Constrained Application Protocol

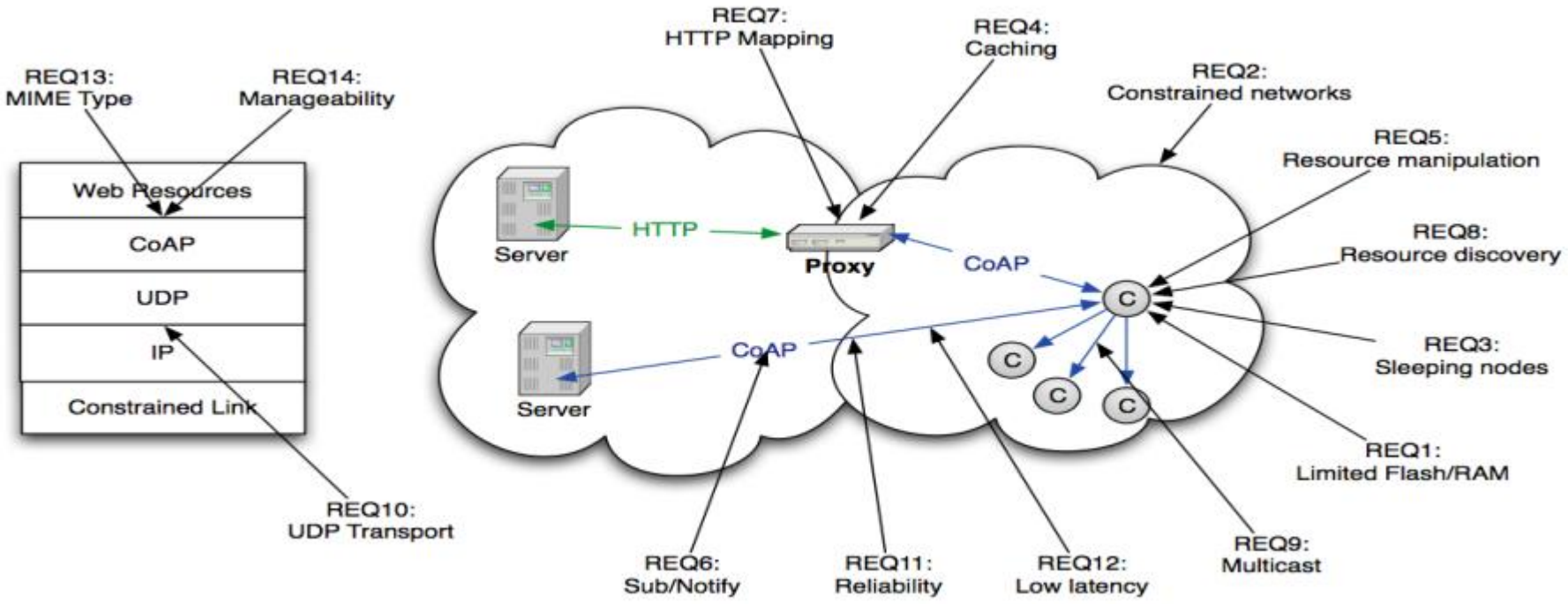
Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InES Institute of Embedded Systems

CoAP: Framework (1)

CoAP Requirements

Session 1: CoAP Basics

- **Framework**
- Messaging Fundamentals
- Lifetimes and Timeouts
- Discovery (RFC 6690)
- The Header
- Group Communication
- Observer
- Block Communication



http://community.arm.com/servlet/JiveServlet/previewBody/8633-102-2-15471/ARM%20CoAP%20Tutorial%20April%2030%202014.pdf

CoAP: Framework (2)

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InES Institute of Embedded Systems

- **Defined in RFC 7252**

Session 1: CoAP Basics

- **Framework**
- Messaging Fundamentals
- Lifetimes and Timeouts
- Discovery (RFC 6690)
- The Header
- Group Communication
- Observer
- Block Communication

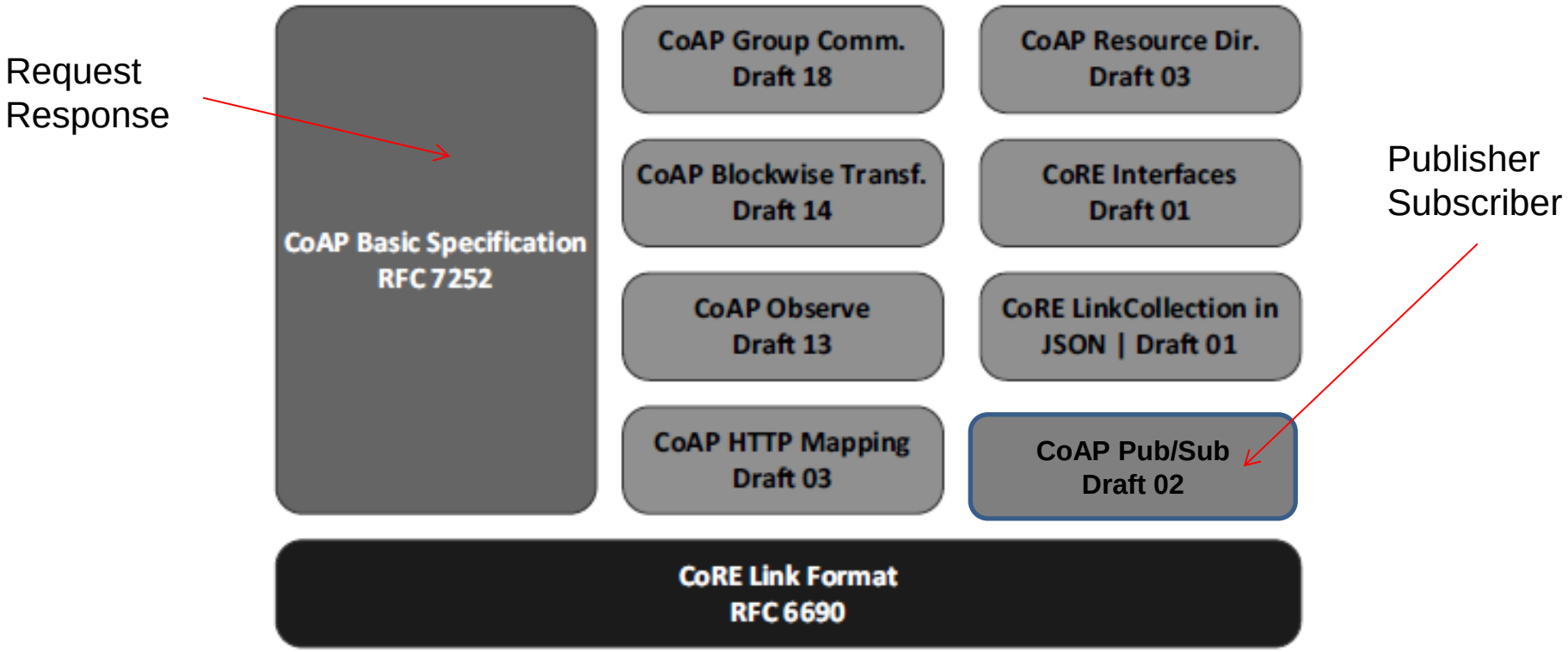
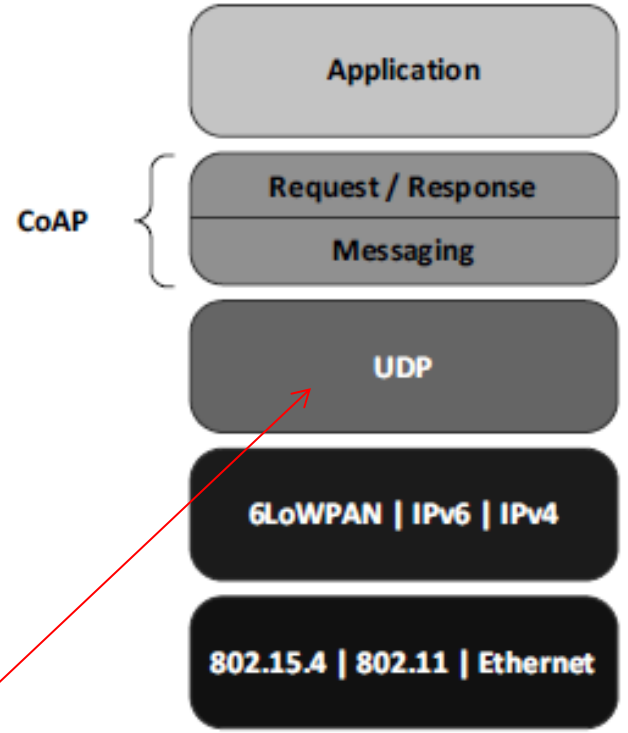


Fig. 1. Development status of CoAP

Real-Time Communication for the Internet of Things using jCoAP, Björn Konieczek, Michael Rethfeldt, Frank Golatowski and Dirk Timmermann

CoAP: Framework (3)

- Constraints:
- UDP offers lower overhead for cable based network
- IEEE 802.15.4 – maximum payload 127 bytes !!
- Simple CoAP server + TinyOS + 6LoWPAN stack -> 32 kBytes ROM, 5.5 kBytes RAM



Default UDP Port 5683

Fig. 2. Basic CoAP stack

Real-Time Communication for the Internet of Things using jCoAP, Björn Konieczek, Michael Rethfeldt, Frank Golatowski and Dirk Timmermann
https://www.ietf.org/mail-archive/web/core/current/msg01231.html

CoAP: Framework (4)

- Small (ish) Message Header
- 4 bytes + tokens and options

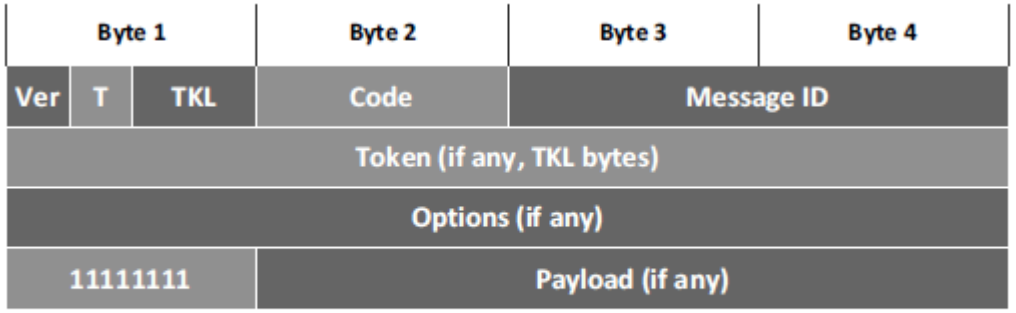


Fig. 3. Structure of the CoAP header

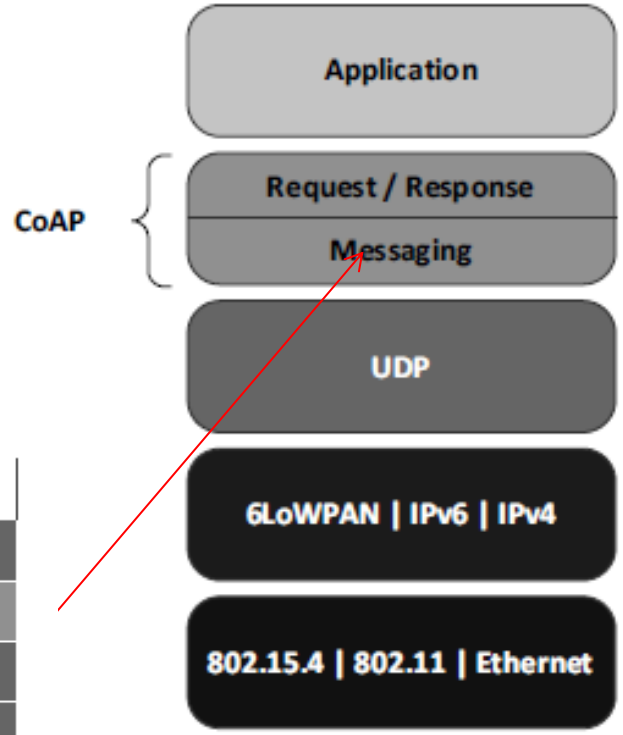


Fig. 2. Basic CoAP stack

Real-Time Communication for the Internet of Things using jCoAP, Björn Konieczek, Michael Rethfeldt, Frank Golatowski and Dirk Timmermann
https://www.ietf.org/mail-archive/web/core/current/msg01231.html

CoAP: Framework (5)

- Embedded web transfer protocol (coap://)
- Asynchronous transaction model
- UDP binding with reliability and multicast support
- GET, POST, PUT, DELETE methods
- URI support
- Small, simple 4 byte header
- DTLS based PSK, RPK and Certificate security
  - Datagram Transport Layer Security
- Subset of MIME types and HTTP response codes
- Built-in discovery
- Optional observation and block transfer
- Publisher subscriber model under definition

CoAP: Framework (5)

- Embedded web transfer protocol (coap://)
- Asynchronous transaction model
- UDP binding with reliability and multicast support
- GET, POST, PUT, DELETE methods
- URI support
- Small, simple 4 byte header
- DTLS based PSK, RPK and Certificate security
  - Datagram Transport Layer Security
- Subset of MIME types and HTTP response codes
- Built-in discovery
- Optional observation and block transfer
- Publisher subscriber model under definition

CoAP Header (1)

Version Number: currently (RFC 7252) == 0x01

Message Type: [0x00 … 0x03]

Confirmable Message =	0x00
Non-Confirmable Message =	0x01
Acknowledgement =	0x02
Reset =	0x03

Token Length

Token

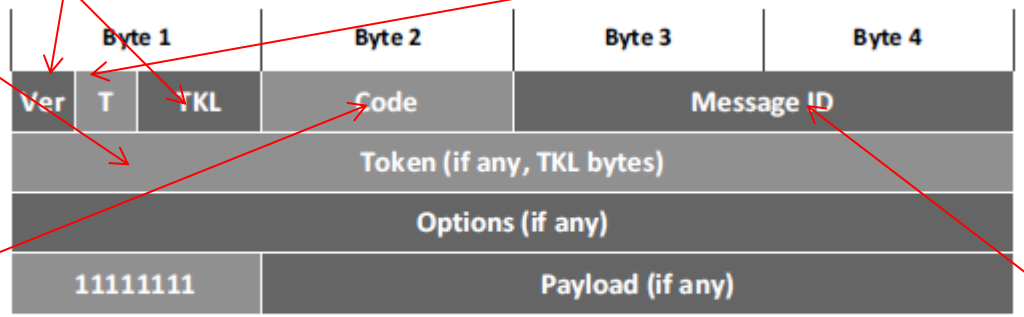


Fig. 3. Structure of the CoAP header

Code:

[0.00]	Empty
[0.01 … 0.31]	Request
[2.00 … 5.31]	Response
[1.00 … 1.31, 6.00 … 7.31]	Reserved

Message ID: uint_16

CoAP Header: Code

- Code is an 8 bit value split into a 3-bit class and 5- bit detail

- Request Codes
  - GET -> 0.01 -> 0x01
  - POST -> 0.02 -> 0x02
  - PUT ->0.03 -> 0x03
  - DELETE -> 0.04 -> 0x04
- Example Response Code
  - «Not Found» -> 4.04 -> 100'00100b -> 0x84

