – no more options

M=0
NUM=0 **0000 0 011** 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   |  11    |  11   |      "temperature" (11 B) ...
Uri-Path  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   Option Number  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    CON  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) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

      ACK  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          lost confirmable message
   |       |
   |       |
   +----X  |      Header: GET (T=CON, Code=0.01, MID=0x7d36)
   | GET   |       Token: 0x31
   |       |    Uri-Path: "temperature"
TIMEOUT    |                                                  same message ID
   |       |  request retransmitted
   +----->|       Header: GET (T=CON, Code=0.01, MID=0x7d36)
   | GET   |       Token: 0x31                 same token
   |       |    Uri-Path: "temperature"
   |       |
   |       |                                      Content
   |<-----+      Header: 2.05 Content (T=ACK, Code=2.05, MID=0x7d36)
   | 2.05  |       Token: 0x31
   |       |     Payload: "22.3 C"
   |       |
                                          piggybacked response
```

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]

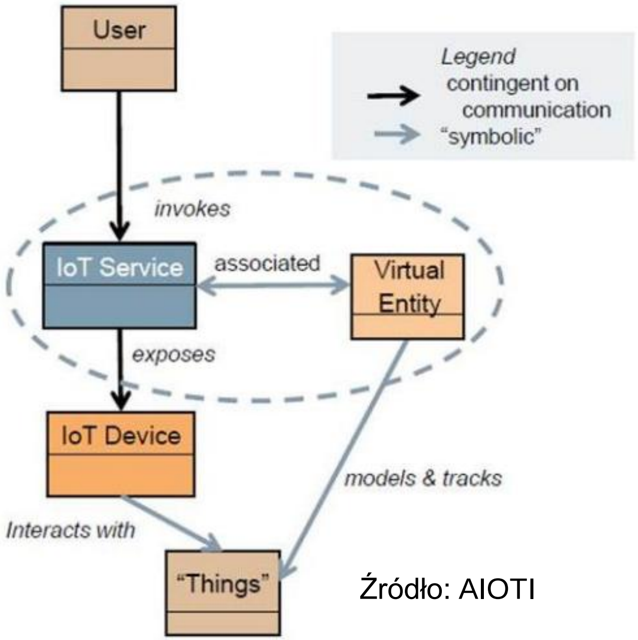
the response to CON request could also be NON

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      |
loss of state |       |      unexpected ACK silently ignored
              |<- - -+      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 clients 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 OF A STORED RESPONSE

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

CACHING: VALIDATING A STORED RESPONSE

- a client has some stored responses … but none of them is fresh
- maybe one of the stored responses is can be reused after all?
- need to validate stored responses
- to do so, a new GET is required, but it may turn out that transmitting a payload from the server is not needed
  - responses may be tagged by the server, with the **`ETag`** Option
  - the client can inquire if a stored response is valid by sending a **`GET`** with its **`Etag`**
  - multiple **`Etag’s`** may be included if the client has multiple stored responses
  - the server may respond with **`2.03 Valid`** and one of the **`Etag’s`** (without a payload, but possibly with a **`Max-Age`** option) …
  - … or the server may respond with **`2.05 Content`** and include a new payload

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

- proxies can cache responses

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: pretends it is an origin server

```
 __________                         ____________
|          |                       |            |
| 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: THE CLIENT KNOWS IT'S A PROXY

