Application level of IoT networks -- Part 1

Rafik Zitouni

ECE Paris

rafik.zitouni@ece.fr

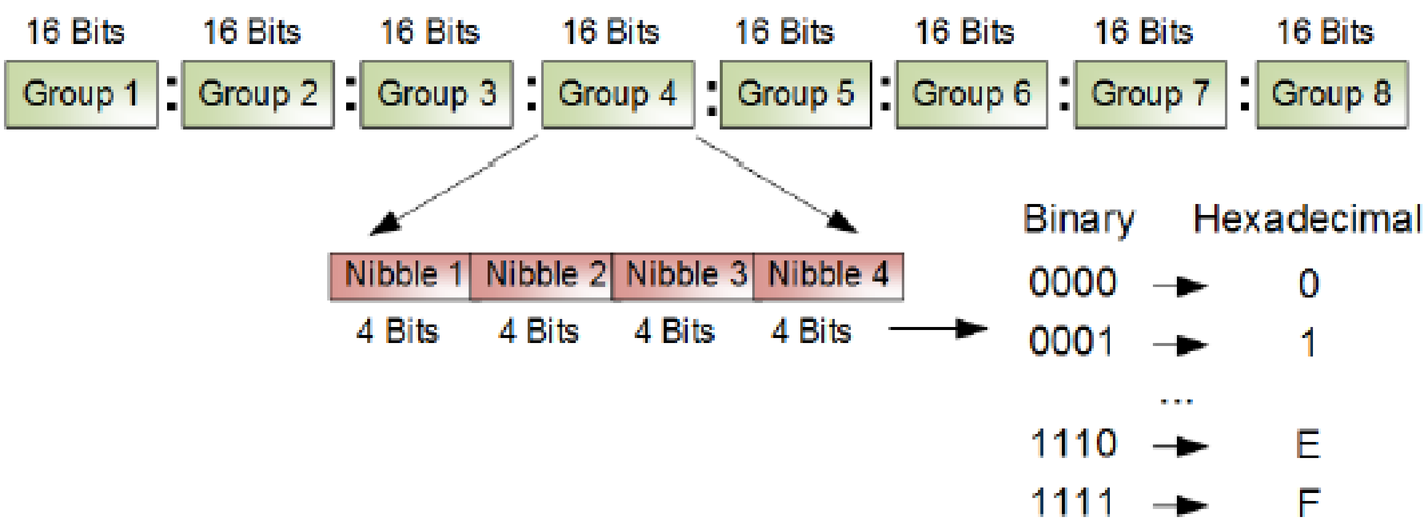
ECE PARIS
ÉCOLE D'INGÉNIEURS

Outline

- IPV6 and 6LowPAN
- CoAP protocol
- Quality of Service (QoS)
  - MQTT
  - DDS, AMQP, XMPP .etc

IPv6 – Overview

- Provides many more addresses, to satisfy current and future needs, with ample space for innovation. 2^{128} addresses
- **40-byte** IP header
- Ability to do end-to-end IPsec (IP security)
- **Unicast (one to one)**, **Multicast (one to many)**, **Anycast (one to nearest)** and Reserved (special uses of some addresses)



IPv6 – Overview

Example 1:

3FFE:085B:1F1F:0000:0000:0000:00A9:1234

Leading zeros can be removed

3FFE:85B:1F1F::A9:1234

:: = all zeros in one or more group of 16-bit hexadecimal numbers

2001:db8:A:0:0:12:0:80

a) 2001:db8:A::12:0:80
b) 2001:db8:A::12::80
c) 2001:db8:A:0:0:12::80

IPv6 – Overview

Network prefix

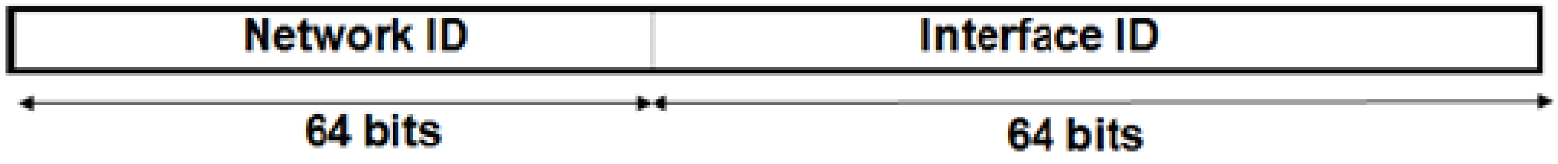
1) Example of compressed form :

2001:0db8:0001:0000:0000:0000:0000:0000 is 2001:db8:1::/48

The first **48 bits** will always be the same **2001:0db8:0001**

2) /64 prefix is always used in a LAN (Local Area Network)

- Rightmost 64 bits are called the interface identifier or host's interface
- Left part defines the network identifier



IPv6 – Overview

Unspecified address → 0:0:0:0:0:0:0:0 (::/128)
Loopback address → 0:0:0:0:0:0:0:1 (::1/128)
Documentation Prefix → 2001:db8::/32 (used in examples and documentation)

Link-local: addresses with prefix **FE80::/10.** They are used to communicate with other hosts on the same local network.

ULA (Unique Local Address) start with the prefix **FC00::/7**, which is in practice means that you can see **FC00::/8** or **FD00::/8**