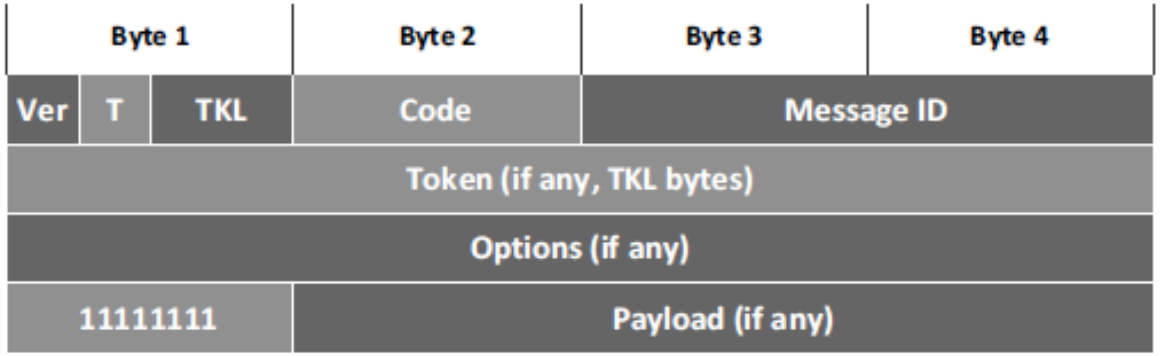


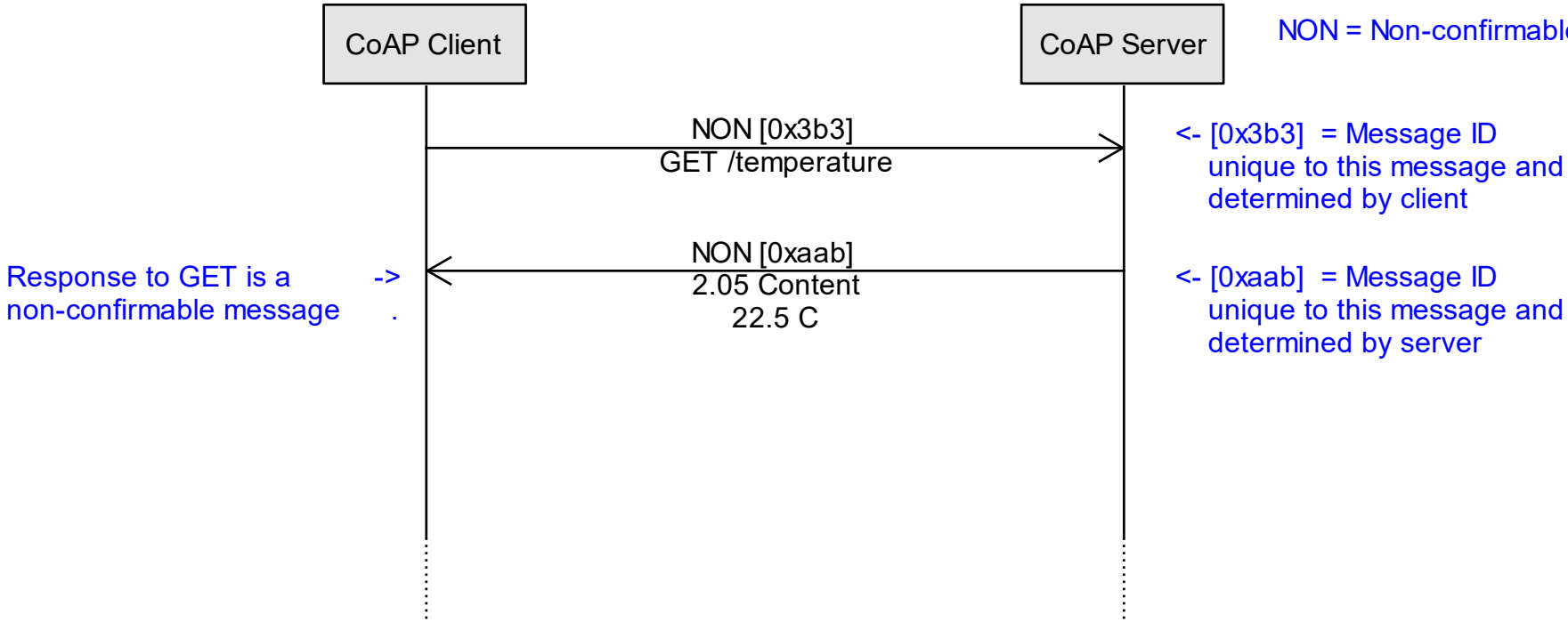
Fig. 3. Structure of the CoAP header

CoAP Fundamental Messaging (1)

- Simplest NON Message

Session 1: CoAP Basics

- Framework
- **Messaging Fundamentals**
- Lifetimes and Timeouts
- Discovery (RFC 6690)
- The Header
- Group Communication
- Observer
- Block Communication



source: CoAP_01_Fundamental_Messaging_1.svg

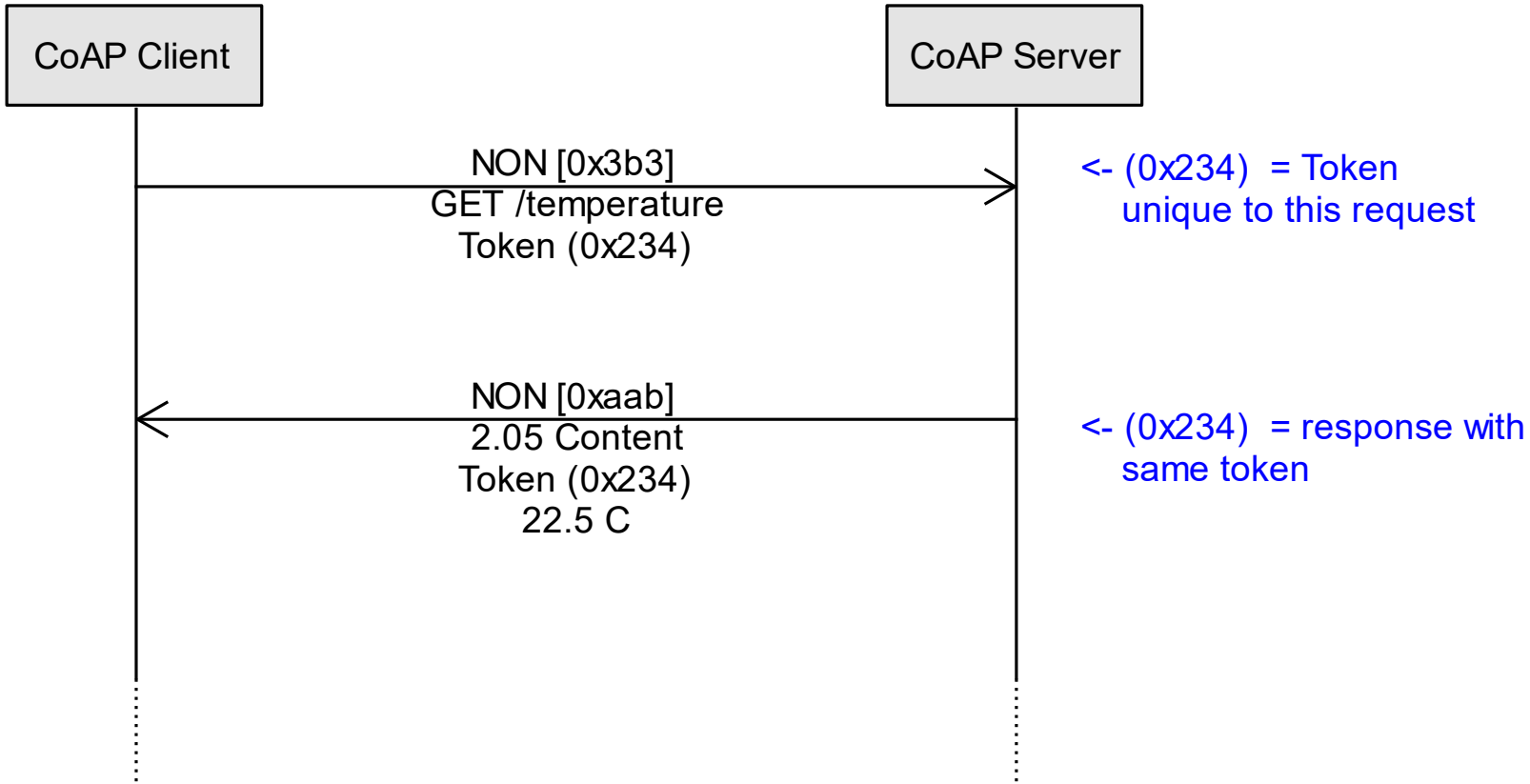
CoAP Fundamental Messaging (2)

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InES Institute of Embedded Systems

- Tokenised NON Message

Session 1: CoAP Basics

- Framework
- **Messaging Fundamentals**
- Lifetimes and Timeouts
- Discovery (RFC 6690)
- The Header
- Group Communication
- Observer
- Block Communication



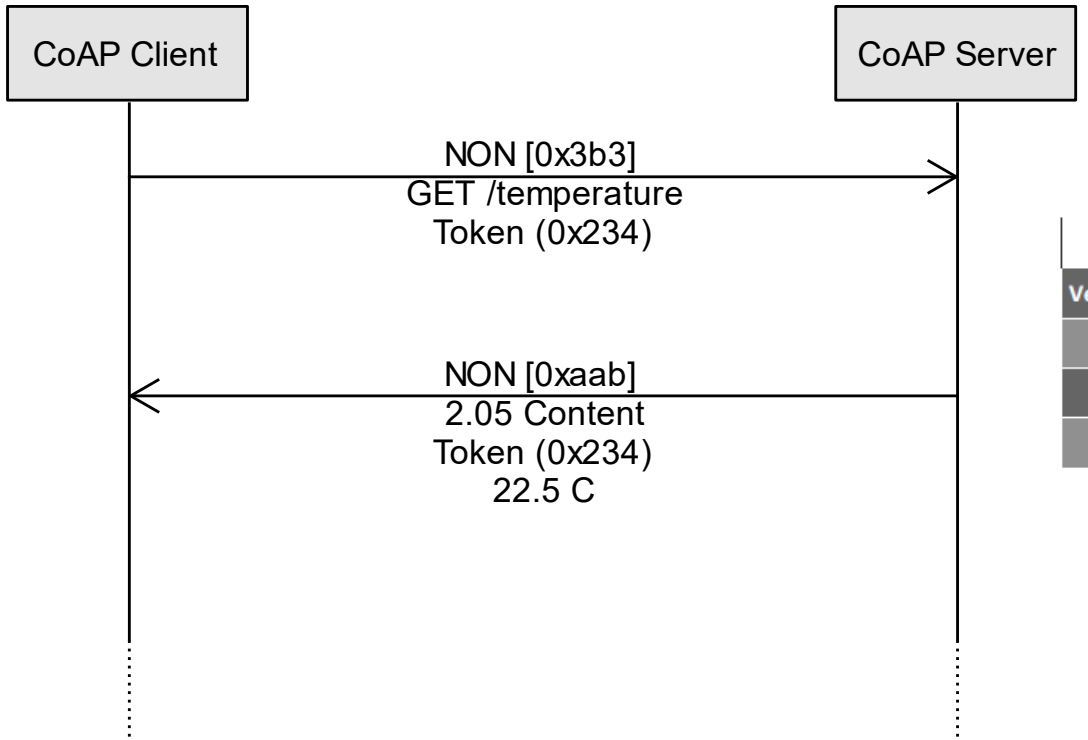
source: CoAP_02_Fundamental_Messaging_2.svg

CoAP Exercise (1)

- Fill out the first 6 fields of the CoAP header for the response message

Session 1: CoAP Basics

- Framework
- **Messaging Fundamentals**
- Lifetimes and Timeouts
- Discovery (RFC 6690)
- The Header
- Group Communication
- Observer
- Block Communication



source: CoAP_03_Fundamental_Messaging_Exercise.svg

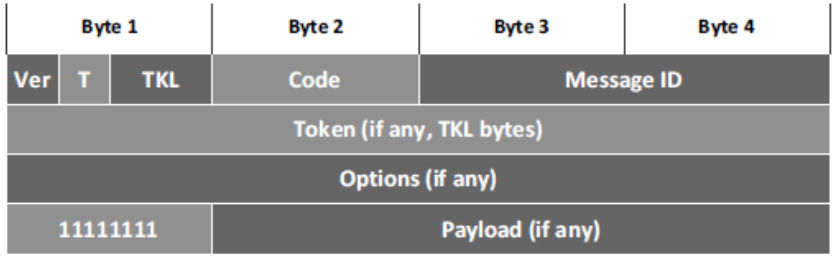
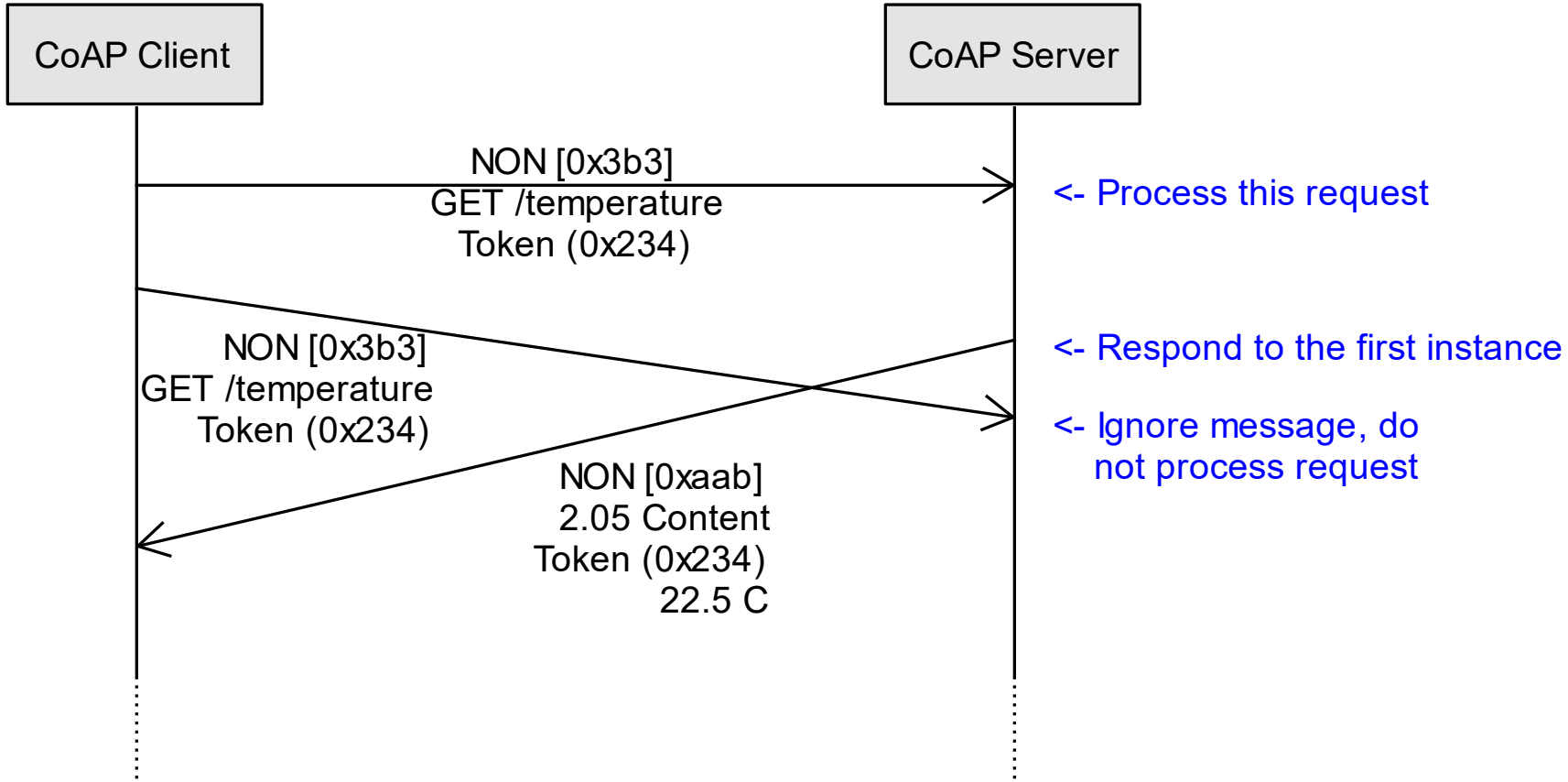


Fig. 3. Structure of the CoAP header

CoAP Fundamental Messaging (3)

- Duplication Handling



source: CoAP_04_Fundamental_Messaging_3.svg

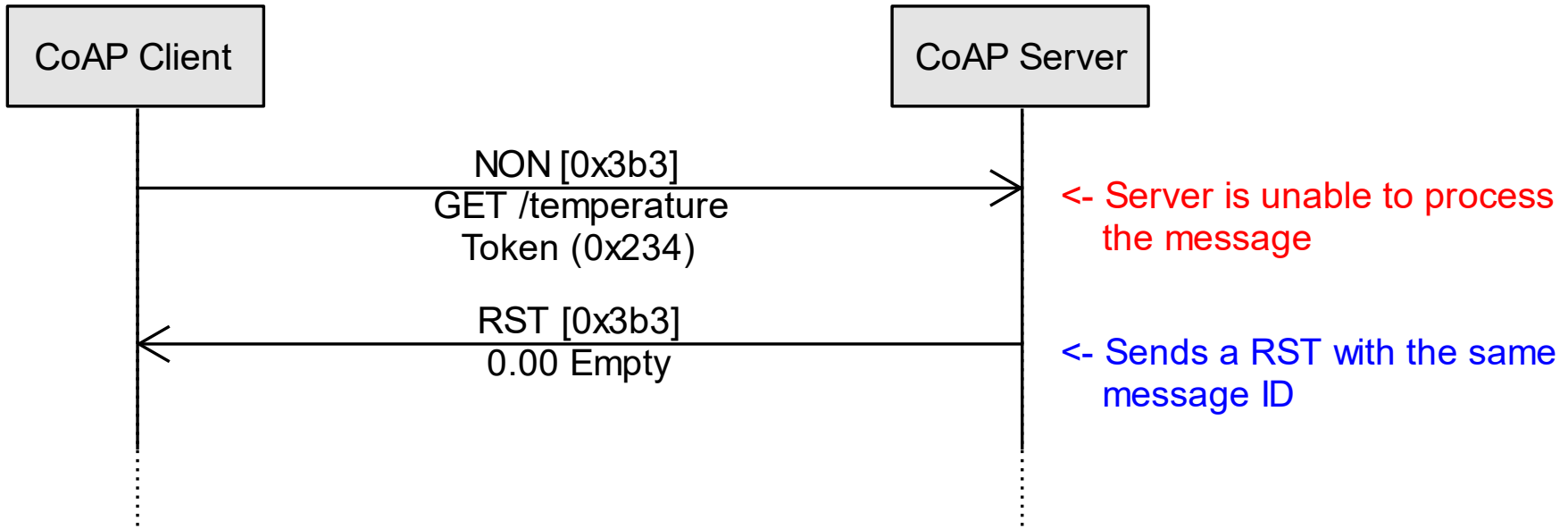
CoAP Fundamental Messaging (4)