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

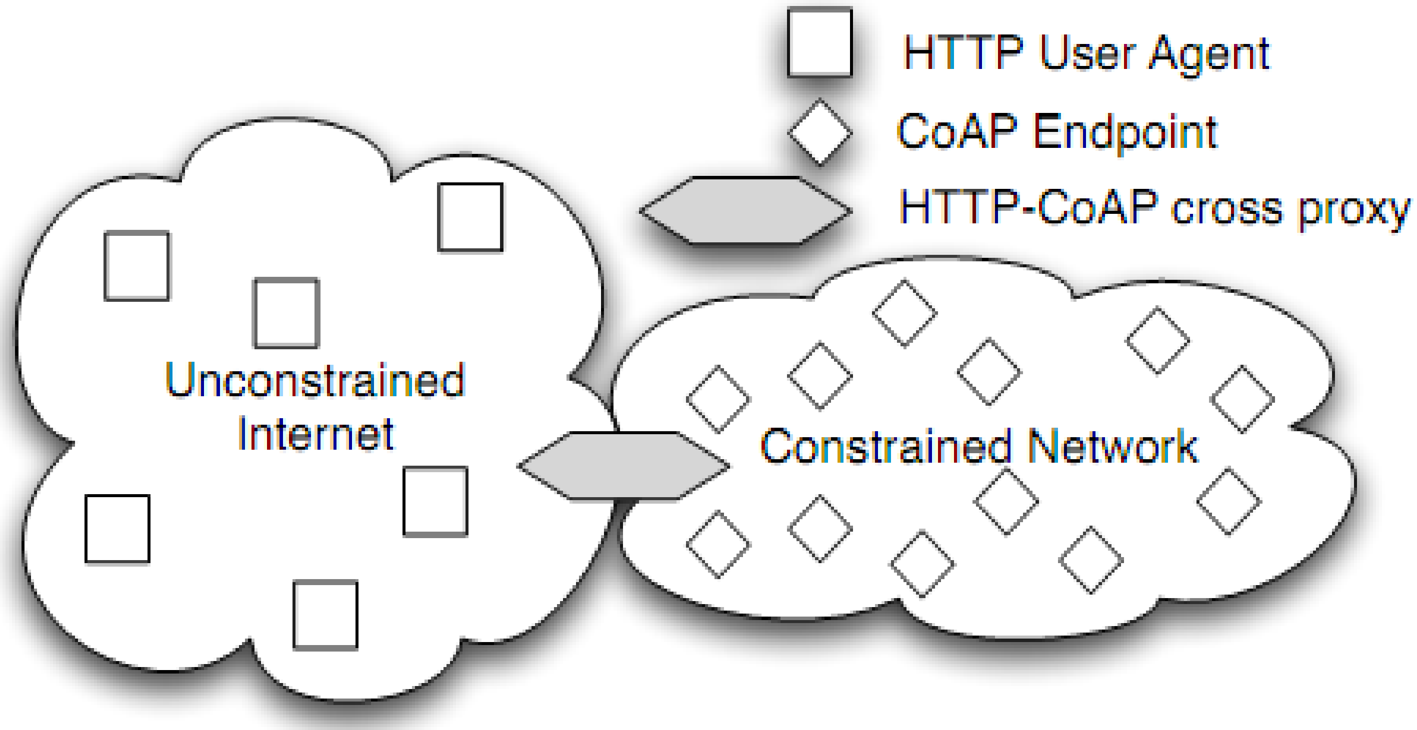
```
 _________                    _________
|         |                  |         |
| User   (C)-----------------(S) Origin |
| Agent   |                  |  Server |
|_________|                  |_________|
(Browser)                    (Web Server)
   Figure 1: Client-Server Communication
```