Global Unicast: Equivalent to the IPv4 public addresses,

IPv6 – Overview

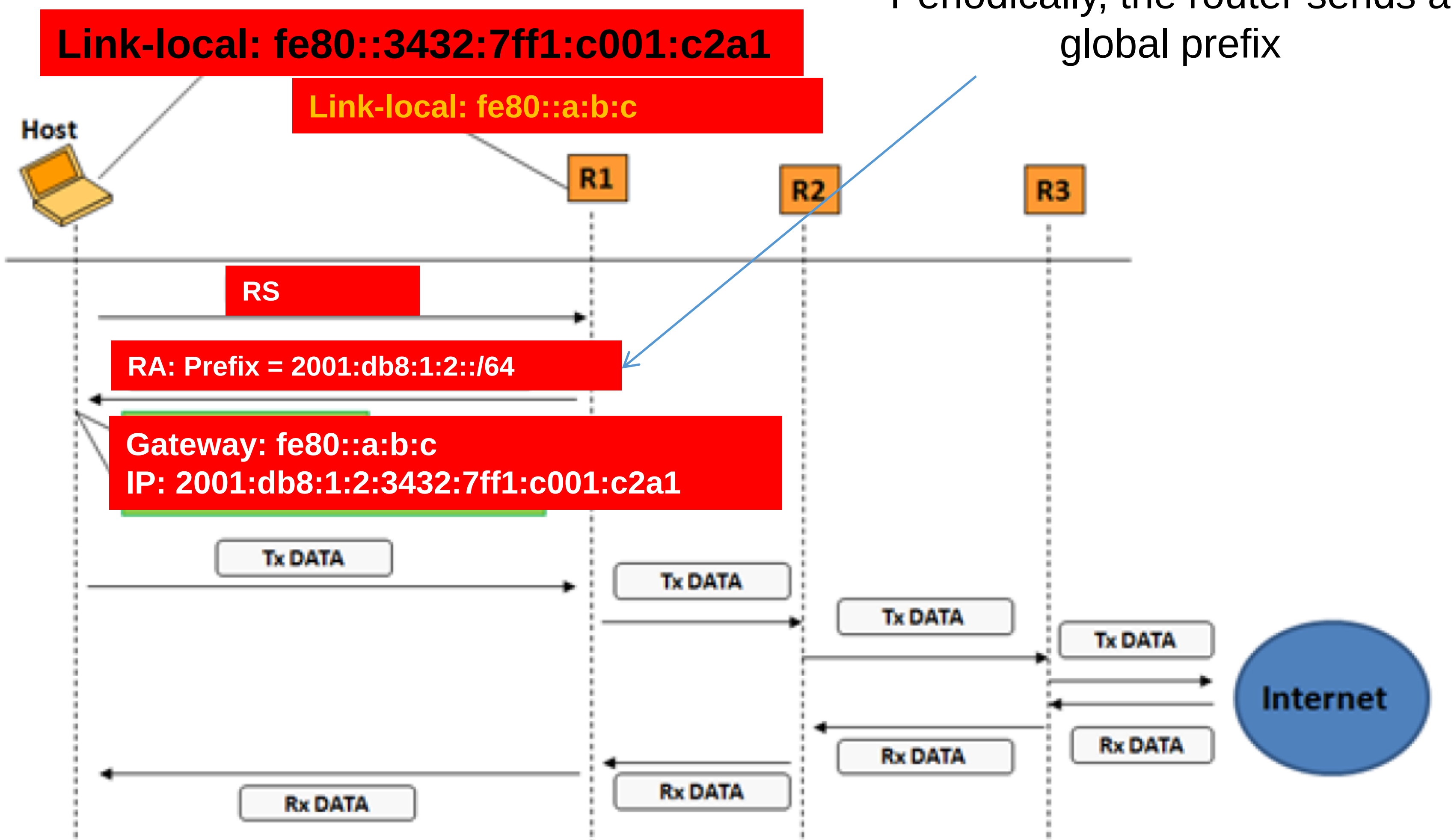
IPv6 can be configured :

- Statically like IPv4
- **DHCPv6** (Dynamic Host Configuration Protocol IPv6) like DHCPv4
- **SLAAC** (StateLess Address AutoConfiguration) : New mechanism to configure automatically all network parameters
  - « Plug and net »