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InES Institute of Embedded Systems

- Server Unable to Process Messages

Session 1: CoAP Basics

- Framework
- **Messaging Fundamentals**
- Lifetimes and Timeouts
- Discovery (RFC 6690)
- The Header
- Group Communication
- Observer
- Block Communication



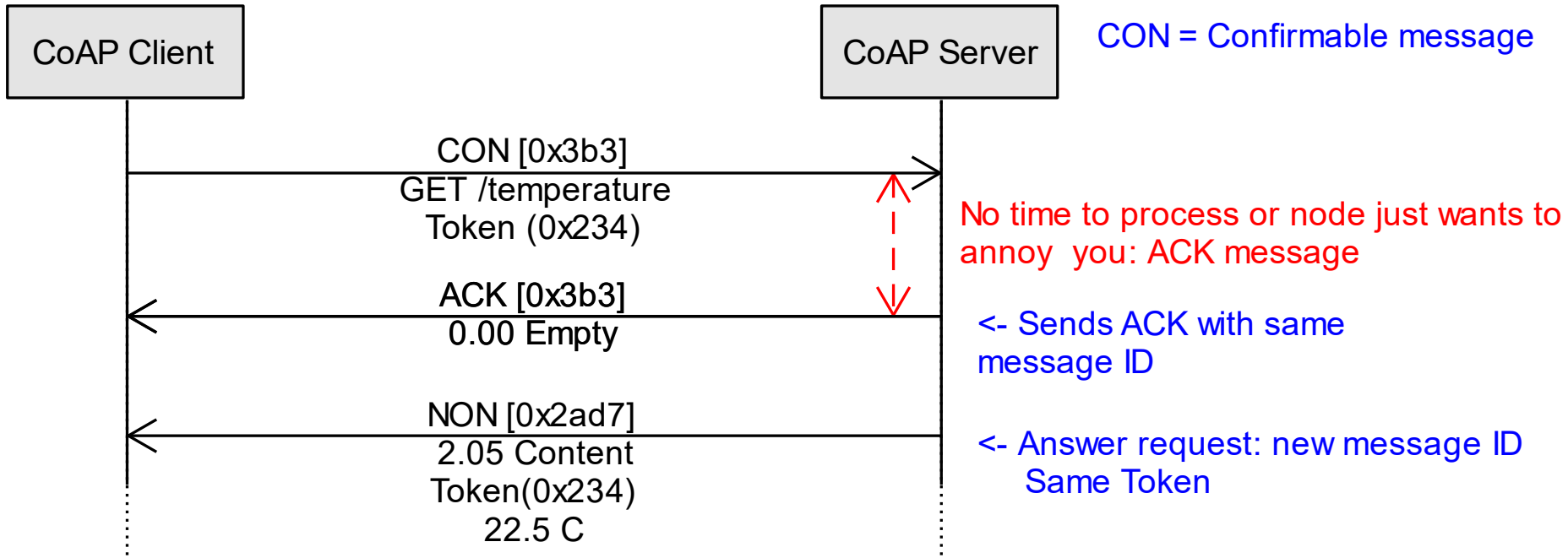
source: CoAP_05_Fundamental_Messaging_4.svg

CoAP Fundamental Messaging (5)

- Confirmable Messages – Separate Response

Session 1: CoAP Basics

- Framework
- **Messaging Fundamentals**
- Lifetimes and Timeouts
- Discovery (RFC 6690)
- The Header
- Group Communication
- Observer
- Block Communication



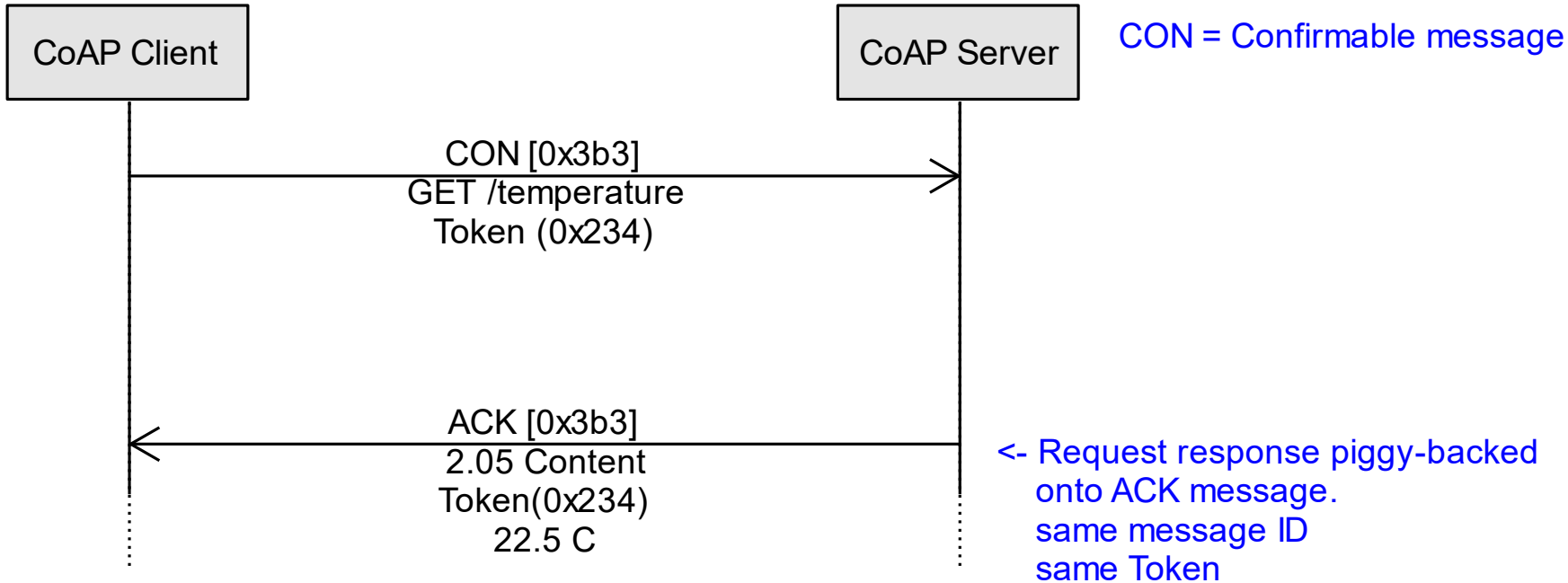
source: CoAP_06_Fundamental_Messaging_5.svg

CoAP Fundamental Messaging (6)

- Confirmable Messages – Piggy-Back Response

Session 1: CoAP Basics

- Framework
- **Messaging Fundamentals**
- Lifetimes and Timeouts
- Discovery (RFC 6690)
- The Header
- Group Communication
- Observer
- Block Communication



source: CoAP_07_Fundamental_Messaging_6.svg

CoAP Exercise (2)

- Exercise

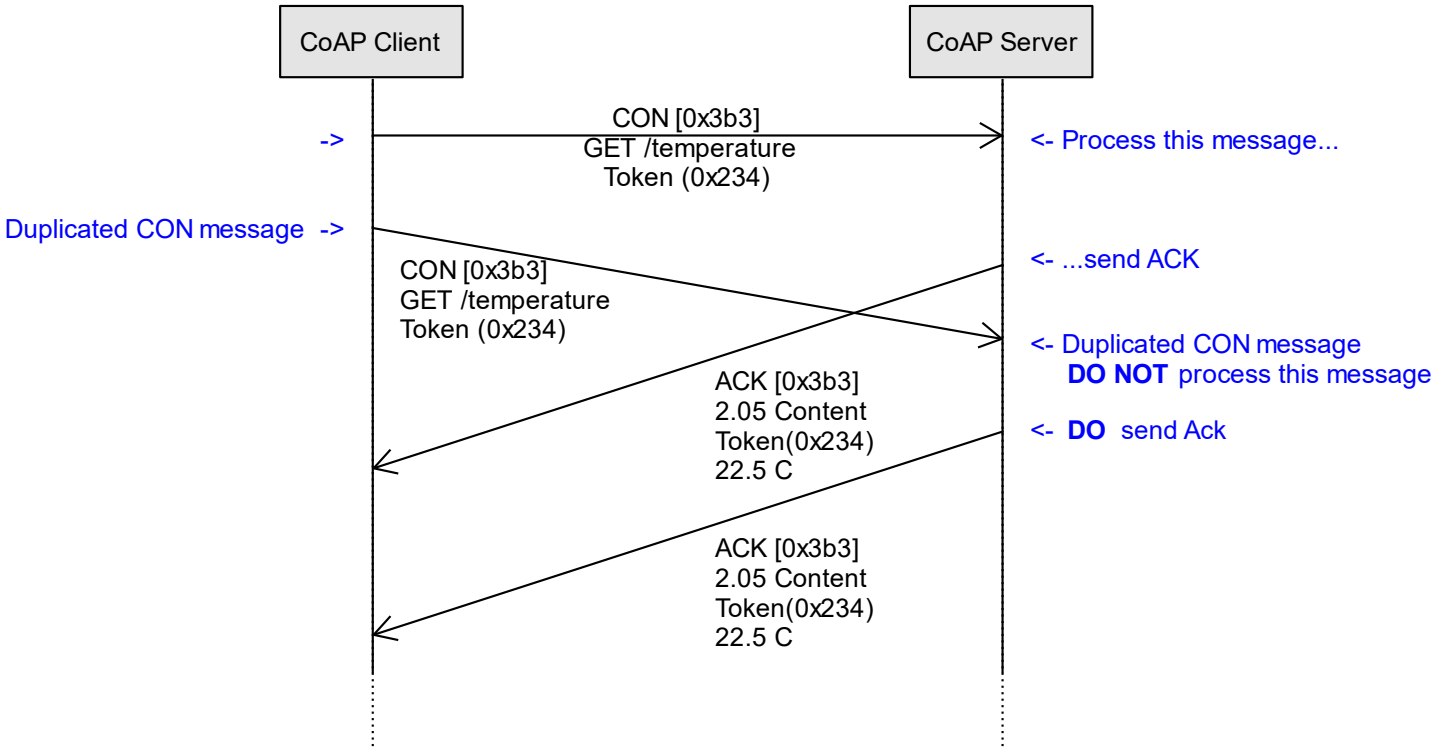
- A server returns an ACK to a CON. It then sends a CON message with the response.
  - Is this possible?
  - What do you expect the client to return?

CoAP Fundamental Messaging (7)

Confirmable Messages Duplication Handling

Session 1: CoAP Basics

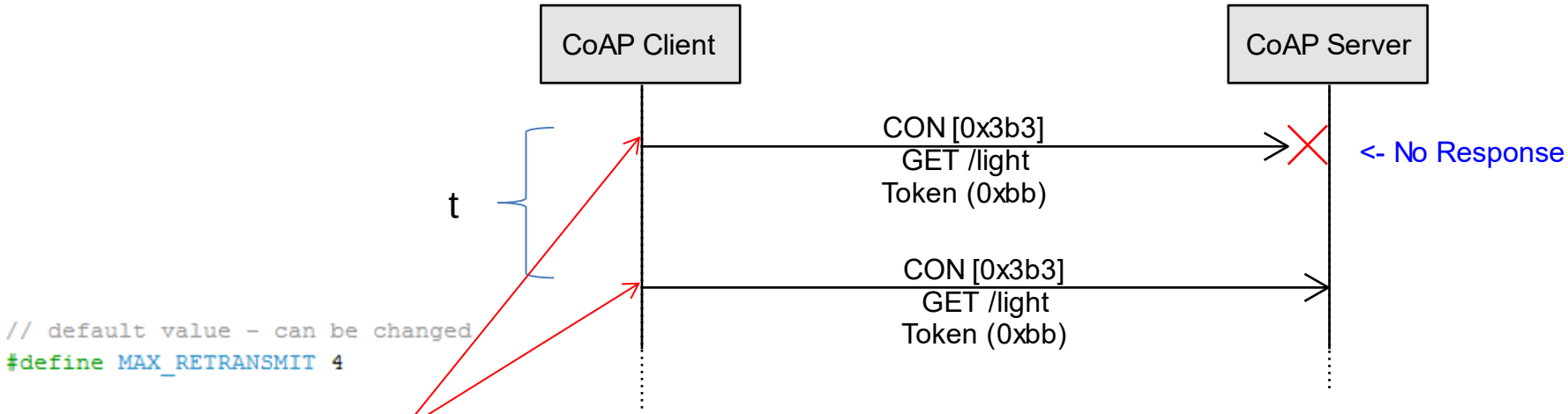
- Framework
- **Messaging Fundamentals**
- Lifetimes and Timeouts
- Discovery (RFC 6690)
- The Header
- Group Communication
- Observer
- Block Communication



source: CoAP_08_Fundamental_Messaging_7.svg

Message Lifetimes and Timeouts (1)

- Timeout




```
// default value - can be changed
#define MAX_RETRANSMIT 4

uint_8 retransmit_number = 0;

retransmit_number ++;
if (retransmit_number <= MAX_RETRANSMIT)
    tranmsit(message_ID)
```

source: CoAP_09_Messaging_1.svg

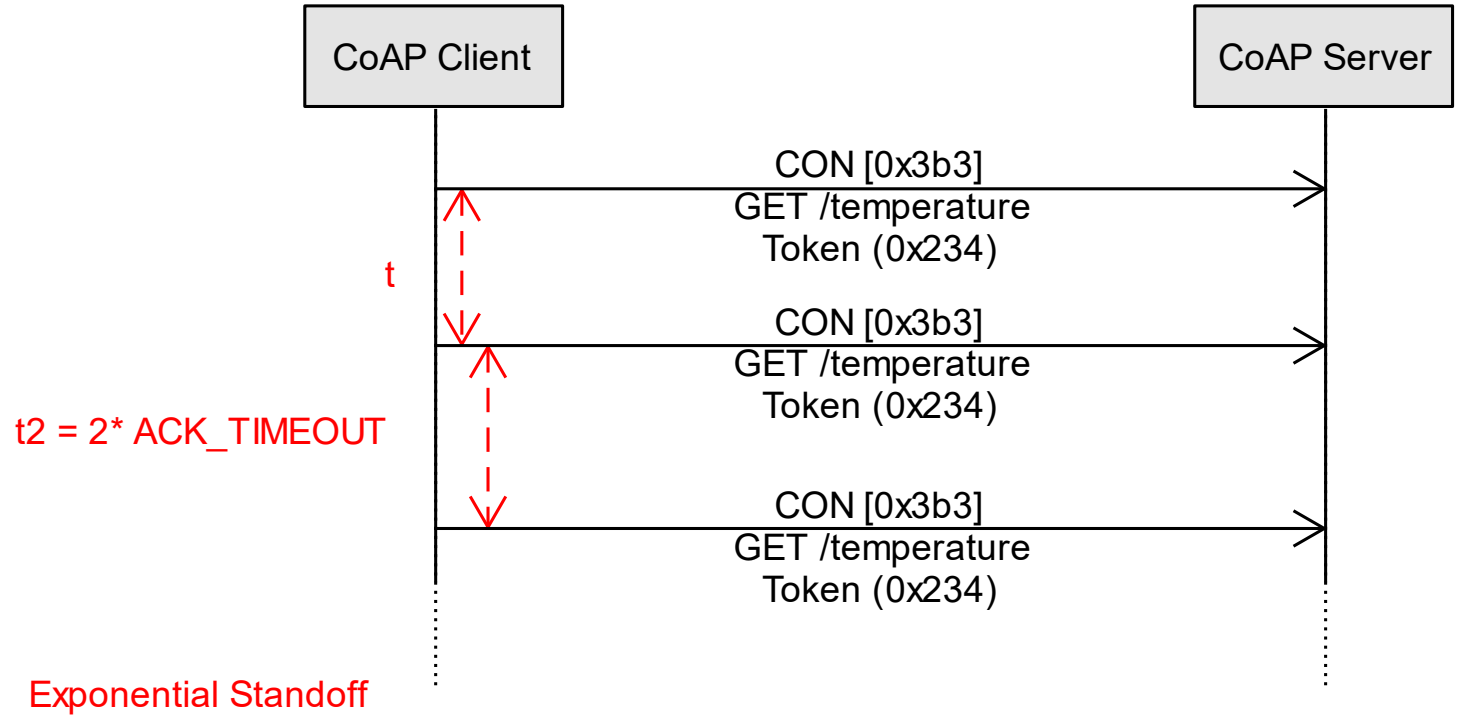
ACK_TIMEOUT <= t >= (ACK_TIMEOUT * ACK_RANDOM_FACTOR)