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

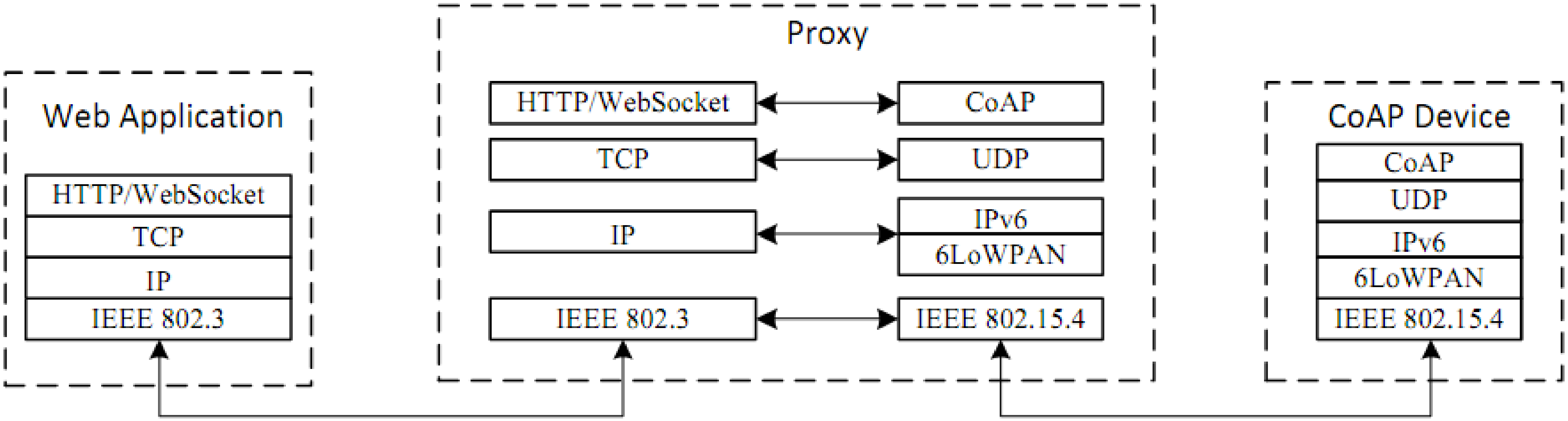
```
 _________       __________                    _________
|         |     |          |                  |         |
| User   (C)---(S) Inter- (C)-----------------(S) Origin |
| Agent   |     |  mediary |                  |  Server |
|_________|     |__________|                  |_________|
(Browser)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



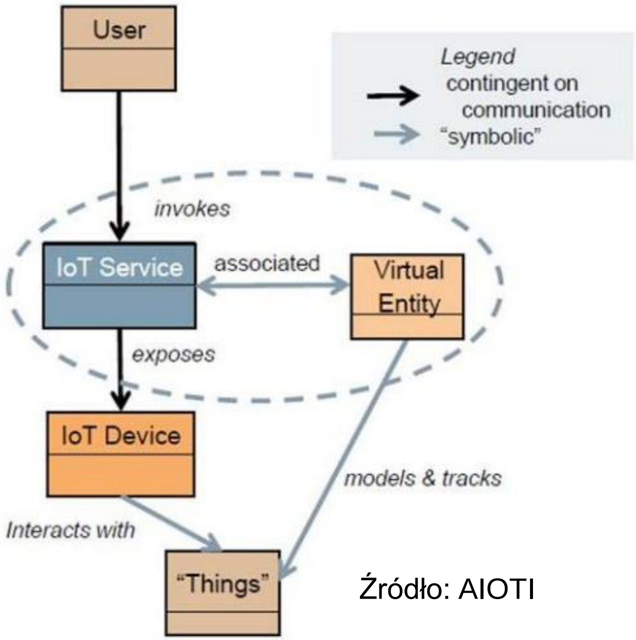
Source:
A Proxy Design to Leverage the Interconnection of CoAP Wireless Sensor Networks with Web Applications
A. Ludovici, A. Calveras
Sensors, 2015

- 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 an up-to-date representation of a resource over a period of time.
- HTTP: polling, long polling
- what's bad: timers, traffic, delays …
- polling vs. interrupts, pull vs. push

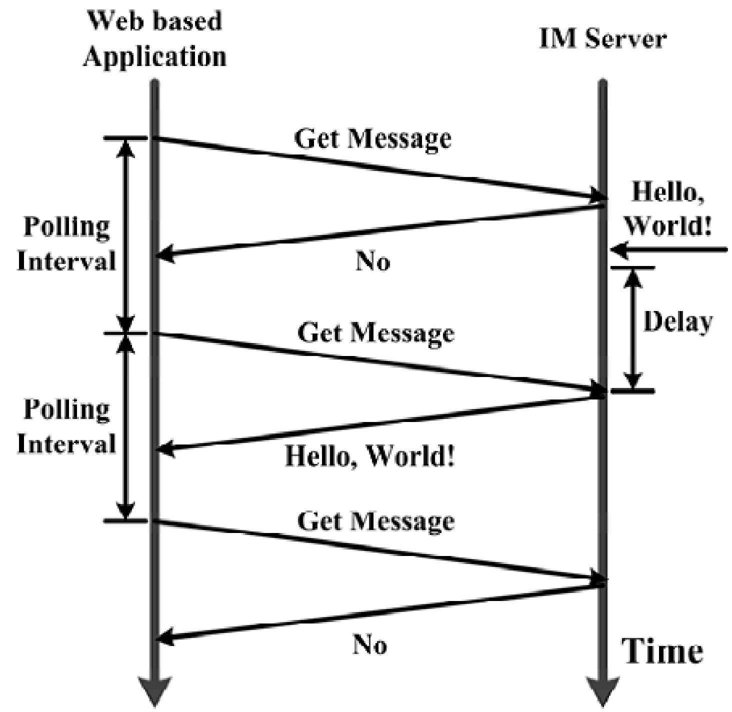


Figure 1. Workflow of HTTP Polling

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      |
   +--------------------->|   one registration
   |                      |
   |    Notification      |
   |<---------------------+   multiple notifications
   |                      |
   |    Notification      |
   |<---------------------+
   |                      |
   |    Notification      |
   |<---------------------+
   |                      |
```