IPv6 – Overview

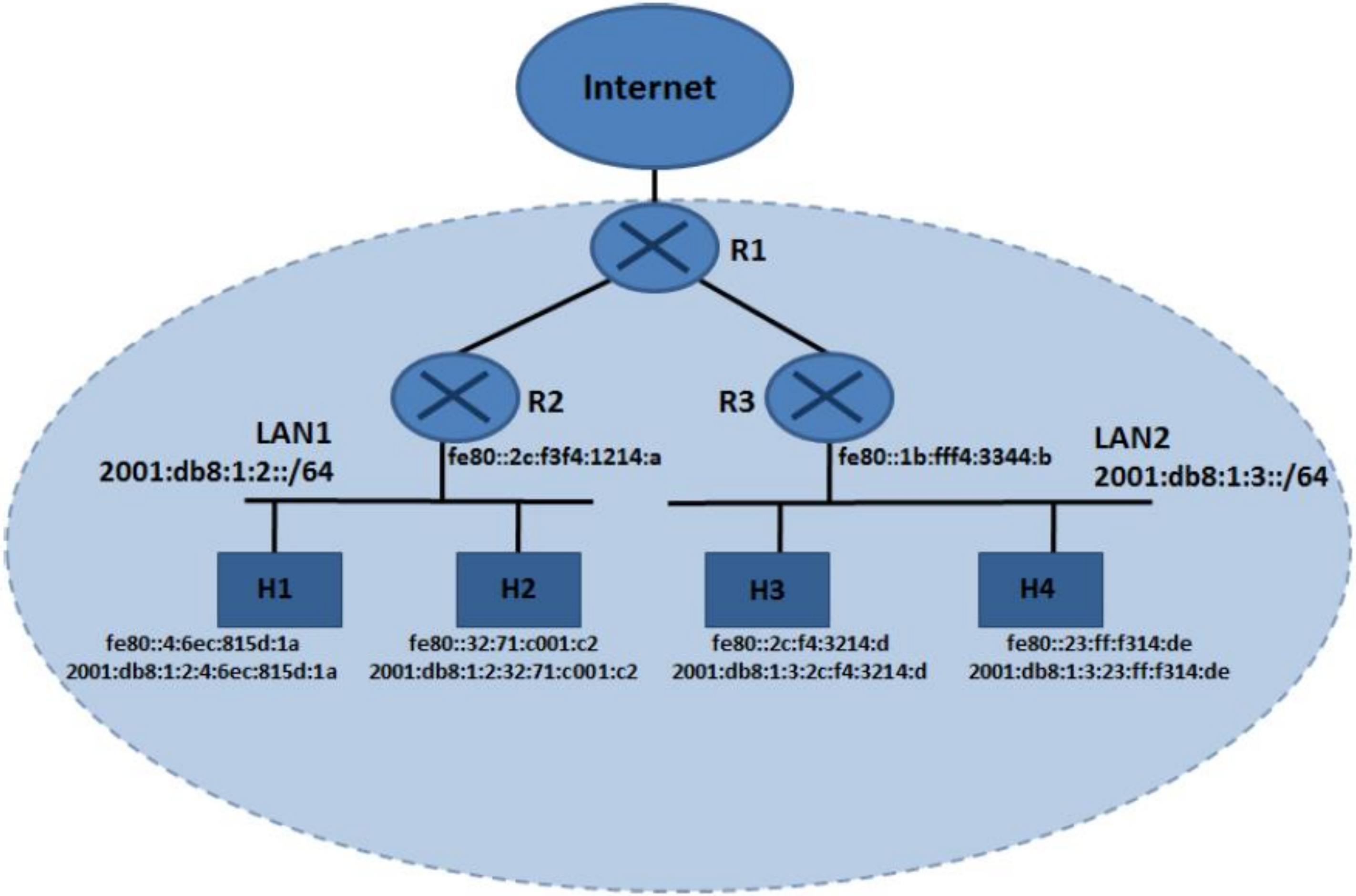
SLAAC mechanism

Periodically, the router sends a global prefix



IPv6 – Overview

Example of an IPv6 Network



IPv6 – Overview

Three cases with IPv6 addressing

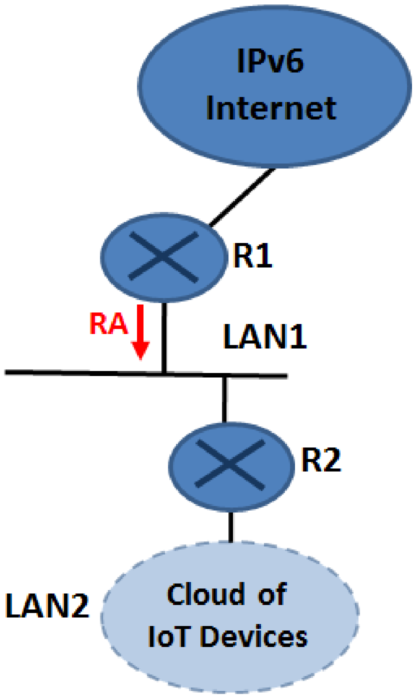
Case 1:
Native IPv6 connectivity: both router and ISP support IPv6 addressing→IPv4 and IPv6 are supported dual stack

Case 2:
No IPv6 connectivity: ISP doesn't support IPv6 but IPv6 is supported by router → IPv6 transition mechanism → **6in4 tunnel**

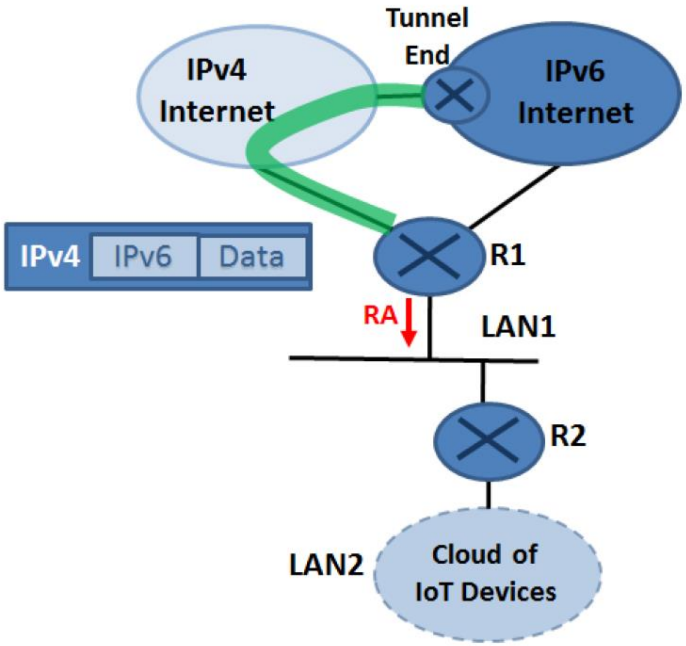
Case 3:
No IPv6 connectivity and No IPv4 capable router: → Add a **new router** that support IPv6 and IPv4 and **create 6in4 tunnel**

IPv6 – Overview

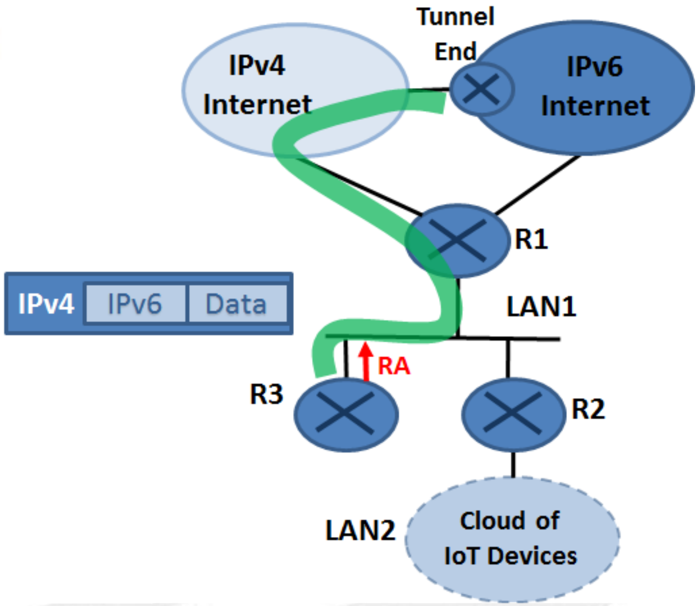
Case 1:



Case 2:



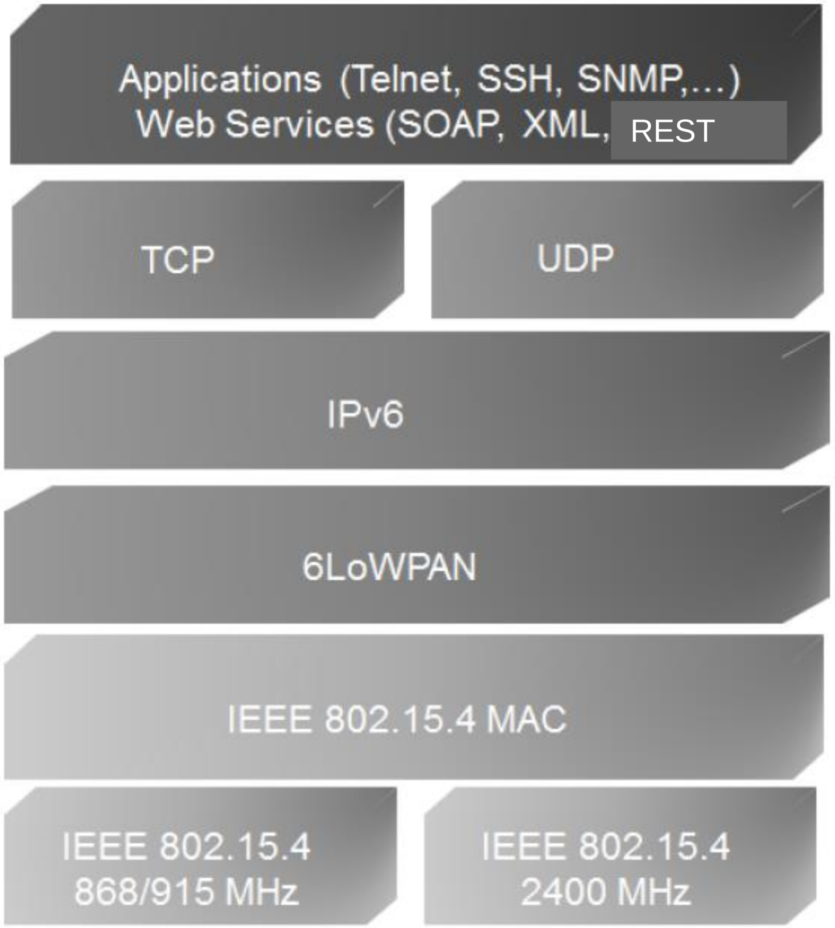
Case 3:



6LowPAN

6LowPAN: IPv6 over low Power Wireless Personal Area Networks

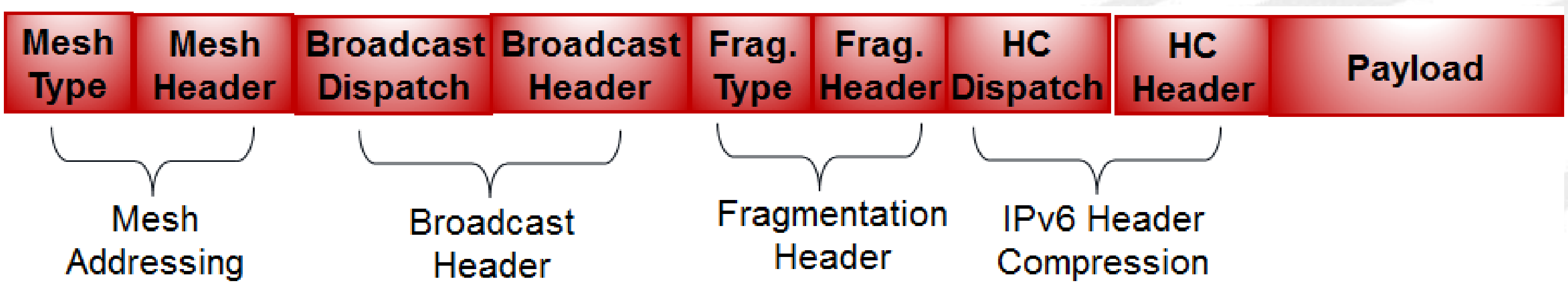
- It defines IPv6 over IEEE 802.15.4 standard



6LowPAN

Main proprieties of 6LowPan

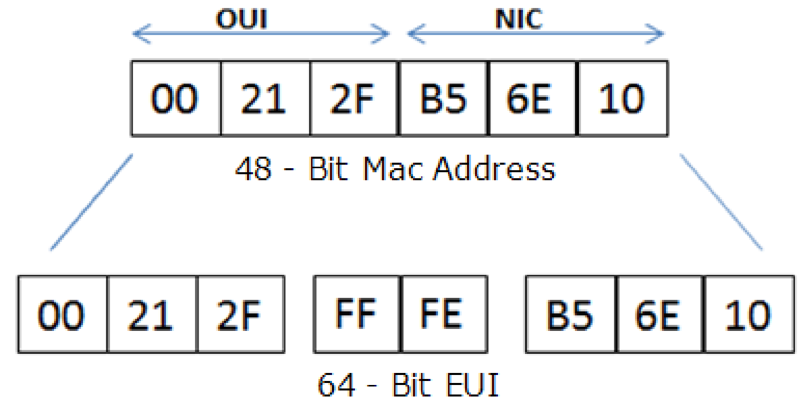
- The maximum size available for transmitting IP packets over an IEEE 802.15.4 frame is **81 bytes** (payload)
- The minimum MTU (Maximum Transmission Unit) that link layer should offer to IPv6 layer is **1280 bytes**
- Mesh Routing Protocol [RFC 6550]: **RPL** (Routing protocol for Low Power and Lossy Networks)
- IEEE 802.15.4 defines **four types** of frames:
    - Beacon frames
    - MAC command frames
    - Acknowledgement frames
    - Data frames



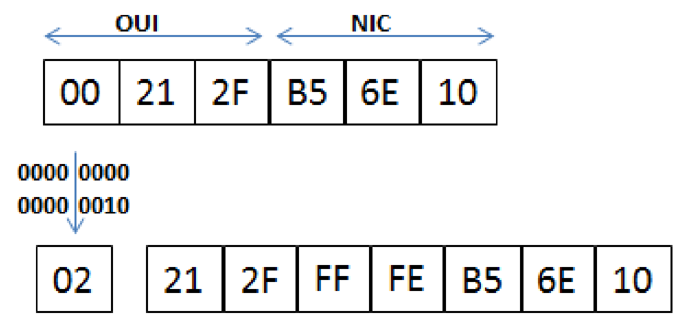
6LowPan headers

6LowPAN – Link-local address

IPv6 Interface Identifier (IID) from EUI-48 or MAC address



7th bit from the left is flipped



The generated IPv6 address is : **FE80::**0221:2FFF:FEB5:6E10

Link local prefix

CoAP Constrained Application Protocol

Why the web has the actual success ?

- Uniform representation of documents → **HTML**
- Uniform Referents for Data and Services on the Web → **URI**
- Universal Transfert Protocol → **HTTP**
- Representational State Transfert Architecture → **REST**