ACK_TIMEOUT = 2 seconds
ACK_RANDOM_FACTOR = 1.5 seconds

Default Values

Message Lifetimes and Timeouts (2)

- Congestion Control



source: CoAP_10_Messaging_2.svg

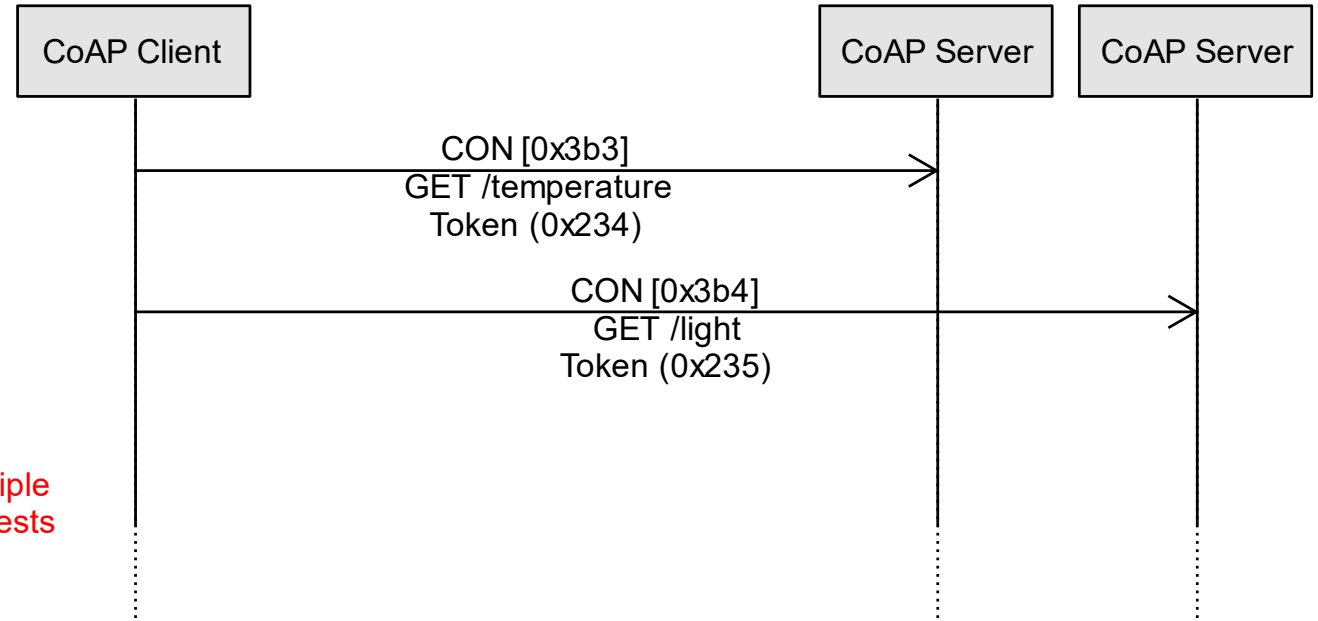
Message Lifetimes and Timeouts (3)

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InES Institute of Embedded Systems

- Multiple Pending Requests

Session 1: CoAP Basics

- Framework
- Messaging Fundamentals
- **Lifetimes and Timeouts**
- Discovery (RFC 6690)
- The Header
- Group Communication
- Observer
- Block Communication

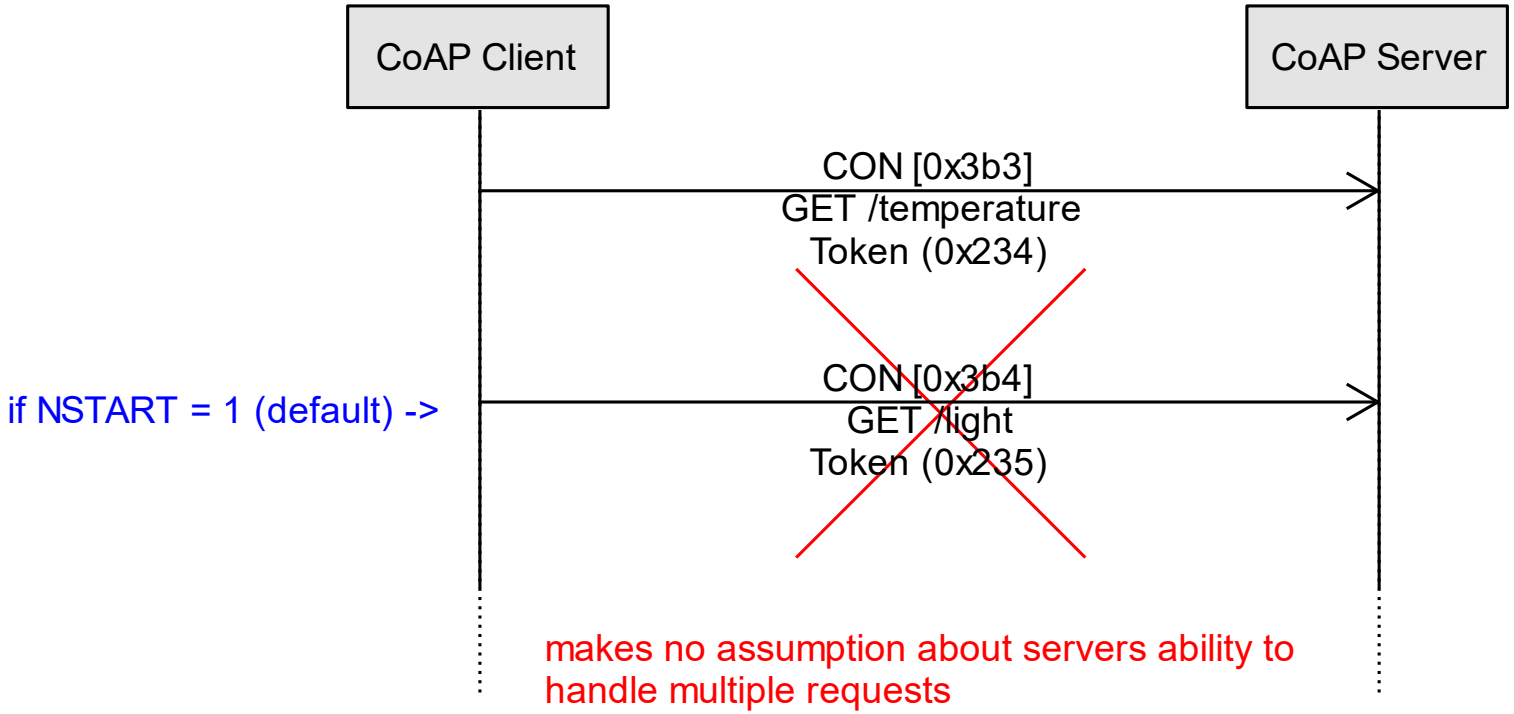


source: CoAP_11_Messaging_3.svg

Question: What mechanism allows the client to maintain multiple requests?

Message Lifetimes and Timeouts (4)

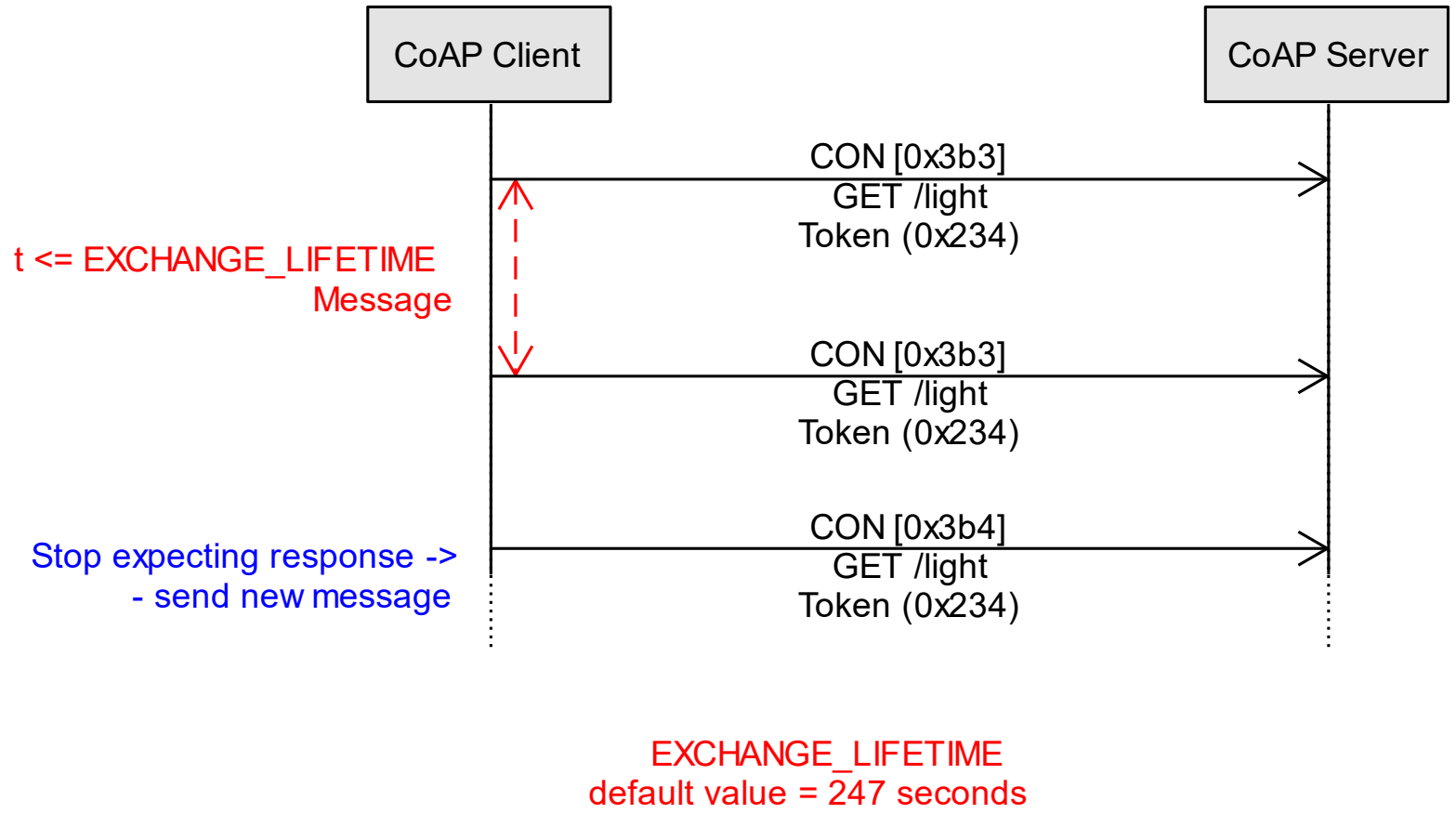
- Congestion Control «Communication Relationship»



source: CoAP_12_Messaging_4.svg

Message Lifetimes and Timeouts (5)

- Maximum Lifetime of Message



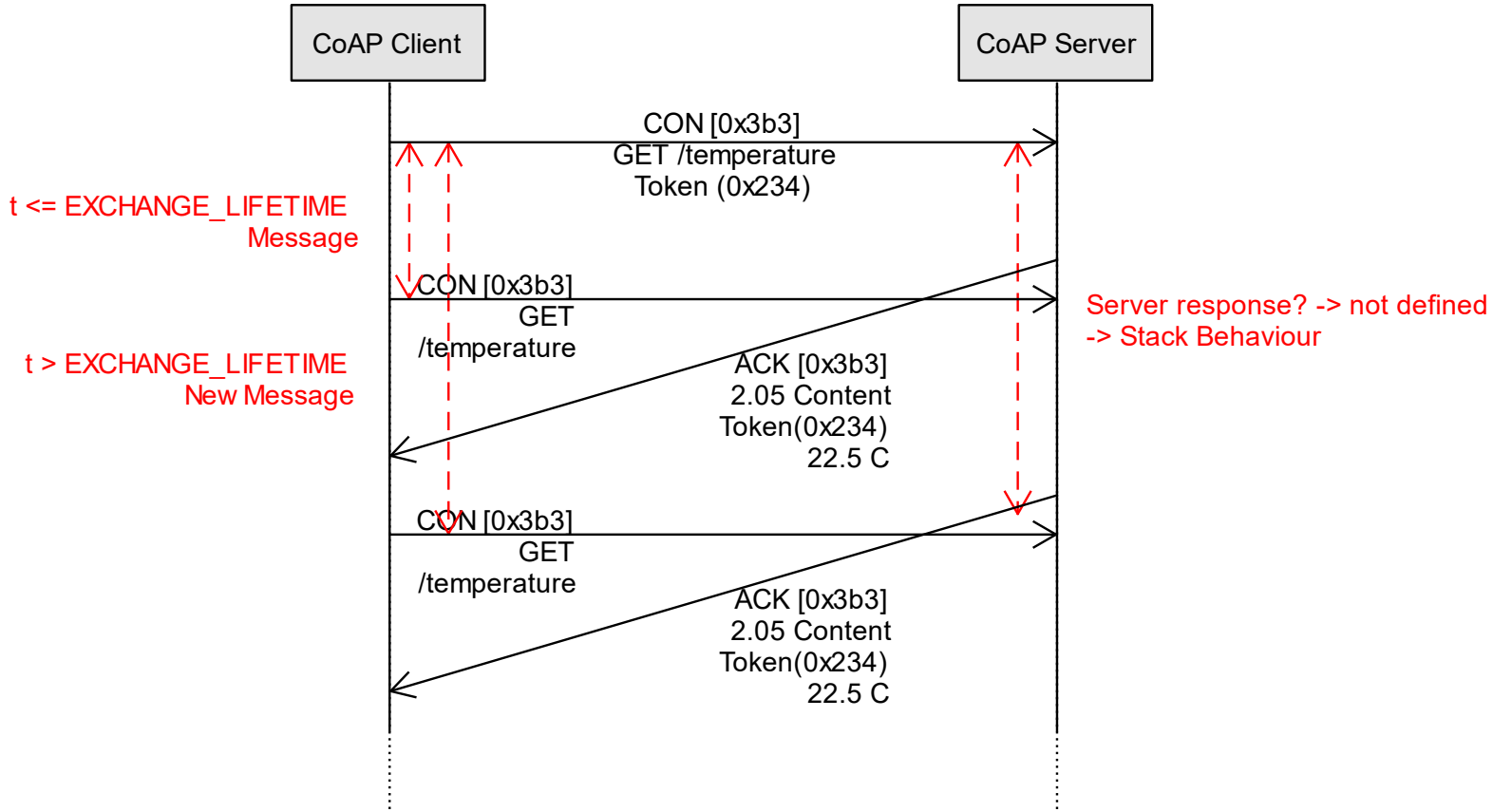
source: CoAP_13_Messaging_5.svg

Message Lifetimes and Timeouts (6)

- Maximum Lifetime of Message ID

Session 1: CoAP Basics

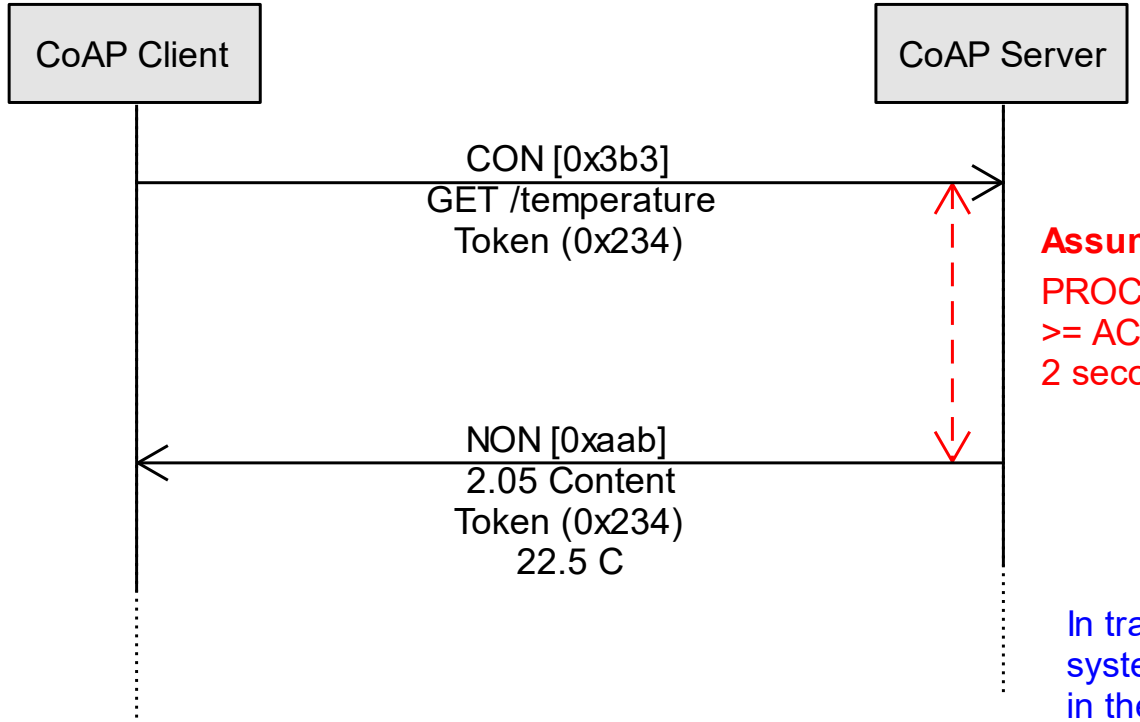
- Framework
- Messaging Fundamentals
- **Lifetimes and Timeouts**
- Discovery (RFC 6690)
- The Header
- Group Communication
- Observer
- Block Communication



source: CoAP_14_Messaging_6.svg

Message Lifetimes and Timeouts (7)

- Processing Delays



Assumption!!