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 of the option 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 …
  - 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

lista obserwatorów

- 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

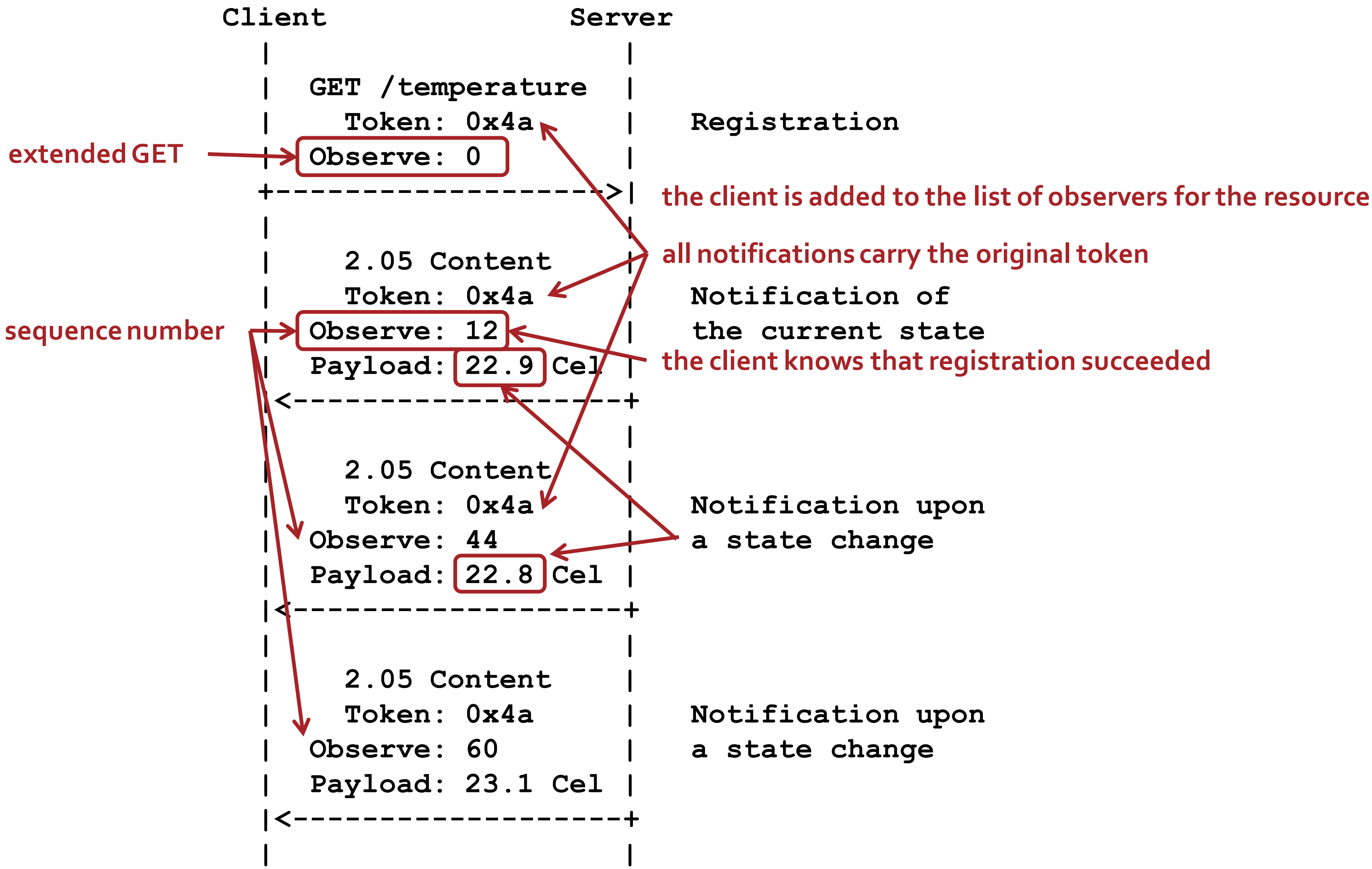


Figure 2: Observing a resource in CoAP [RFC7641]