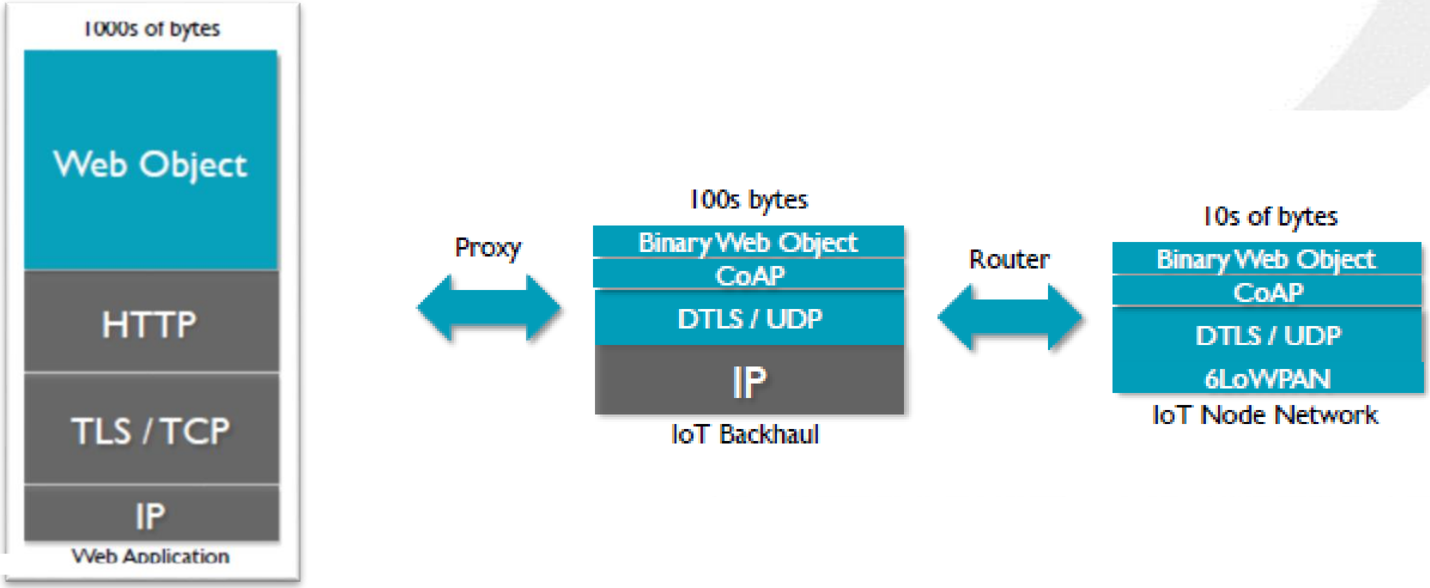
Is it possible to establish web communication with constrained 8/16-bit Microcontrollers ?

- TCP as the Transport Protocol, too heavy for LLN motes;
- SSL/TLS for security: too heavy;

CoAP Constrained Application Protocol

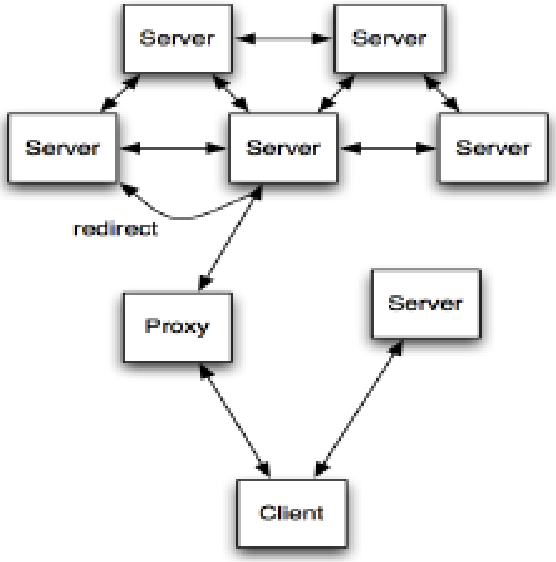
Characteristics (RFC 7252 https://tools.ietf.org/html/draft-ietf-core-coap-18)

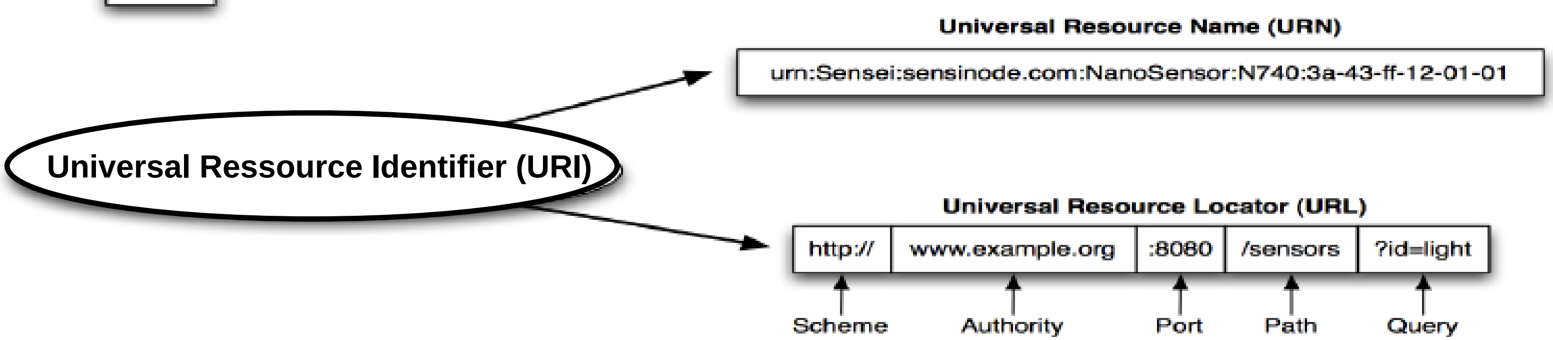
- Constrained machine-to-machine web protocol
- Representational State Transfer (REST) architecture
- UDP binding (may use IPsec or DTLS)
- Asynchronous message exchanges
- Simple proxying and caching capabilities



CoAP Constrained Application Protocol

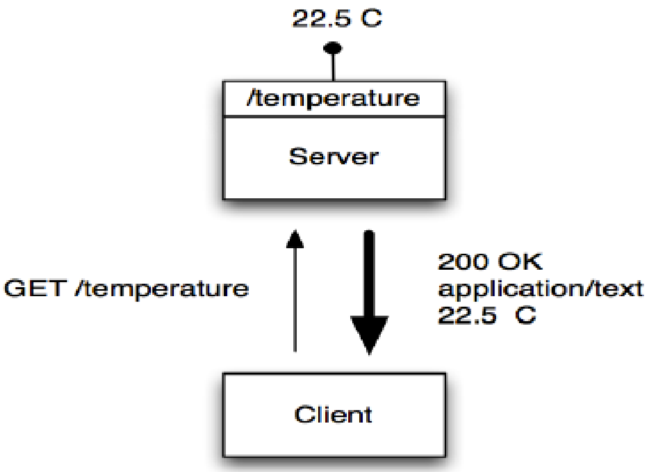
Web Architecture and Web naming





CoAP Constrained Application Protocol

Client/server Architecture



CoAP implementation acts both in client and server role

- Response codes
- 2.xx success code
- 2.01 HTTP 201 “created”
- 2.02 Deleted or HTTP 204 “No content”