PROCESSING_DELAY >= ACK_TIMEOUT = 2 seconds (default)

In traditional bus systems this is defined in the Electronic ata Sheet (EDS) and used by engineering tool

source: CoAP_15_Messaging_7.svg

Header: Specifics (1)

- Messages are matched – where appropriate – by Message ID
- Responses are matched to Requests by Tokens
- Tokens are generated by the client and must be used, unmodified by the server
- Tokens SHOULD be unique for a specific endpoint pair
  - Endpoint -> destination/source address and port
- Zero Token OK if
  - No other tokens are in use to a destination
  - Piggy-backed responses guaranteed
- A client connected to the Internet SHOULD use at least 32-bits of randomness in the token
  - Security is the reason why the token field is scalable

Header: Specifics (2)

- Options: For instance used to transfer URI information

Options

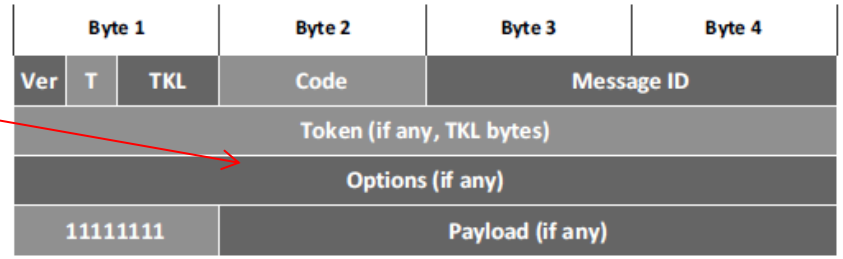


Fig. 3. Structure of the CoAP header

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

- Uri-Host, Uri-Port, Uri-Path and Uri-Query options
  - Uri-Host and Uri-Port options default to UDP header contents
  - Uri-Path contains one path segment, Uri-Query contains one argument

Header: Specifics (3)

Options: Example

Options

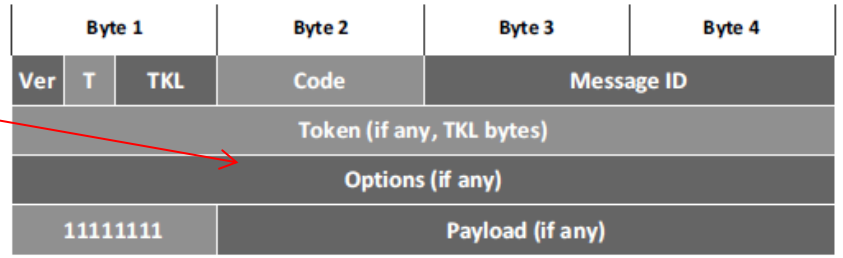


Fig. 3. Structure of the CoAP header

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

- Destination UDP port 5683
- Uri-Host = example.net
- Uri-Path = «.well-known»
- Uri-Path = «core»

Output = coap://example.net/.well-known/core

Header: Specifics (4)

- Other Options: Content-Format
- Generally the returned value is in some undetermined format
- Option Content-Format allows the server to return the encoding type of the answer in ***numerical format***
  - text/plain (01)
  - charset=utf-8 (02)
  - application/link-format (40)
  - application/xml (41)
  - application/octet-stream (42)
  - application/exi (47)
  - application/json (50)

Range	Registration Procedures
0-255	IETF Review or IESG Approval
256-2047	Specification Required
2048-64999	Expert Review
65000-65535	Experimental use (no operational use)

Header: Specifics (5)

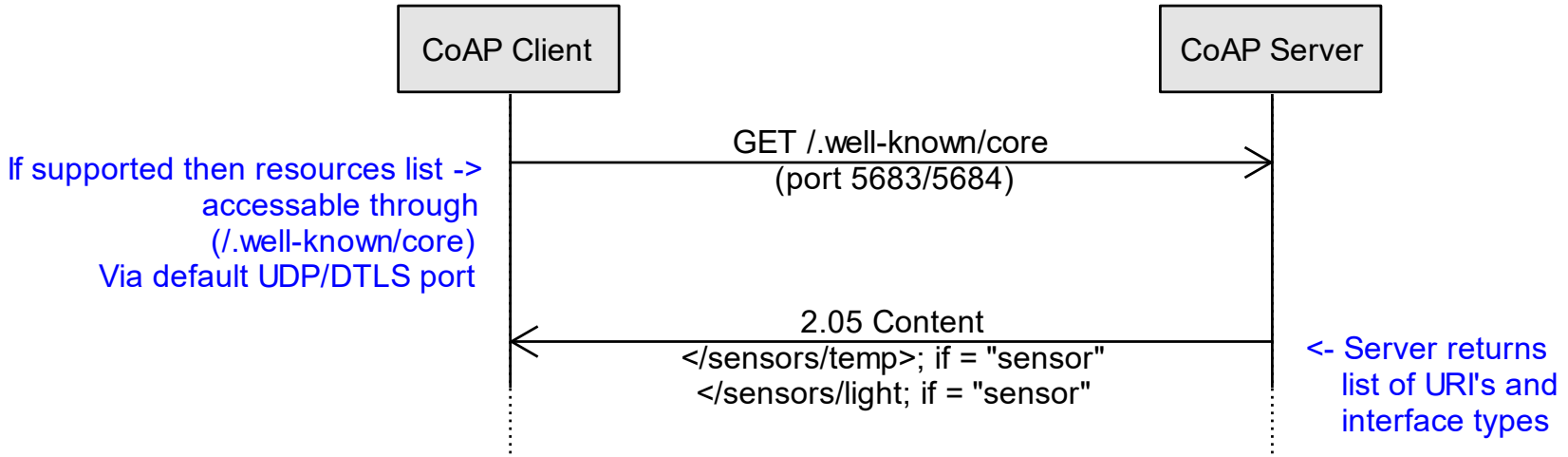
- Other Options: Accept
- If the client prefers a particular format then it can request it using the Accept option and the numeric encoding seen previously
- If available the server will return a 2.05 frame, if not then a 4.06 «Not Acceptable»
- Other options -> see RFC7252

Number	Name	Reference
0	(Reserved)	[RFC7252]
1	If-Match	[RFC7252]
3	Uri-Host	[RFC7252]
4	ETag	[RFC7252]
5	If-None-Match	[RFC7252]
7	Uri-Port	[RFC7252]
8	Location-Path	[RFC7252]
11	Uri-Path	[RFC7252]
12	Content-Format	[RFC7252]
14	Max-Age	[RFC7252]
15	Uri-Query	[RFC7252]
17	Accept	[RFC7252]
20	Location-Query	[RFC7252]
35	Proxy-Uri	[RFC7252]
39	Proxy-Scheme	[RFC7252]
60	Size1	[RFC7252]
128	(Reserved)	[RFC7252]
132	(Reserved)	[RFC7252]
136	(Reserved)	[RFC7252]
140	(Reserved)	[RFC7252]

Discovery: Basics (1)

- We know now what we get (content-format and accept) and where we get it from (Uri-x) But
  - We don't have an Electronic Data Sheet – we have to query everything
  - CoAP enables service discovery via the **RFC6690** CoRe Link Format spec

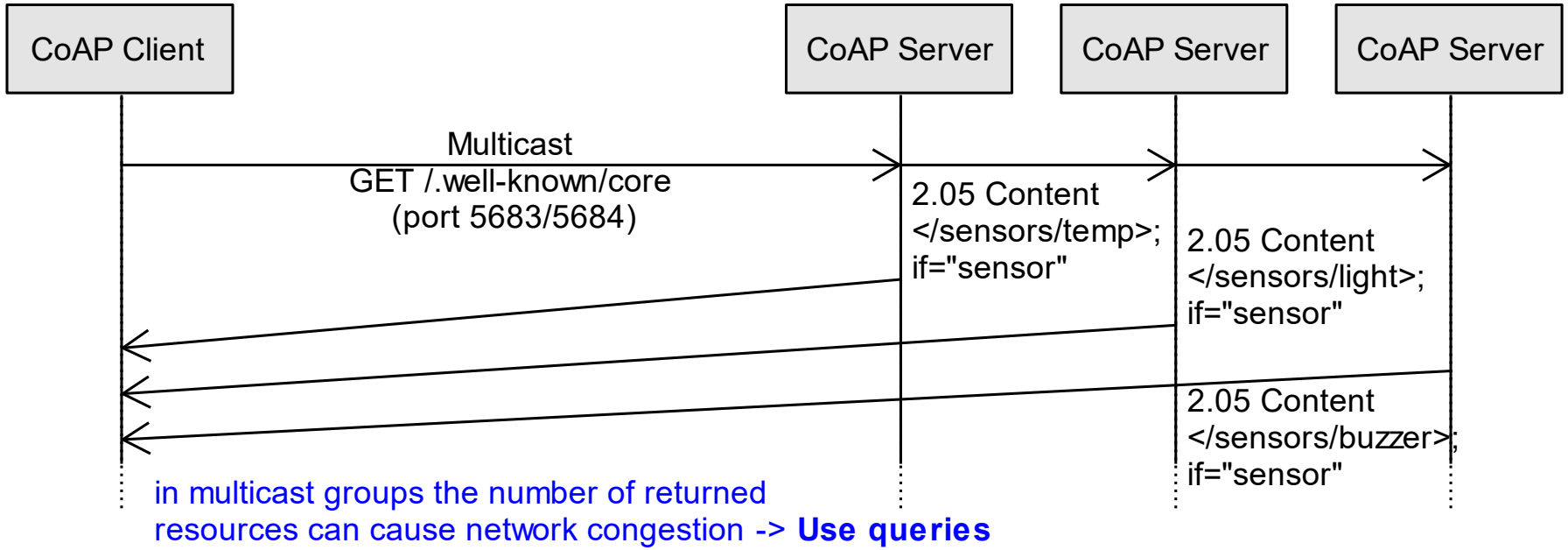
Service Discovery optional !!



source: CoAP_16_Request_Response_Particulars_6.svg

Discovery: Basics (2)

- If we don't know the URI of the servers we can use multicast CoAP



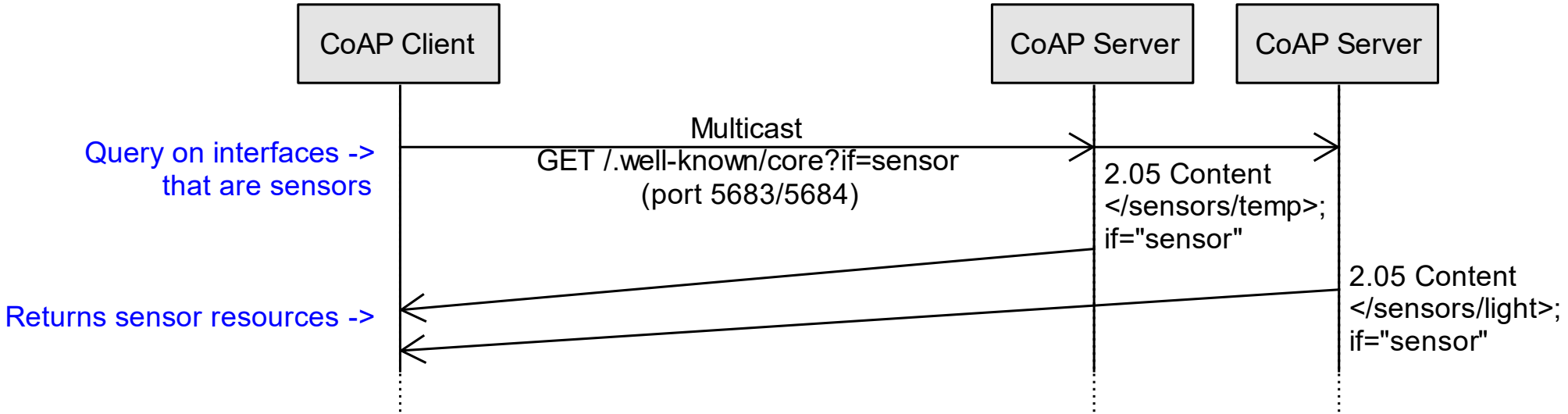
source: CoAP_17_Request_Response_Particulars_7.svg

Discovery: Basics (3)

- Using queries on various attributes

Session 1: CoAP Basics

- Framework
- Messaging Fundamentals
- Lifetimes and Timeouts
- The Header
- **Discovery (RFC 6690)**
- Group Communication
- Observer
- Block Communication



source: CoAP_18_Request_Response_Particulars_8.svg

Discovery: Basics (4)

- Other attributes:

- If = interface = superset of resource types, rt
  - Example: if = «sensor» rt: = «outdoor-temperature»

- sz = size attribute
  - Relevant if list is too large to be transferred in a single frame and block transfer is required.

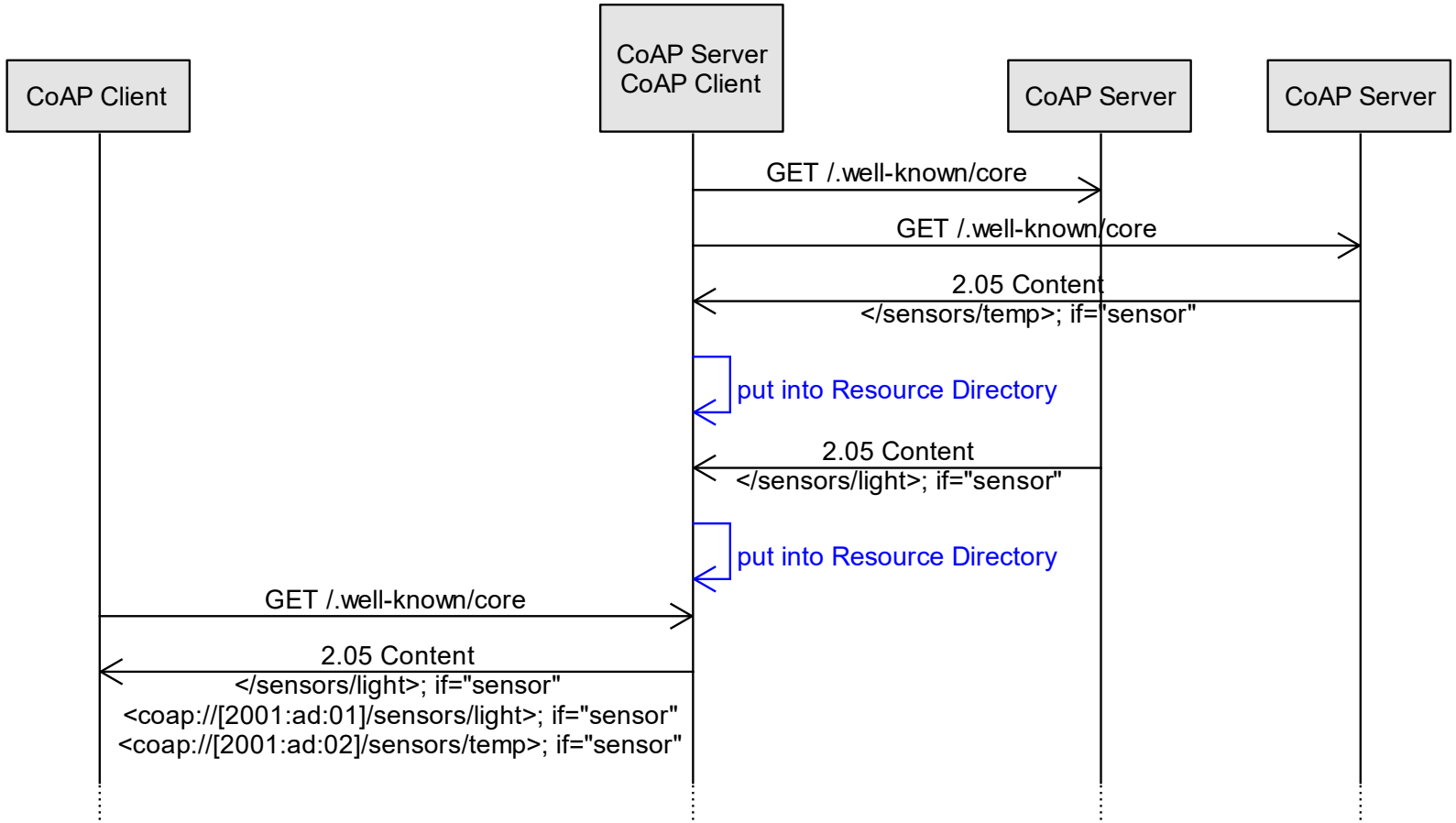
  - What situation is this realistic?

