Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

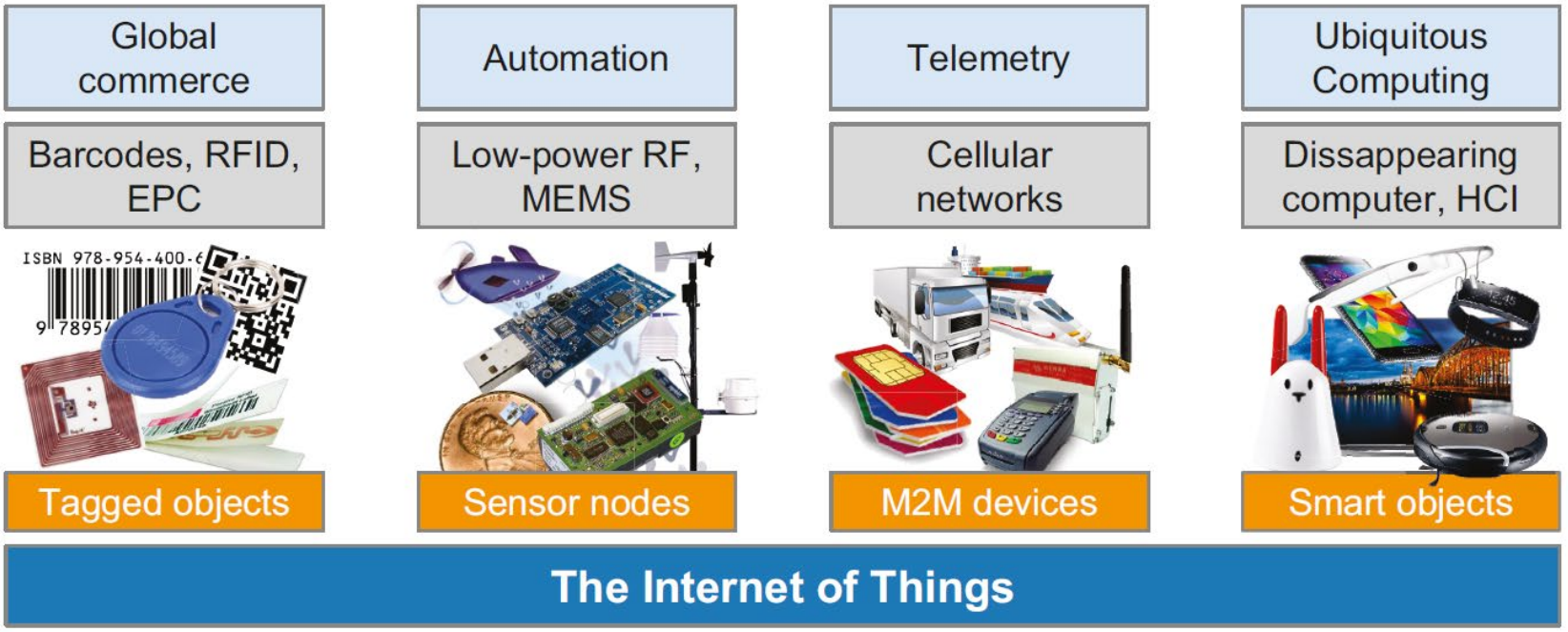
APPLICATIONS INTERNET

PROTOCOLS FOR THE INTERNET OF THINGS – PART I

SERGE AYER - HEIA-FR – TÉLÉCOMMUNICATIONS
CLASSES ISC-2D // 2023-2024

THE INTERNET OF THINGS CONCEPT

Different fields of research and application



Source: Scalable Web Technology for the Internet of Things, F.M. Kovatsch, Ph.D. Thesis

THE INTERNET OF THINGS CONCEPT

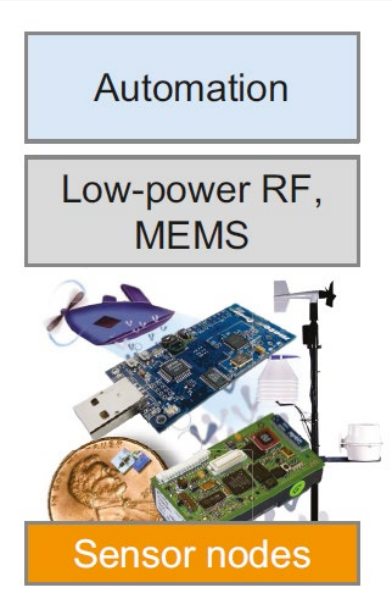
First establishment in 1999, in the field of networked RFID and sensing technology