You can find others in the RFC definition

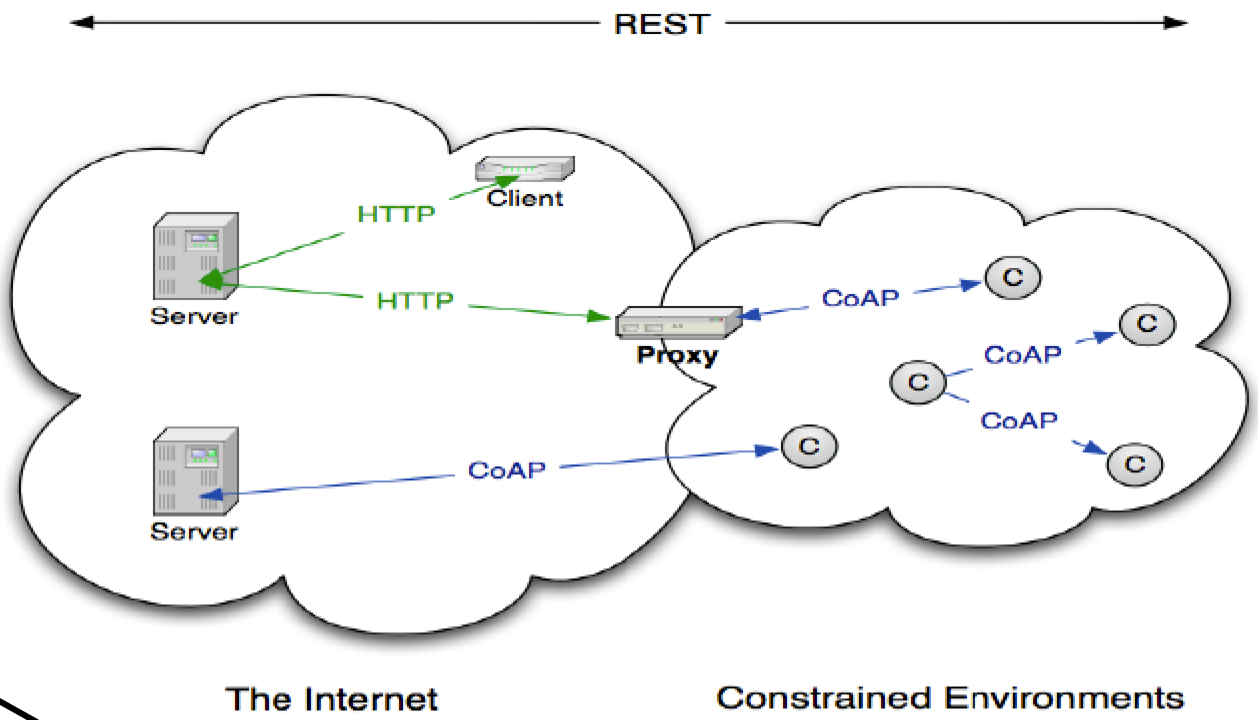
- Method codes
- GET, POST, PUT and DELET

- Asynchronous Exchange

CoAP Constrained Application Protocol

The CoAP Architecture

Web Ressources
CoAP
UDP
IP
Constrained Link



Port

coap://[aaaa::a00:f7ff:b9bc:4643]:5683/test/hello

CoAP Constrained Application Protocol

The CoAP two layer approach

```
+----------------------+
|      Application     |
+----------------------+
+----------------------+  \
|  Requests/Responses  |  |
|----------------------|  | CoAP
|       Messages       |  |
+----------------------+  /
+----------------------+
|          UDP         |
+----------------------+
```

Messages Layer: It deals with UDP and the asynchronous nature of the interactions

Request Response Layer: Method and Response codes

CoAP is a single protocol, with messages and request/response **just features of the CoAP header**