Discovery: Resource Directories (1)

- Resource Directories

Session 1: CoAP Basics

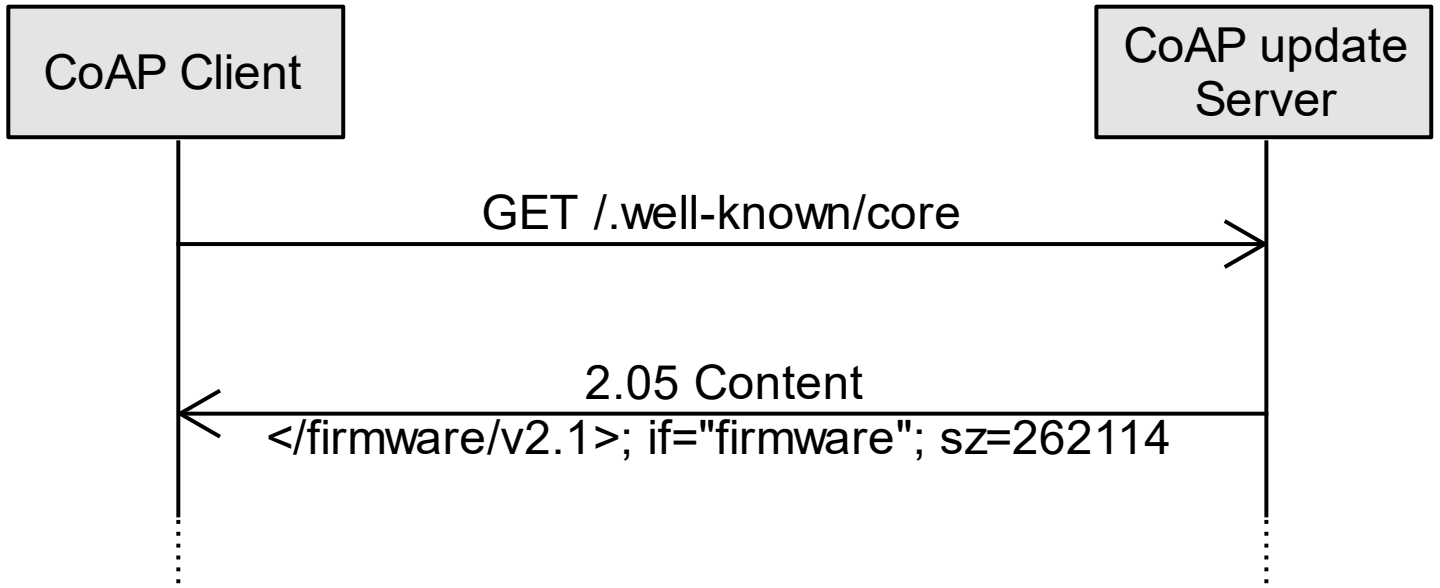
- Framework
- Messaging Fundamentals
- Lifetimes and Timeouts
- The Header
- **Discovery (RFC 6690)**
- Group Communication
- Observer
- Block Communication



source: CoAP_19_Request_Response_Particulars_11.svg

Discovery: Resource Directories (2)

- Or for instance firmware upgrades:



source: CoAP_20_Request_Response_Particulars_12.svg

Discovery: Resource Directories (3)

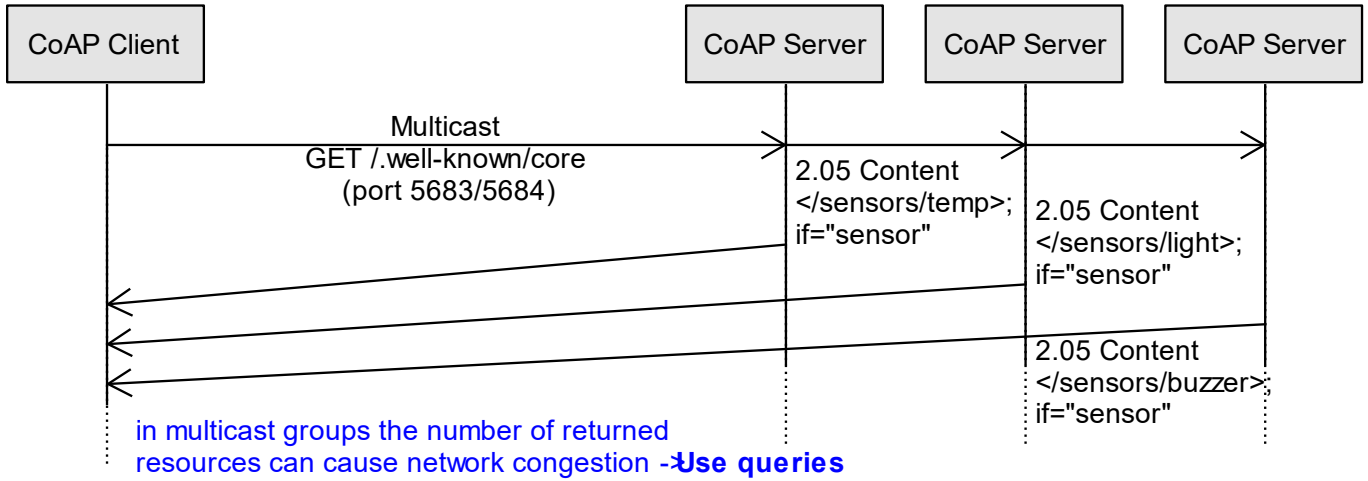
- Problem

- Attributes are not defined
  - «sensor» used in examples but «actuator» nowhere to be seen
  - «outdoor-temperature» **or** «outdoor-temp» **or** «otdr-temp»?
  - Attributes supposed to be approved and placed in registry sub-registry maintained by the ietf

- Problem
  - Attributes very coarsed grained – there is no explicit consideration of variation of sensor/actuator types and limits
  - **Solution** – define sensor profiles -> SOMETIME LATER

Group Communication: Recap Multicast

- We have seen multicasts already in the context of building a resource directory



source: CoAP_2_00_Story_3.svg

Group Communication: Basics Multicast

- In IPV4
  - unicast (one to one)
  - broadcast (one to all)
  - and multicast (one to a group)
- In IPV6 there is
  - Unicast
  - Multicast – link local
  - Multicast – site local
  - Anycast
- The basis for multicasts is the IPV6 multicast addressing capability is the reserved address space FF0x
  - Link local FF02::FD
  - Site local FF03::FD

Group Communication: Basics Groups

- For the use-case «lights-off»
- Multicast group may be defined by
  - By DNS resolvable name
  - By IP address
- Groups can be configured
  - Statically (hard-coded)
  - Per DNS
  - By a configuration tool
    - Typical in building automation
- CoAP offers the RESTfull interface
  - Defined as «Experimental» in RFC 7390

Group Communication: Storing Configurations

- The configurator knows which devices should be there
- The «system engineer» will create the basic configuration
- The configurator will PUT/POST this information to the node

POST /coap-group (format: application/coap-group+json)

{ «n»: «All-Devices.floor1.west.bldg6.example.com»,

«a»: «[ff15::4200:f7fe:ed37:abcd]:4567

}

Format of resource = JSON

Standard directory

If address («a») given then this is the default

Res: 2.01 created «/coap-group/12»

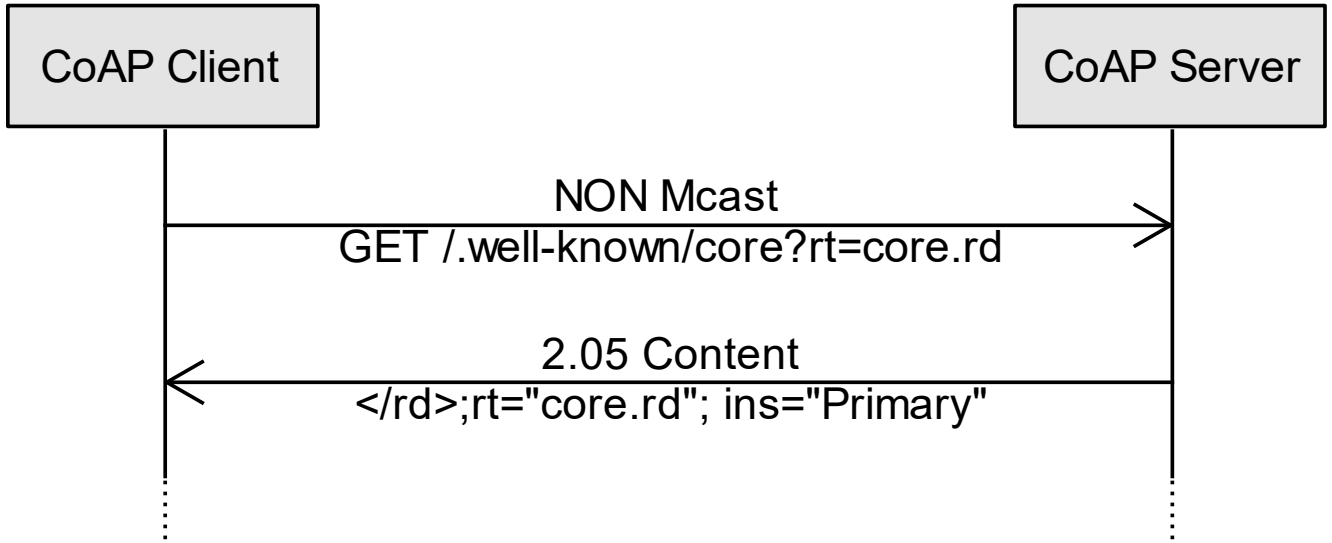
- The server is now committed to listening from traffic on port 4567 for multicast communication to the address ff15::4200:f7fe:ed37:abcd

Group Communication: Finding

- A configurator may read, per GET, information from the node

Session 1: CoAP Basics

- Framework
- Messaging Fundamentals
- Lifetimes and Timeouts
- The Header
- Discovery (RFC 6690)
- **Group Communication**
- Observer
- Block Communication



source: CoAP_2_01_Story_3_Finding.svg

Group Communication: Problems

- NO allowance for security
  - So server should not accept multicasts that have not, in some way, been authenticated.
- Congestion
  - To avoid multiple responses the server should be able to suppress responses to Multicasts with no useful content
    - For instance suppress 2.xx (success) empty responses to a lights-off command
- Required server resources?
  - Server must listen to standard UDP port and SHOULD listen to all Coap-Nodes multicast group

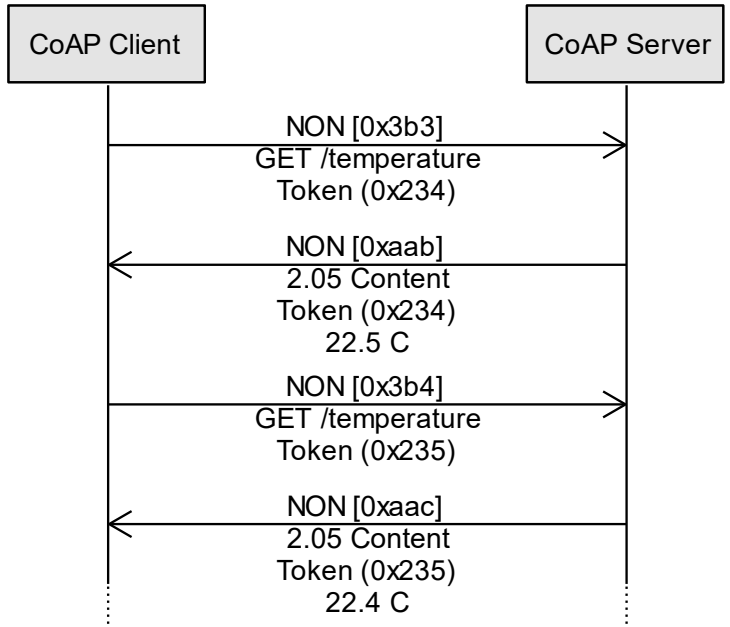
- **Q: is multicast a good option for lights on/off?**

Observer: Expectation

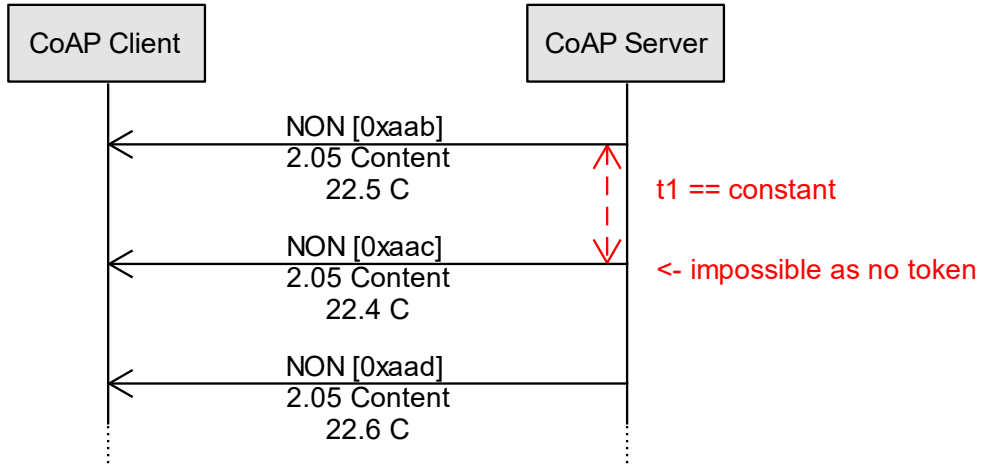
- Classical approach is to poll or transmit isochronously

Session 1: CoAP Basics

- Framework
- Messaging Fundamentals
- Lifetimes and Timeouts
- The Header
- Discovery (RFC 6690)
- Group Communication
- **Observer**
- Block Communication



source: CoAP_2_02_Story_4_Result_1.svg



source: CoAP_2_02_Story_4_Result_2.svg

Observer: Background Theory REST

- Theory:
  - REST – Representational State Transfer
  - A representation of the state of the server is transferred
    - This representation may change over time
  - The client tracks the state
    - The client must ask for updates
  - The server transfers cacheability of representation
    - Saves network bandwidth
    - Based on the idea that state of client doesn‘t change often
      - ►Example html pages
    - Idea: Get the server to transfer representation whenever the state changes!