- Objects tagged electronically
- Physical objects acting as nodes in a networked physical world

THE INTERNET OF THINGS CONCEPT

Wireless sensor networks:

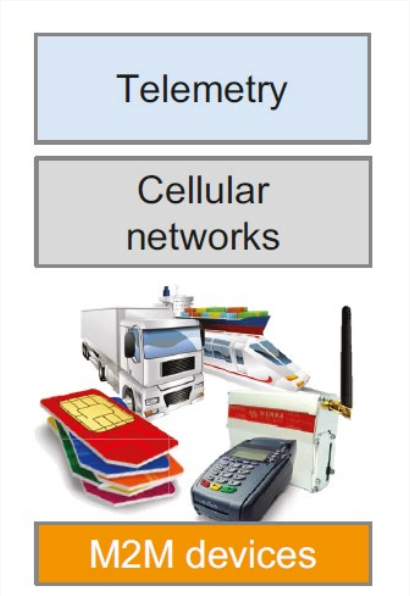
- Low-power radios
- Multi-hop communication (covering large areas with autonomous sensor nodes)
- For real-time sensing of physical phenomena



THE INTERNET OF THINGS CONCEPT

Machine-to-machine (M2M):

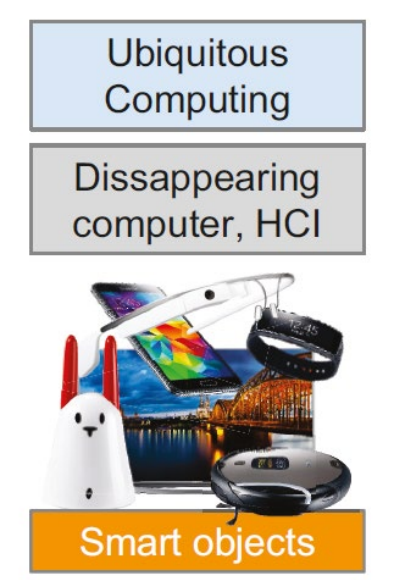
- (Cellular) networks for connecting stationary sensors and mobile objects to a central IT system
- Emerging other long-range radio technologies such as LoRA or Sigfox



THE INTERNET OF THINGS CONCEPT

Smart objects (Ubiquitous computing):

- Everyday objects endowed with processing and communication capabilities, together with sensors and actuators
- Can provide human-computer interaction in everyday lives



THE INTERNET OF THINGS CONCEPT

- Issues with those systems:
  - They all form silo applications, which are closed vertical systems
  - They fulfill a dedicated task
  - They are hard to integrate with systems from other application domains
- One attempt for a cross-application framework for the Internet of Things
  - Adopt the Internet Protocol (IP) for interconnecting physical objects
  - First attempt in 2003 showing that IP support is feasible in WSNs
  - Standardization of 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks (2006-2011)
  - IPv6 over BLE almost standardized (draft-17 in August 2015)

THE INTERNET OF THINGS CONCEPT

- Belief:
  - The IP-based IoT will enable the seamless integration of the physical world into the virtual world
  - It will allow to connect previously isolated systems for the creation of novel applications
  - This is the Web of Things concept (WoT).
- Aims at adapting World Wide Web concepts / patterns to the Internet of Things concept

THE WEB OF THINGS CONCEPT

- The application layer of the WWW is HTTP
  - It allows the creation of "web mashups", or web applications aggregating content (or services) from several sources into a single graphical interface
- Aggregating services from several devices into a single application or "physical mashup" is also the intent of the WoT
- However, HTTP is not appropriate for resource-constrained devices

WHY NOT HTTP?

- Built on Transmission Control Protocol (TCP), that behaves poorly in low-power networks
- Is a text, verbose based protocol
  - Requires large memory buffers on devices
  - Requires a lot of useless bandwidth

WHY NOT HTTP?