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 <u>server</u> 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"          sequence
14                    |      |                                                numbers
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                    |      |
```

aside: decode 0x61451633

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       a new state
24                 |       |   ____________
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                   |       |    re-registration   Observe: 0 (register)
 71                   |       |    is response with     ETag: 0x78797a7a79
 72                   |       |    this Etag valid?
 73                   |       |                              Content
 74    ____________   |<-----+                        Header: 2.05 0x61451635
 75                   | 2.05 |                         Token: 0xf9
 76     20.0 Cel      |       |                      Observe: 74
 77                   |       |    no, need to send  Max-Age: 15
 78                   |       |    notification      Payload: "20.0 Cel"
 79                   |       |    with payload
 80                   |       |                              Valid
 81    ____________   |<-----+    ____________       Header: 2.03 0x5143aa0c
 82                   | 2.03 |    19.7 Cel             Token: 0xf9
 83     19.7 Cel      |       |    now it's valid    Observe: 81
 84                   |       |    again, can refer     ETag: 0x78797a7a79
 85                   |       |    to it with ETag   Max-Age: 15
 86   stored response |       |                              no payload!
```

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

ETag

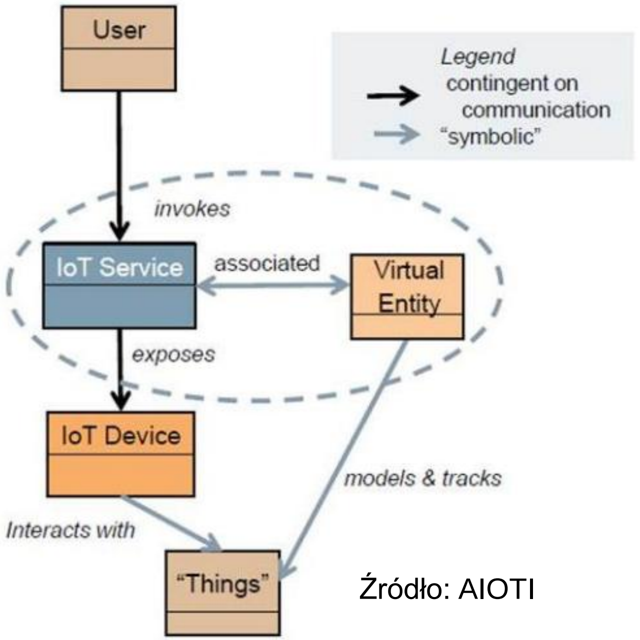
```
        Observed   CLIENT  SERVER     Actual
    t   State         |       |         State
        ____________  |       |  ____________
   87                 |       |
   88    19.7 Cel     |       |    19.7 Cel
   89                 |       |
   90                 |       |  ____________    regular 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
  - **`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",...
```