Observer: Mechanism

- Now require an Observer and a Subject
- Subject is in the namespace of the server
- If subject changes state then it notifies the observer

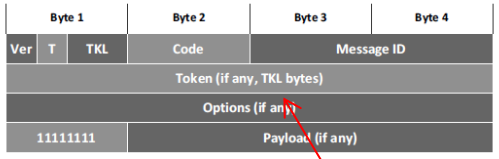
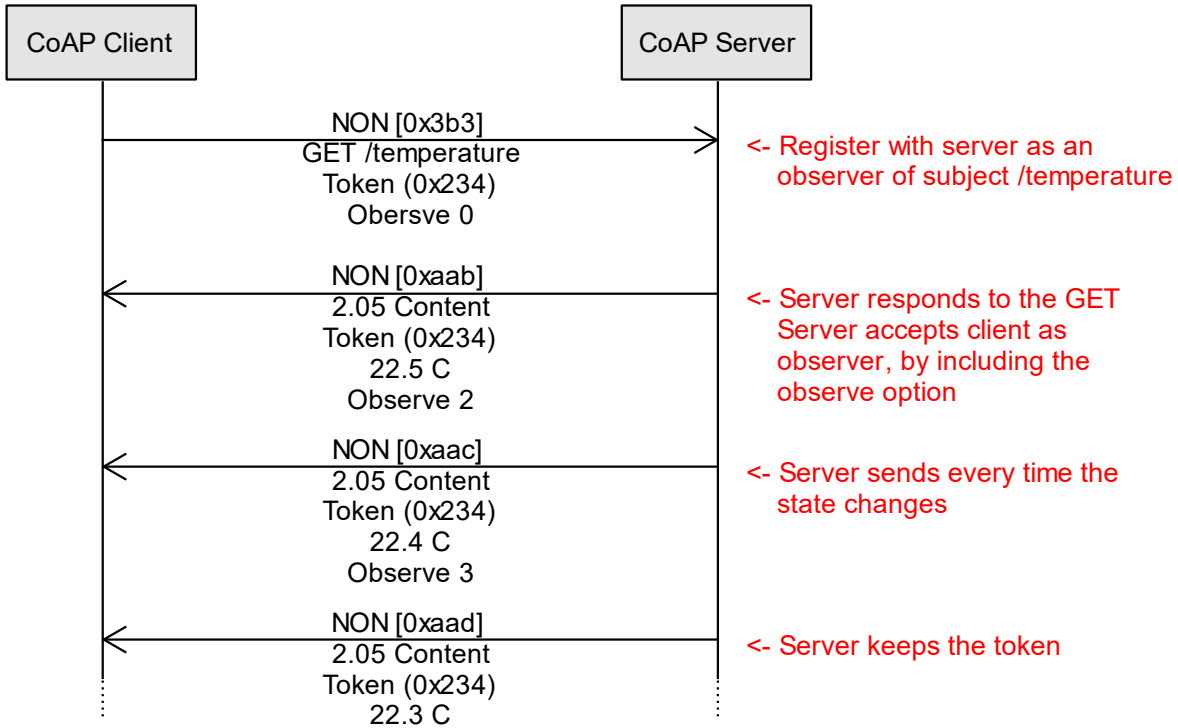


Fig. 3. Structure of the CoAP header

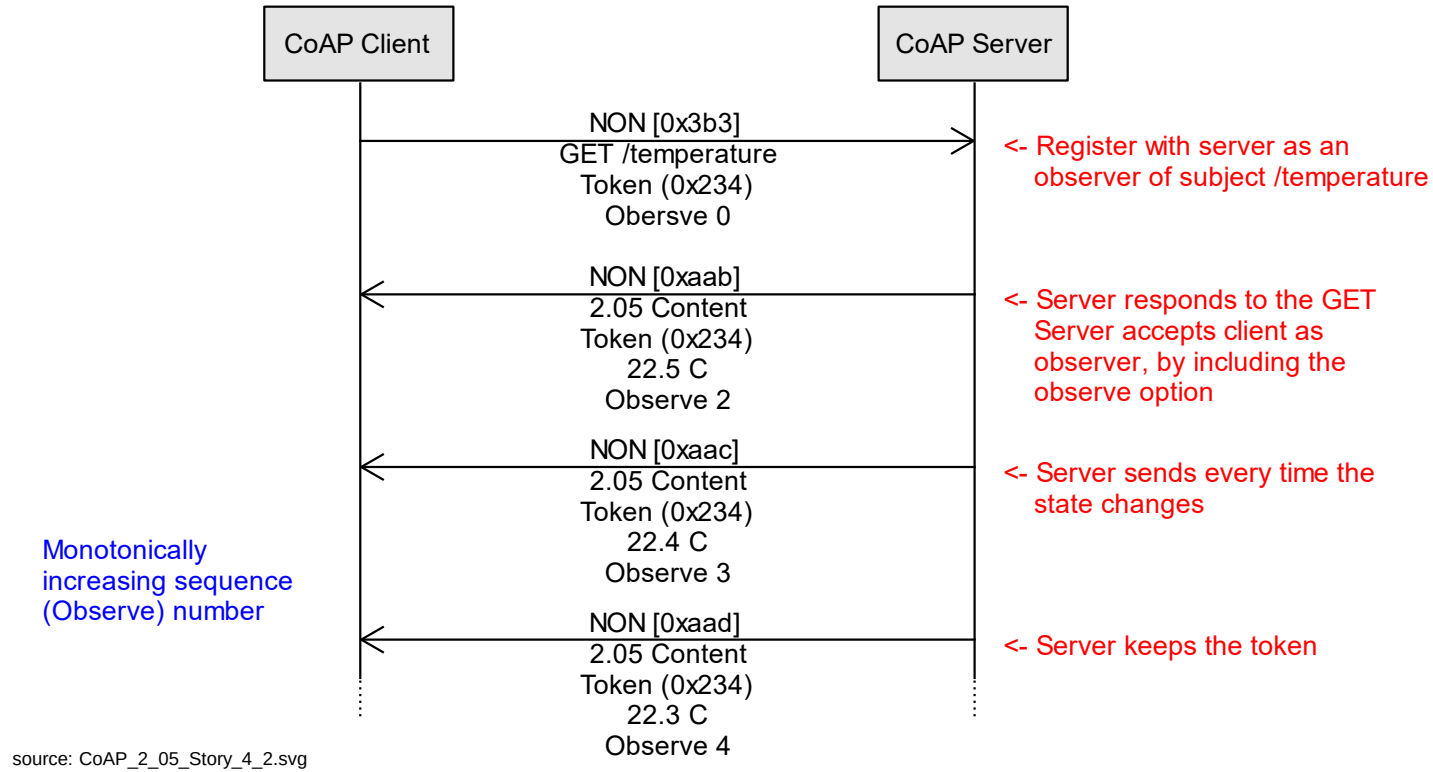
Options



source: CoAP_2_04_Story_4_1.svg

Observer: Mechanism

- Now have an Observer and a Subject
- Subject is in the namespace of the server
- If subject changes state then it notifies the observer



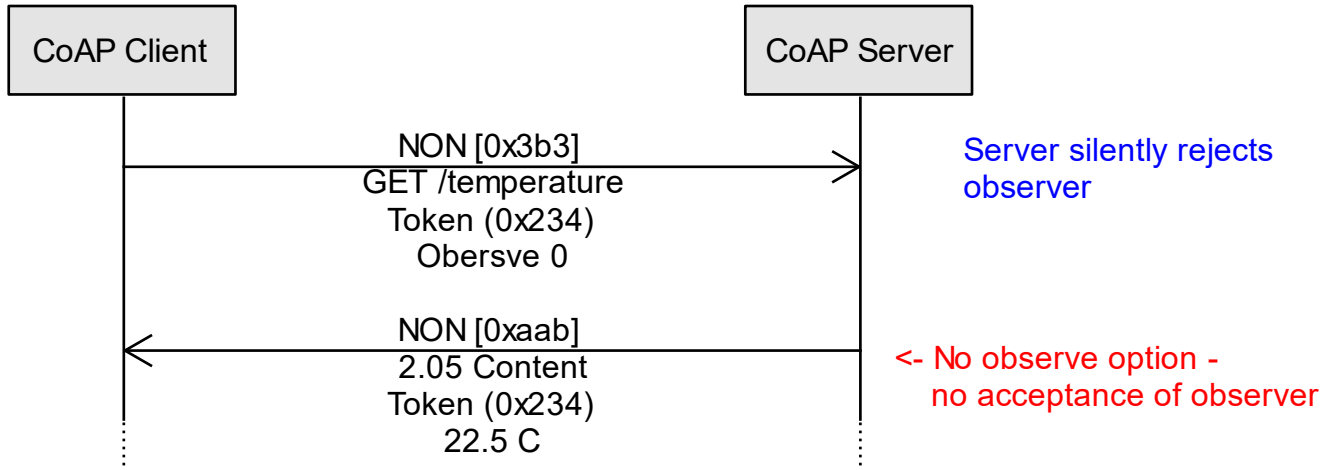
source: CoAP_2_05_Story_4_2.svg

Observer: Mechanism

- Server silently rejects observer

Session 1: CoAP Basics

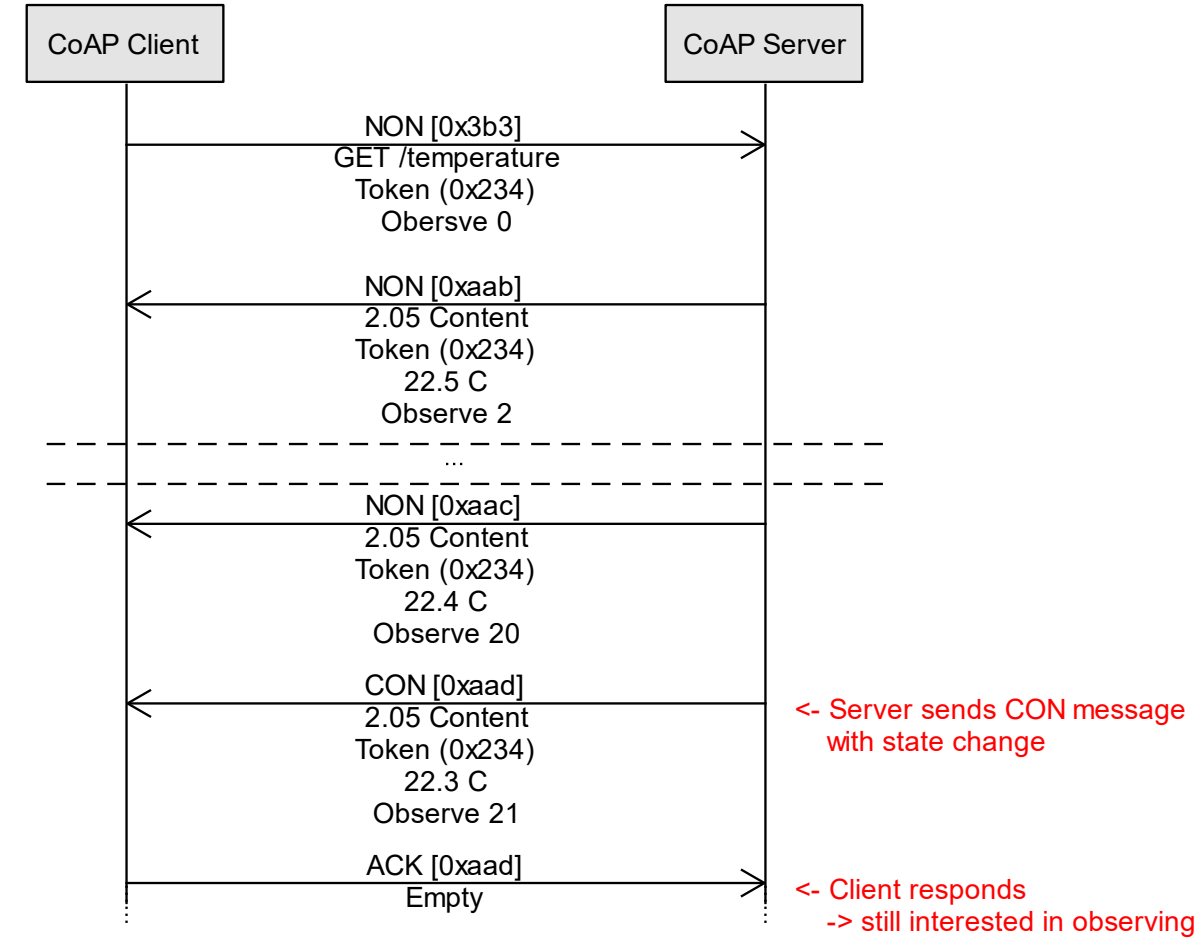
- Framework
- Messaging Fundamentals
- Lifetimes and Timeouts
- The Header
- Discovery (RFC 6690)
- Group Communication
- **Observer**
- Block Communication



source: CoAP_2_06_Story_4.svg

Observer: Mechanism

- Server can transmit NON or CON messages



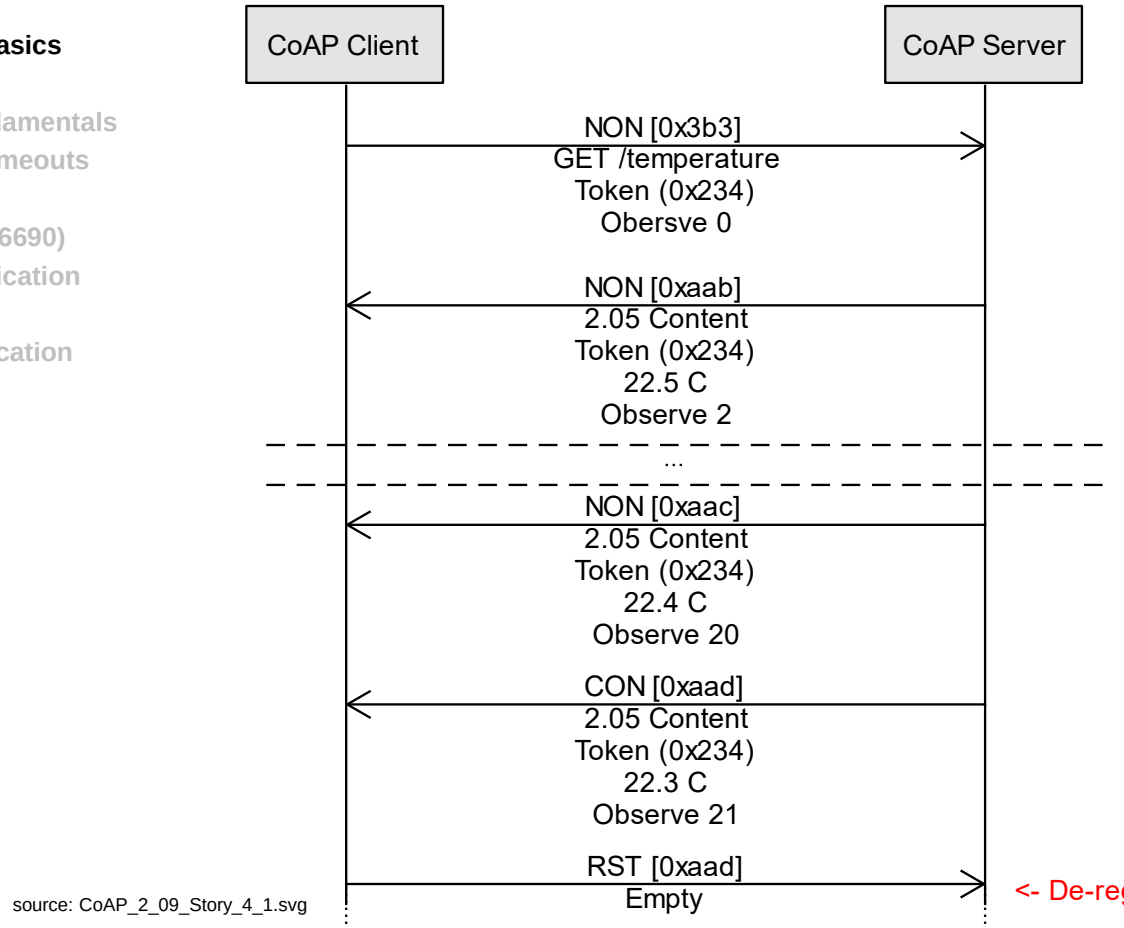
source: CoAP_2_07_Story_4.svg

Observer: Mechanism

- RST or ignore means I'm not interested as does de-register

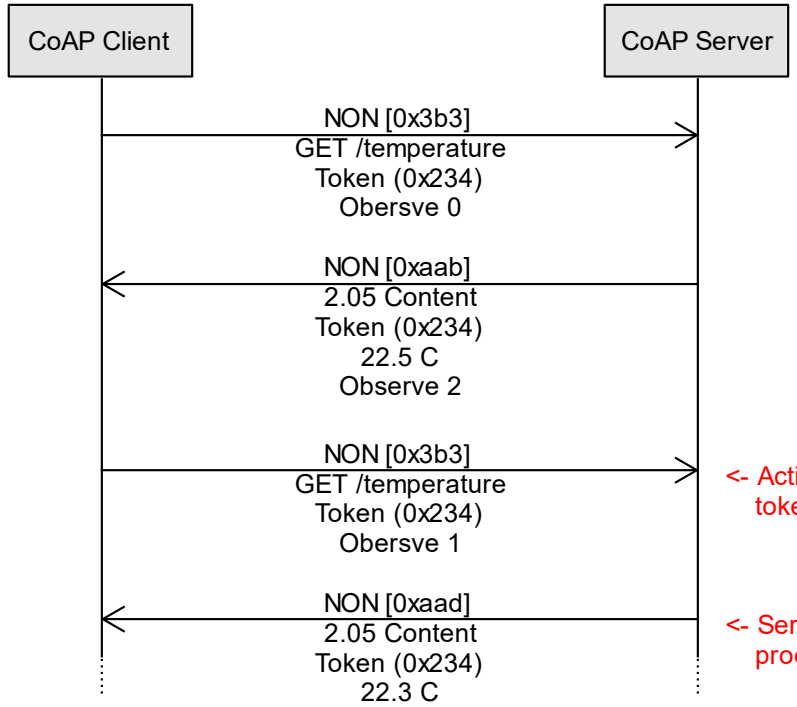
Session 1: CoAP Basics

- Framework
- Messaging Fundamentals
- Lifetimes and Timeouts
- The Header
- Discovery (RFC 6690)
- Group Communication
- **Observer**
- Block Communication



source: CoAP_2_09_Story_4_1.svg

<- De-register on server enquiry



<- Actively de-register with token and option set to 1

<- Server removes observer and processes as normal request

source: CoAP_2_09_Story_4_2.svg

Observer: Discussion

- Is a notification every 0.1 degree Celsius useful?
  - Decidedly not
  - Server decides when notifications are sent
  - Server may expose resources for instance
  - <coap://server/temperature>
    - state change every time the temperature changes
  - <coap://server/termperature/felt>
    - Changes state to «cold» every time the temperature drops below pre-configured level
  - <coap://server/temerpature/critical?above=42>
    - OK if temperature drops below 42 degrees otherwise transmit every status change
  - Either pre-defined by the server or – possibly - configurable