- Attempts to develop constrained implementations of HTTP have been made:
  - Breaking the so-called "end-to-end arguments" of the Internet stating that specific application-level functions should not be built into the lower levels of the system
- Also HTTP is missing some important features of the WoT
  - Device and resource discovery
  - Bi-directional communication (e.g. server push)

WHAT IS REQUIRED FOR THE WOT?

- Scaling down
  - Web technologies must be scaled down to operate on constrained environments
  - A new application protocol replacing HTTP needs to be developed
- Scaling up
  - Web technologies must be scaled up to operate on billions of devices
  - It means scalability, cacheability, etc..
- Improved usability
  - For users and developers

THE CONSTRAINED APPLICATION PROTOCOL

- Closing the gap for the implementation of IPv6 on resource-constrained devices as a new standardized Web application protocol.
- Developed by a working group for Constrained RESTful Environments (CoRE)
- Accepted as Proposed Standard by the Internet Engineering Task Force (IETF) in 2013

CLASSIFICATION OF RESOURCE-CONSTRAINED DEVICES

- Based on Terminology for Constrained-Node Networks. RFC 7228
- Constrained node:
  - Node with limits on power, memory and processing resources
  - Hard upper bounds on state, code space and processing cycles
  - Optimization of energy and network bandwidth is required

CLASSIFICATION OF RESOURCE-CONSTRAINED DEVICES

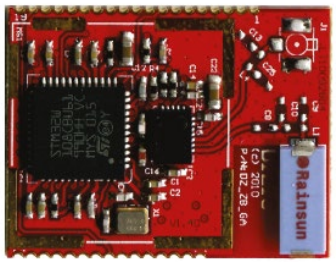
- Constraints are on
  - Flash/ROM (maximum code complexity)
  - RAM (size of state and buffers)
  - Processing power
  - Available power
  - User interface and accessibility

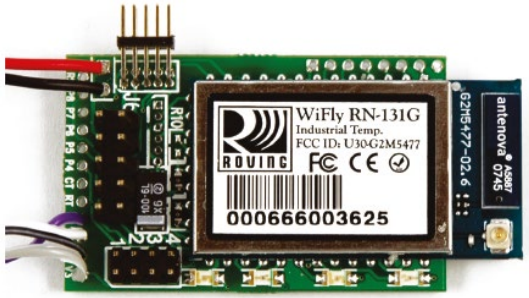
CLASSIFICATION OF RESOURCE-CONSTRAINED DEVICES

- Three classes established on RAM and Flash constraints

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

From left to right:
Tmote sky (class 0)
SMT32W Soc (class 1)
Roving RN-131 (class 2)





CLASS 1 DEVICES

- Cannot run a full IP stack using HTTP over TLS
- Are capable to use a protocol stack designed for constrained nodes, such as CoAP
- Therefore, do not require the use of gateway and can be integrated as fully developed peers into an IP network
- Need to be parsimonious in state memory, code space and power consumption for protocol and application usage
- The nRF51/52 device belongs to this class

COAP

- Requirements
  - RESTful
  - Compact
  - Low parsing complexity
  - Can handle lossy communication links
  - Implements features such as discovery and bidirectional communication
- Answer: CoAP does all this !
- And it is not a mere compression of HTTP

COAP

- Follows the REST architectural style
  - Client-server
  - Resources are addressable by Uniform Resource Identifiers (URIs)
  - Stateless
  - Uniform interfaces with standardized Internet Media Types
  - Caching and proxying are possible for high scalability