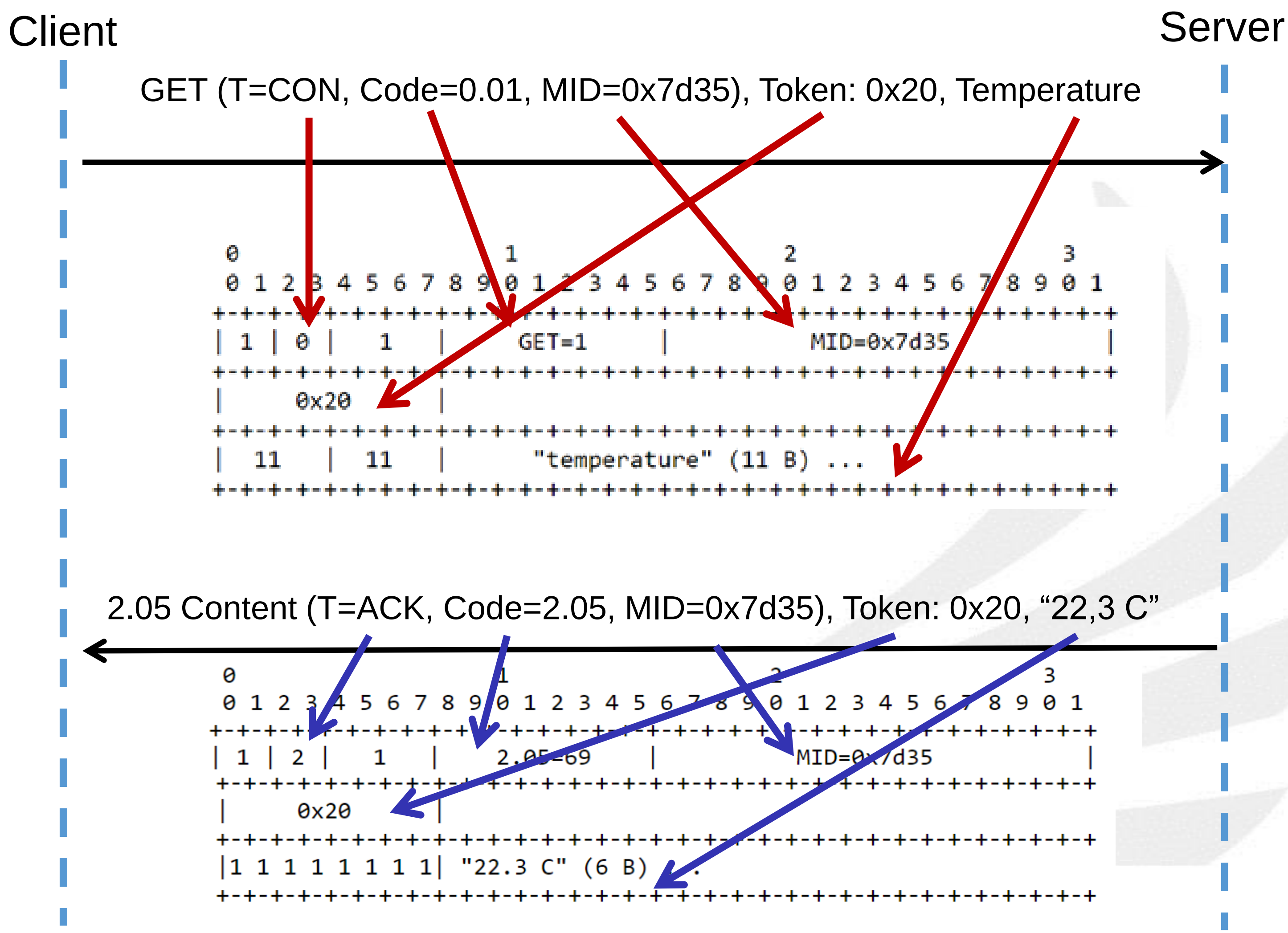
CoAP Constrained Application Protocol

CoAP header (4 bytes)

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

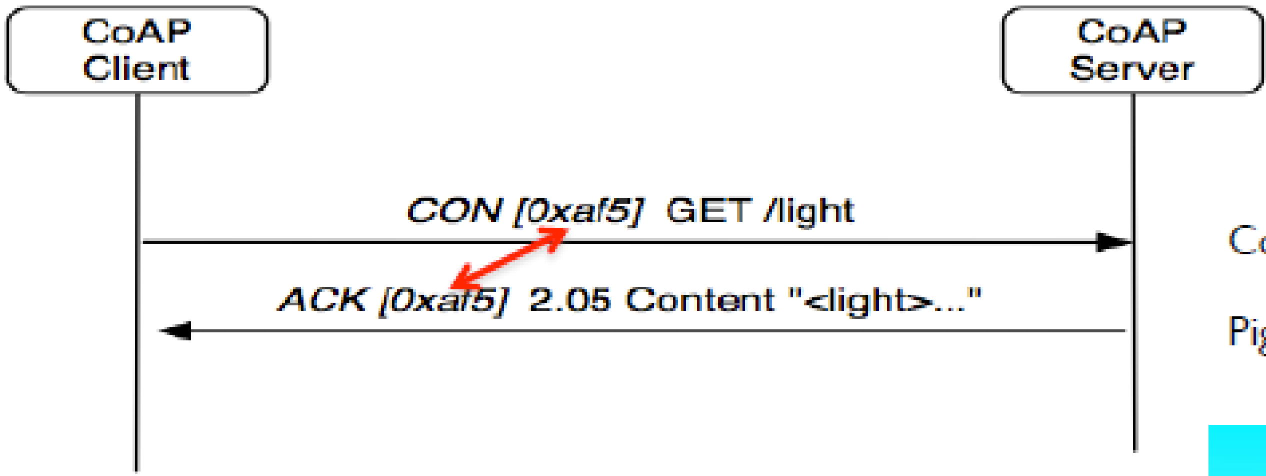
Ver (Version):1;
T (Type): Confirmable, Non-Confirmable, ACK, Reset;
TKL (Token-Length): Length of the token field;
Code: Request Method (1-10) or response code (40-255)
Message ID: 16 bits duplicate detection (NON), matching ack/reset to Requests (CON);
Token: used to correlate requests and responses;