typed link

target URI

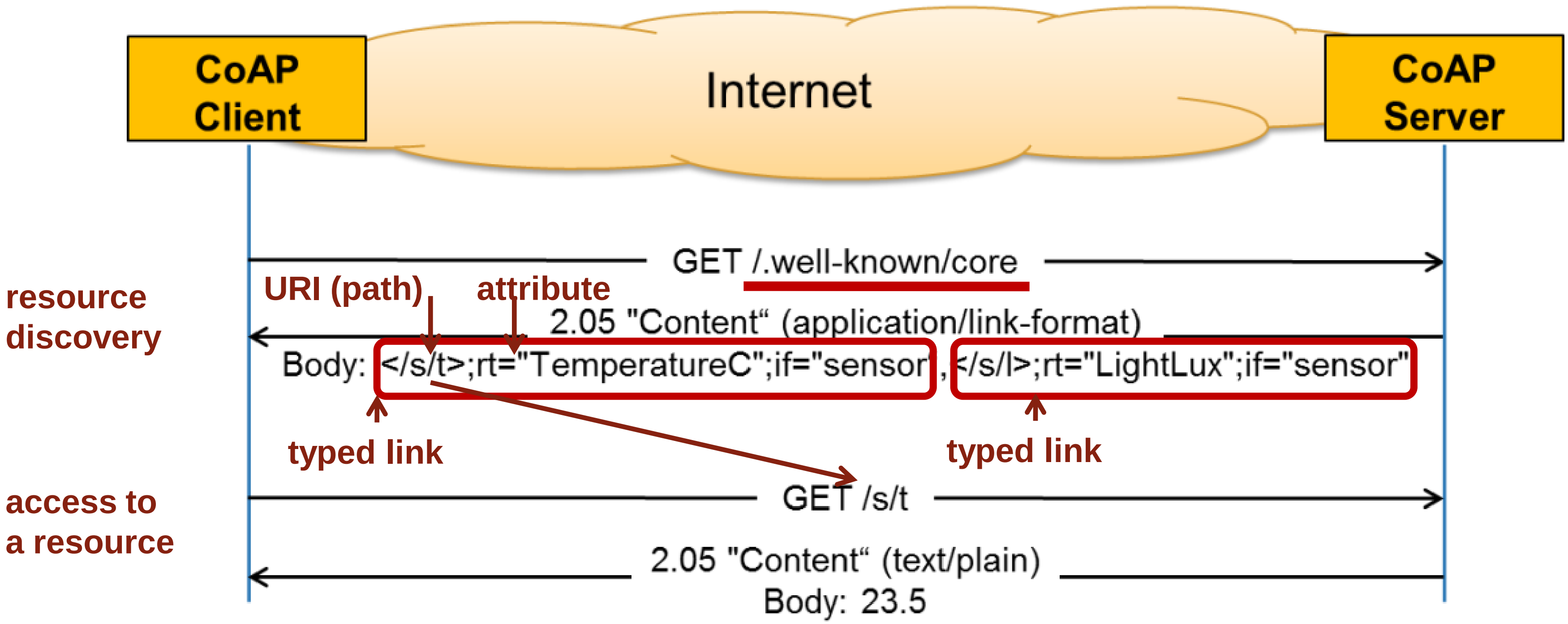
target attributes

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)