Block Communication: Response

Zürcher Hochschule für Angewandte Wissenschaften

School of Engineering
InES Institute of Embedded Systems

- Block Transfer. In brief: Block2 pertains to response payloads

Session 1: CoAP Basics

- Framework
- Messaging Fundamentals
- Lifetimes and Timeouts
- The Header
- Discovery (RFC 6690)
- Group Communication
- Observer
- **Block Communication**

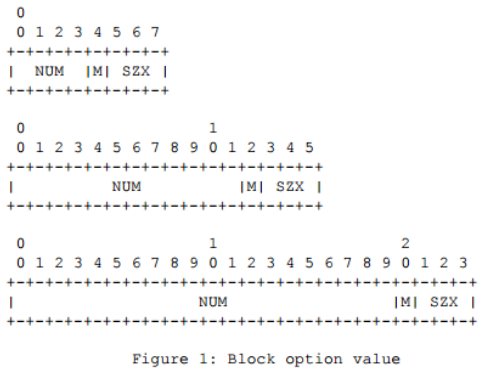
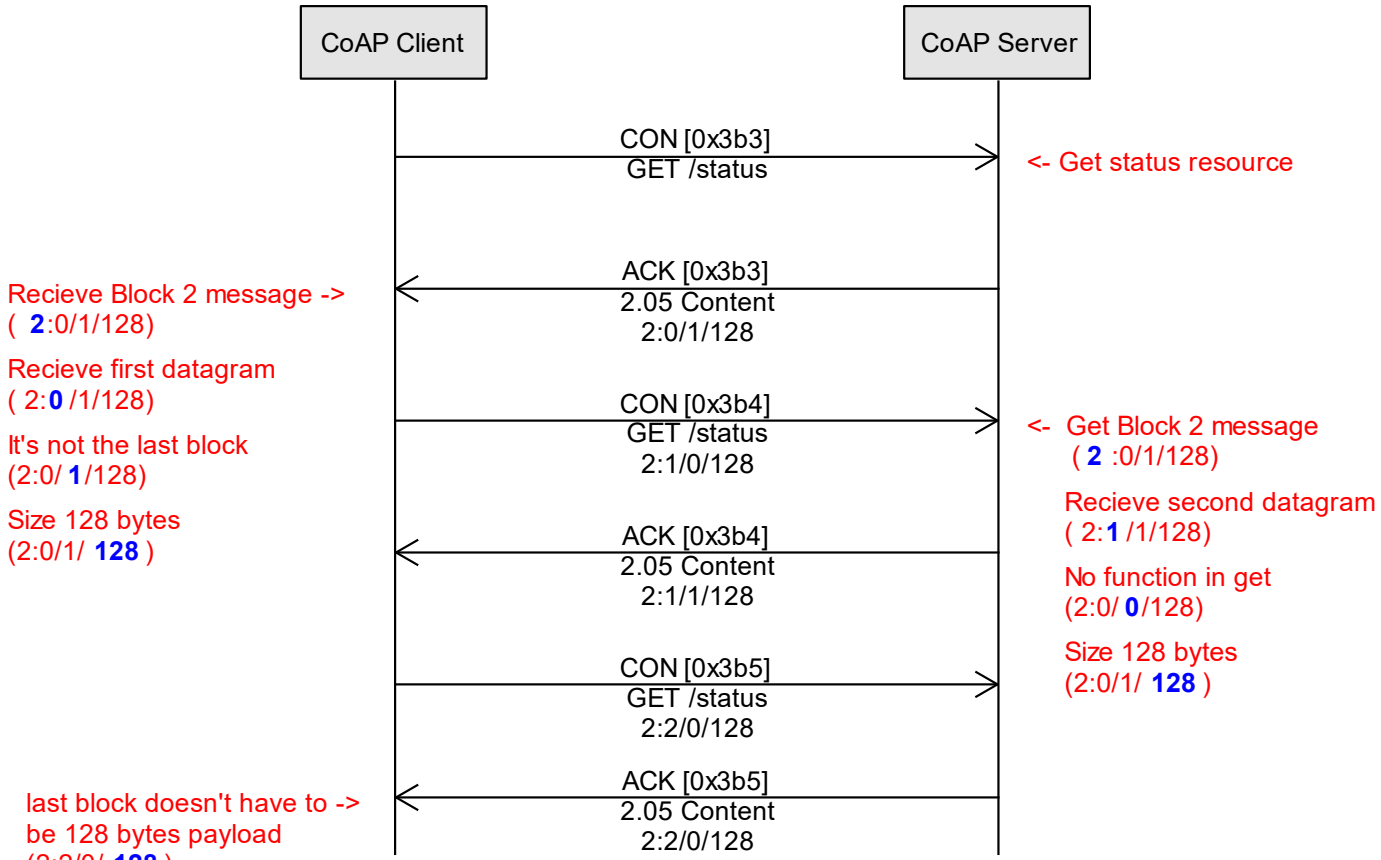


Figure 1: Block option value



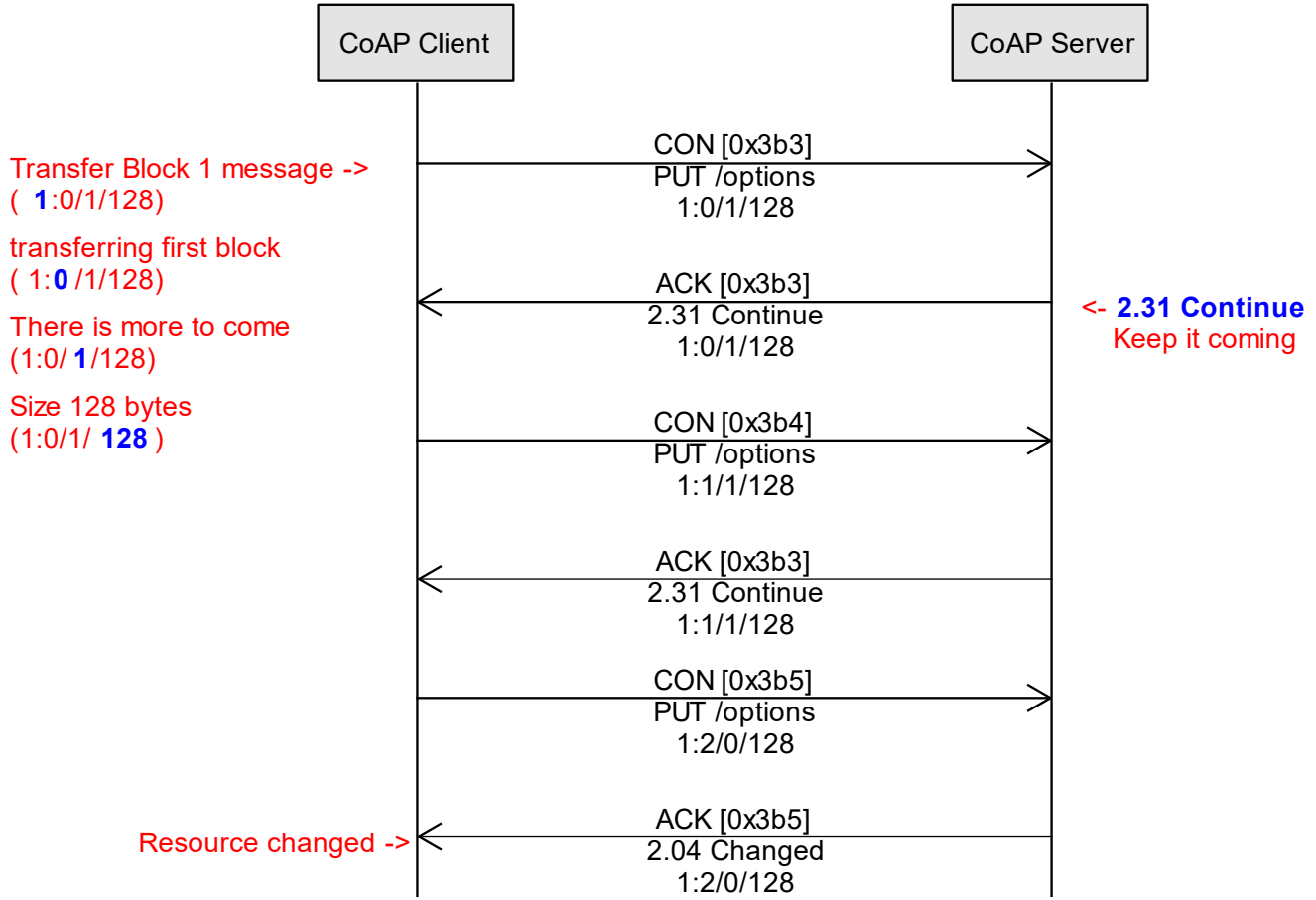
source: CoAP_2_10_Story_5.svg

Block Communication: Requests

- Block 1 transfers pertain to requests

Session 1: CoAP Basics

- Framework
- Messaging Fundamentals
- Lifetimes and Timeouts
- The Header
- Discovery (RFC 6690)
- Group Communication
- Observer
- **Block Communication**



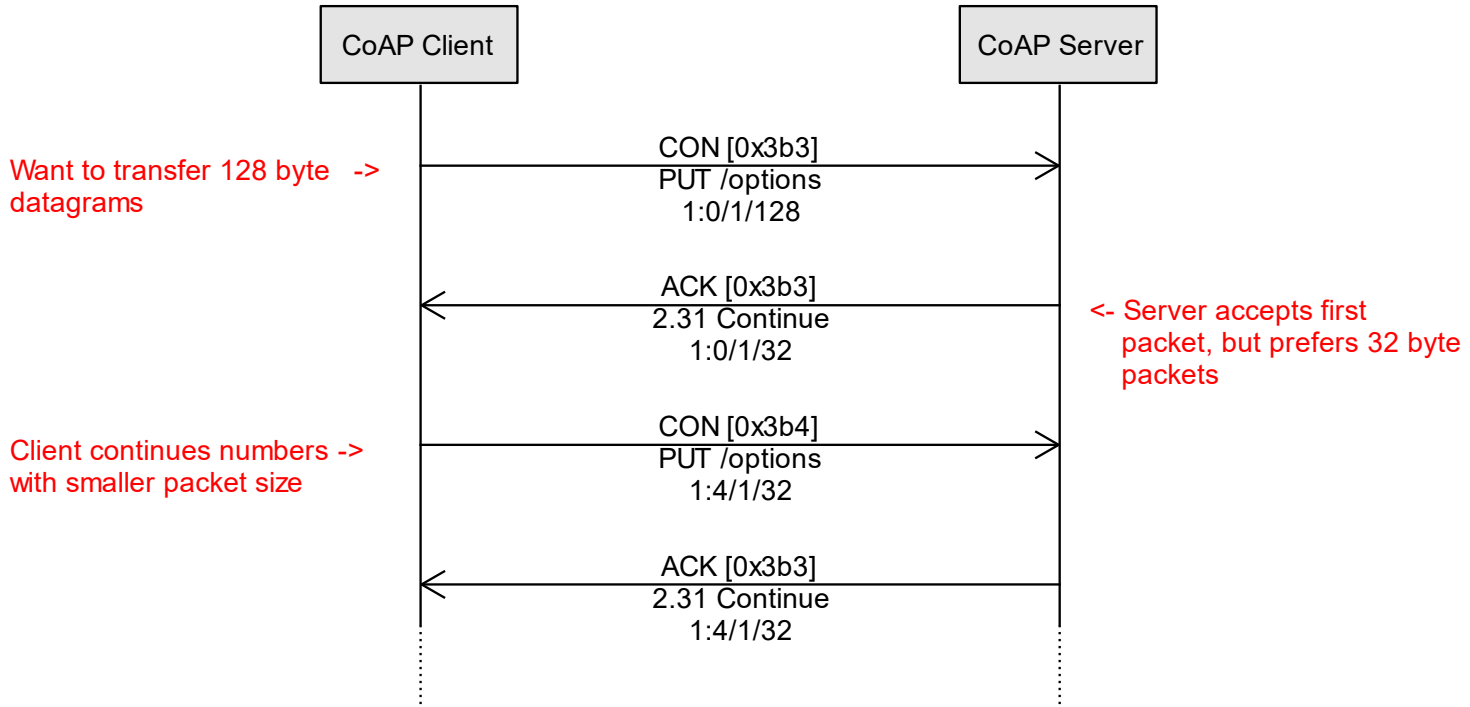
source: CoAP_2_11_Story_5.svg

Block Communication: Size Negotiation

- Can also negotiate

Session 1: CoAP Basics

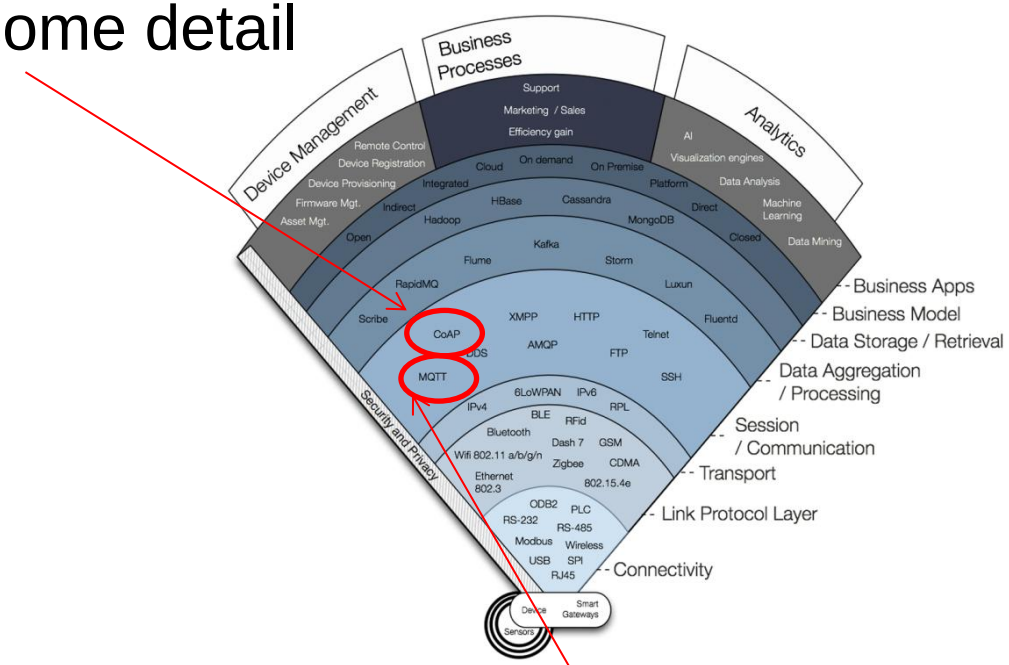
- Framework
- Messaging Fundamentals
- Lifetimes and Timeouts
- The Header
- Discovery (RFC 6690)
- Group Communication
- Observer
- **Block Communication**



source: CoAP_2_12_Story_5.svg

Conclusion

- We have handled CoAP in some detail



- We have hinted at a publisher subscriber version of CoAP but will handle the subject matter next week using the MQTT protocol as an example

http://www.internet-of-things-research.eu/pdf/IoT-From%20Research%20and%20Innovation%20to%20Market%20Deployment_IERC_Cluster_eBook_978-87-93102-95-8_P.pdf page 36