CoAP Constrained Application Protocol



CoAP Constrained Application Protocol

Message Model with request/response

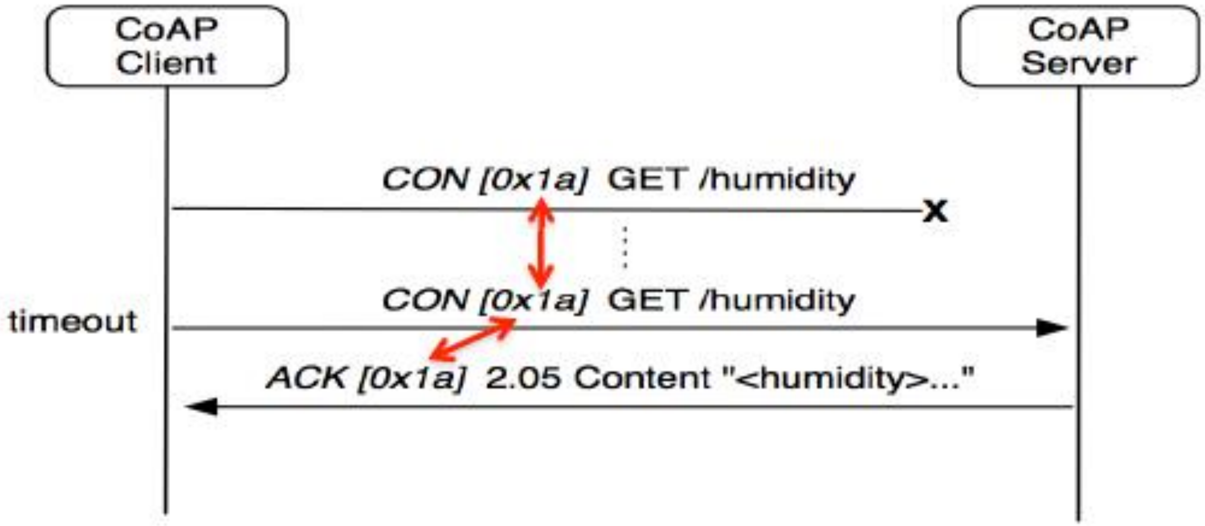


Confirmable Request

Piggy-backed Response

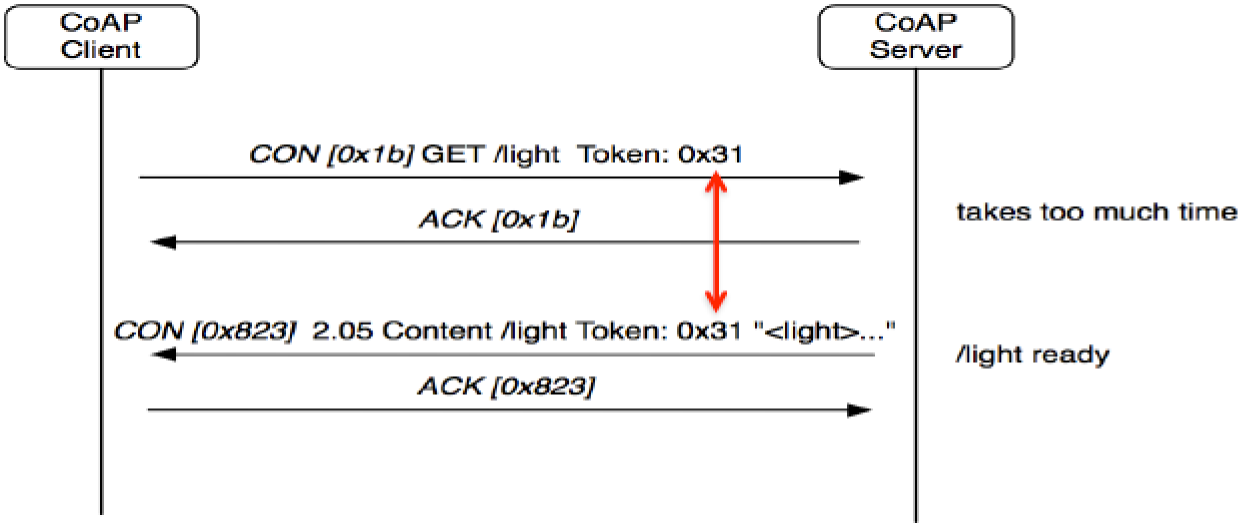
CoAP Constrained Application Protocol

Dealing with packet loss



CoAP Constrained Application Protocol

Separate Response



CoAP Constrained Application Protocol

Interaction model

Confiramble (Con)

Information found

```
Client              Server
   |                  |
   |   CON [0xbc90]   |
   | GET /temperature |
   |   (Token 0x71)   |
   +----------------->|
   |                  |
   |   ACK [0xbc90]   |
   |   2.05 Content   |
   |   (Token 0x71)   |
   |     "22.5 C"     |
   |<-----------------+
   |                  |
```

Confiramble (Con)

Information not found

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

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

Non-Confiramble

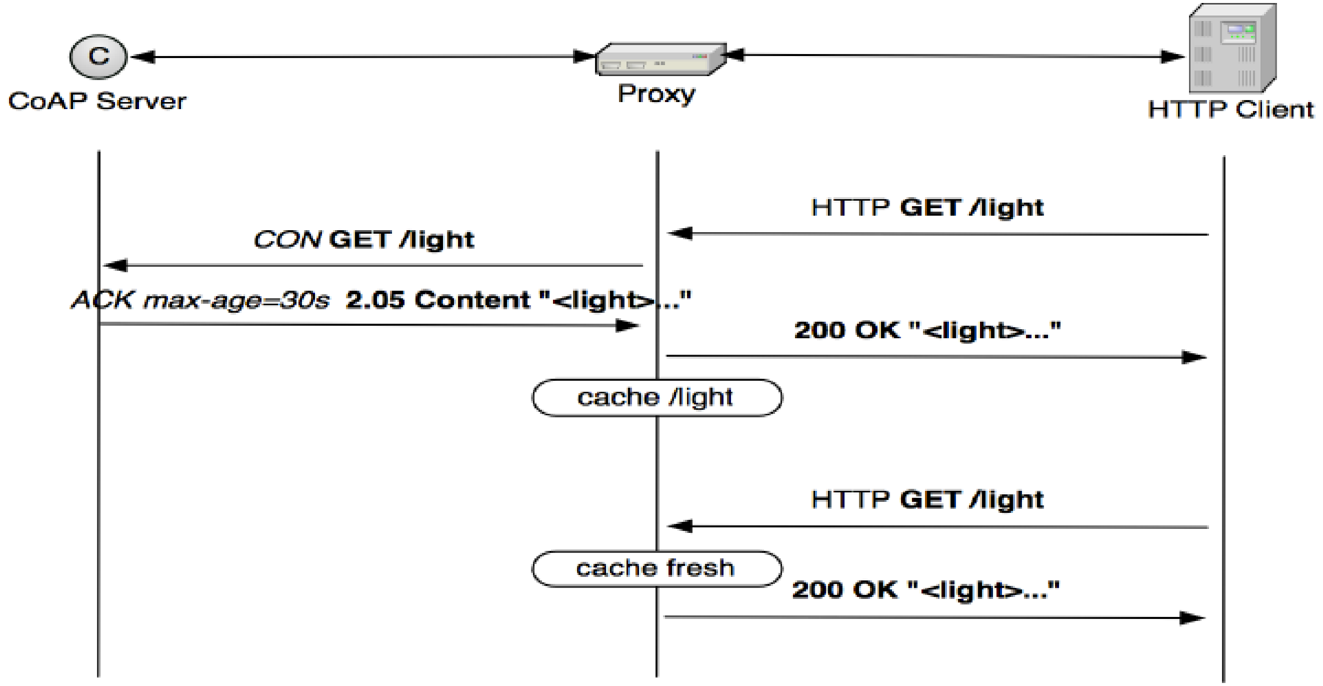
```
Client              Server
   |                  |
   |   NON [0x7a11]   |
   | GET /temperature |
   |   (Token 0x74)   |
   +----------------->|
   |                  |
   |   NON [0x23bc]   |
   |   2.05 Content   |
   |   (Token 0x74)   |
   |                  |
   |<-----------------+
   | 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

CoAP Constrained Application Protocol

Caching and proxiying

CoAP supports **caching of responses** to efficiently fulfill requests. Simple Caches is particularly useful in constrained networks for several reasons:

- Including traffic limiting
- Performance improving,
- Resources accessing times
- Security.



CoAP Constrained Application Protocol

CoAP implementations

- Open source
  - **mbed** includes CoAP support → **ARM**
  - Java CoAP Library **Californium**
  - C CoAP Library **Erbium → Eclipse**
  - **libCoAP** C Library
  - **jCoAP** Java Library
  - **OpenCoAP** C Library
  - **TinyOS** and Contiki include **CoAP** support
- Commercial solution
  - ARM Sensinode NanoService
  - RTX 4100 WiFi Module
- **Firefox** has a CoAP plugin called **Copper**
- **Wireshark** has CoAP dissector support

References

Constrained Application Protocol (CoAP) Tutorial
https://www.youtube.com/watch?v=4bSr5x5gKvA

Home Automation with Node.js and MQTT
https://www.youtube.com/watch?v=80DxfDmoZUI

Using MQTT in Real-World M2M Communication
https://www.youtube.com/watch?v=r6HEQVhgnP8

http://electronicdesign.com/iot/understanding-protocols-behind-internet-things

Contiki 6LoWPAN Quick Guide with nucleo boards (X-NUCLEO-IDS01A4, X-NUCLEO-IDS01A5)

Lauree Magistrali, " Application Layer Solutions for the Internet of Things"

Zach Shelby, "ARM IoT Tutorial"