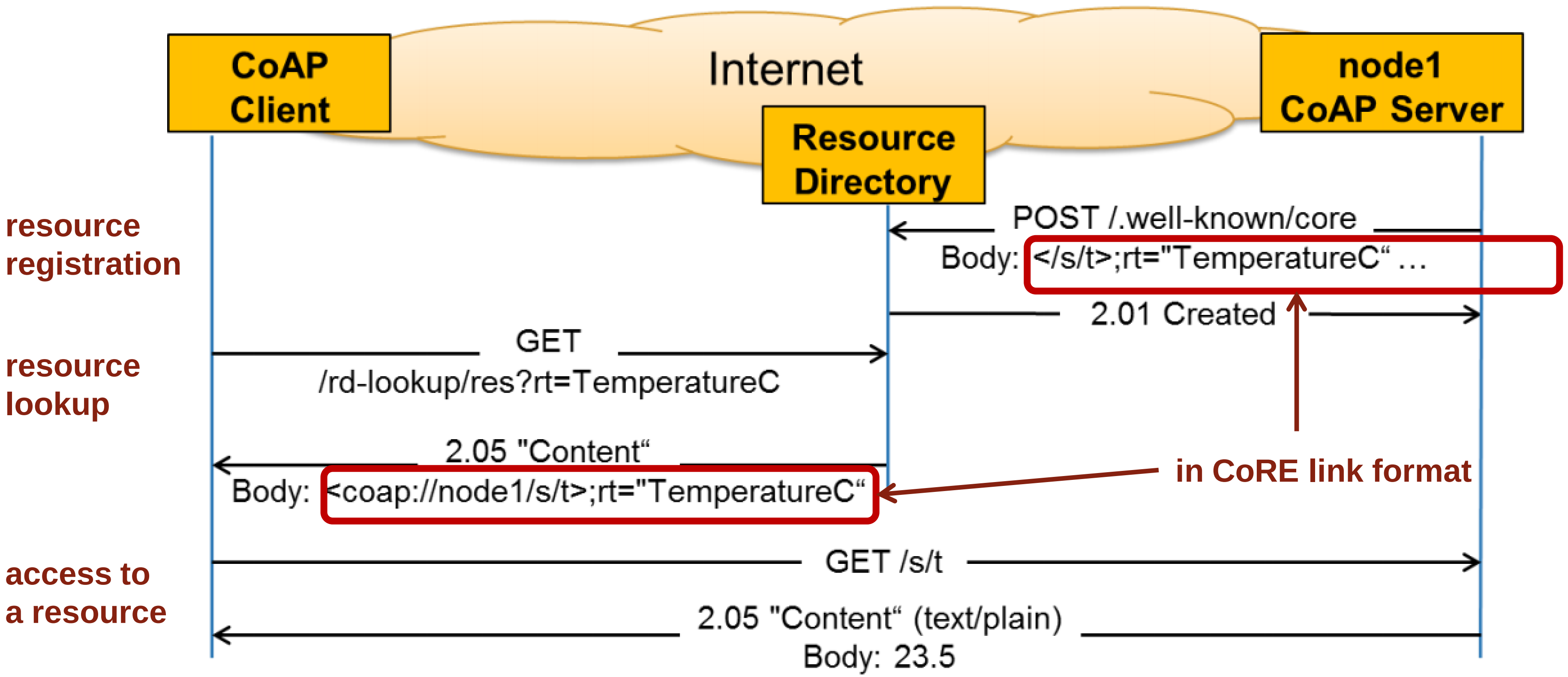
CoRE Resource Discovery: przykład



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

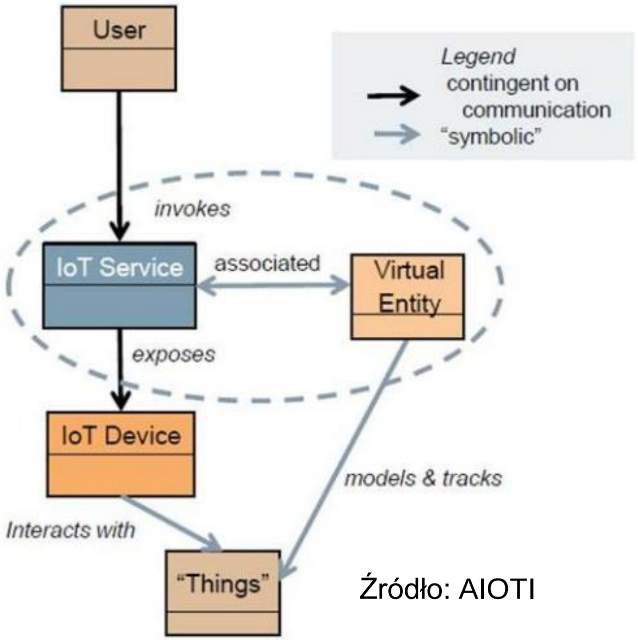
Kilka pytań o odkrywaniu zasobów

- 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 <u>may not be enough</u>
- answer: **semantics**

Block-Wise Transfers in CoAP

how to transfer big resource representations

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

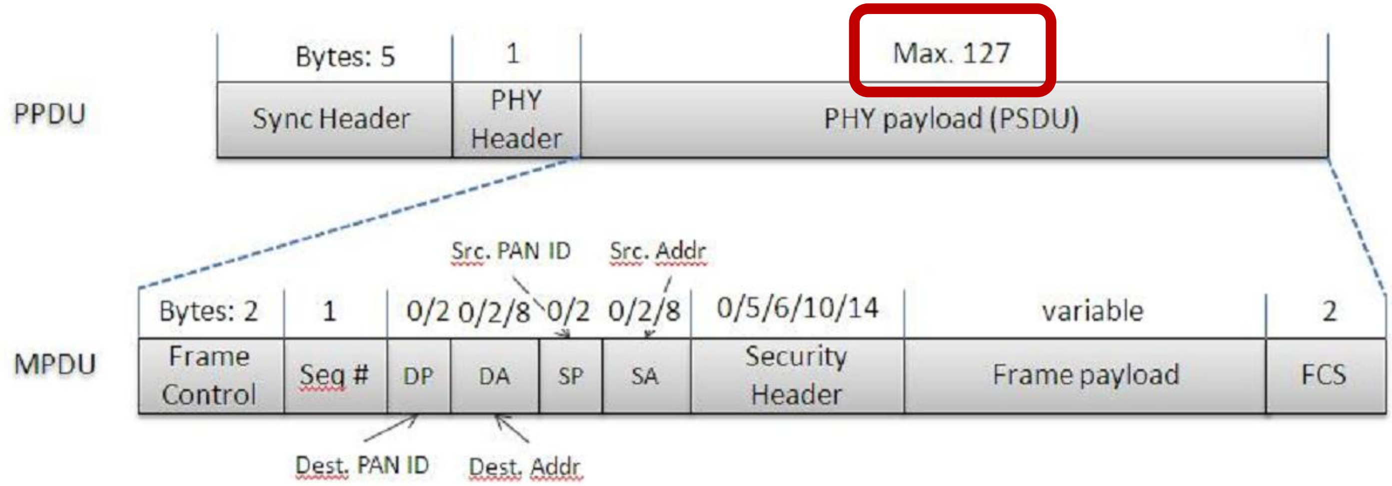


Źródło: AIOTI

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.	Name	Format	Length	Default
GET (payload in response)	23	Block2	uint	0-3	(none)
POST, PUT (payload in request)	27	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 = 0, 1, …, 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                 regular GET                                     SERVER
  |                                      block size proposed by server |
  | CON [MID=1234], GET, /status                               ------> |
  |                               there are more blocks                |
  | <------   ACK [MID=1234], 2.05 Content, 2:0/1/128                  | more blocks, payload 128 B
  |                                 option Block2                      |
  | each block needs to be explicitly requested                        |
  | CON [MID=1235], GET, /status, 2:1/0/128   client agrees to block size
  |                                                            ------> |
  |                                                                    |
  | <------   ACK [MID=1235], 2.05 Content, 2:1/1/128                  | more blocks, payload 128 B
  |                                                                    |
  |                                                                    |
  | CON [MID=1236], GET, /status, 2:2/0/128                    ------> |
  |                                   no more blocks                   |
  | <------   ACK [MID=1236], 2.05 Content, 2:2/0/128                  | 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                                                     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]

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

payload 64 B

payload 64 B

payload 64 B

0 < payload <= 64 B

no more blocks

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                       ------> |
  |                              piggybacked response        |
  | <------   ACK [MID=1234], 2.05 Content, 2:0/1/128         |   late negotiation
  |                                                           |
  | CON [MID=1235], GE///////////////////////////             |
  |                                                           |
  | (timeout)                                 lost CON        |
  |                     retransmission                        |
  | 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)      retransmission      lost ACK              |
  |                                                          |
  | 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 [RFC7959]

OPCJA BLOCK1: PRZYKŁAD

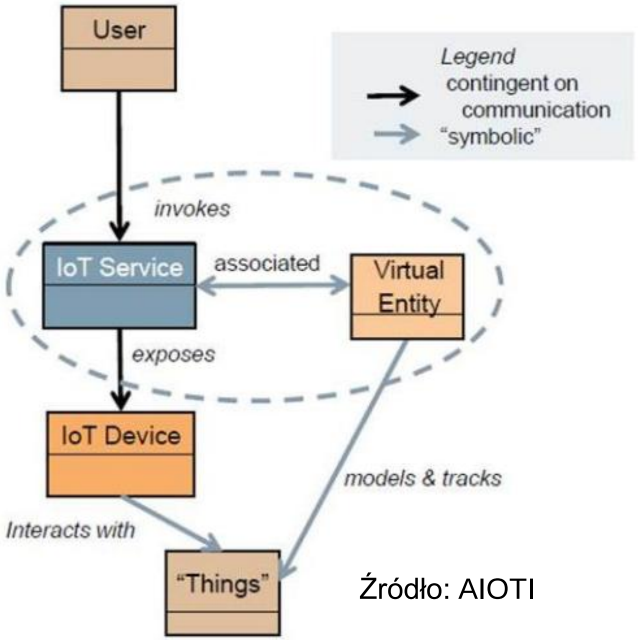
```
                                   option Block1
CLIENT                                                      SERVER
 |                                      |                      |
 | CON [MID=1234], PUT, /options, 1:0/1/128    ------>         | payload 128 B
 |          new response code                                  |
 | <------   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