COAP REQUEST-RESPONSE SEMANTICS

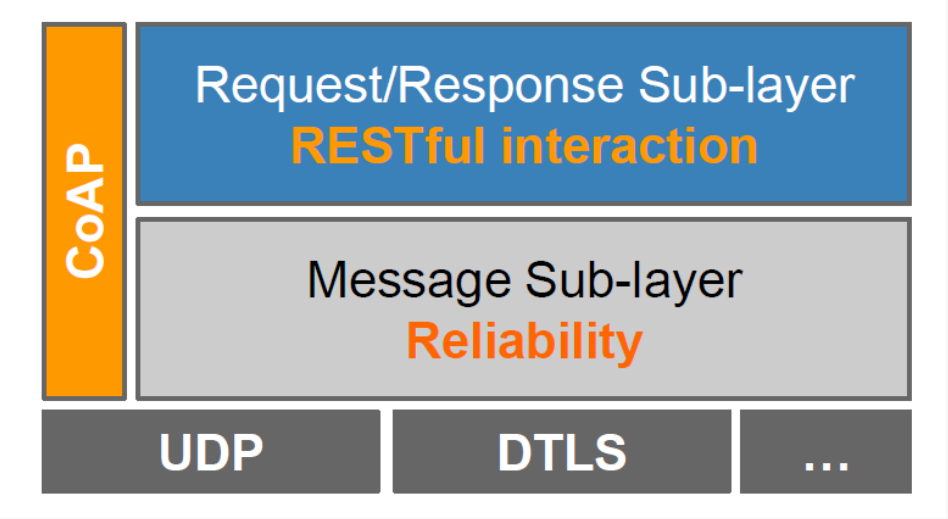
- CoAP operates under a similar request/response model as HTTP:
  - A CoAP endpoint acting as client sends one or more CoAP requests to a server
  - The server services the requests by sending CoAP responses
  - Unlike HTTP, requests and responses are not sent over a previously established connection but are rather exchanged asynchronously over CoAP messages.

COAP

- Uses the User Datagram Protocol (UDP) as transport
  - Better performance as compared to TCP in low-power wireless networks
    - Less overhead
    - No connection setup and tear-down
  - Quality of service is implemented as a thin control layer in CoAP

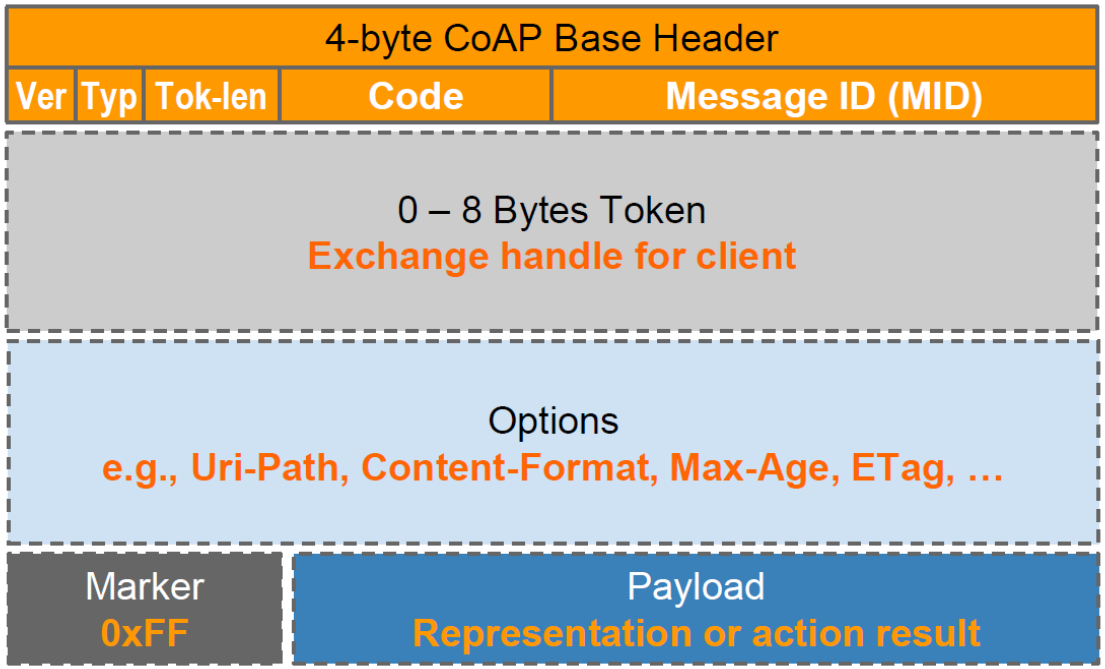
COAP LAYERS

- Layers
  - One sub-layer for handling CoAP messages (request-response model)
  - One sub-layer for handling reliability



COAP MESSAGE FORMAT

- Binary-encoded communication protocol
  - Compact
  - Low parsing complexity
  - 4-byte base header (followed optionally by a token, options, and a payload)

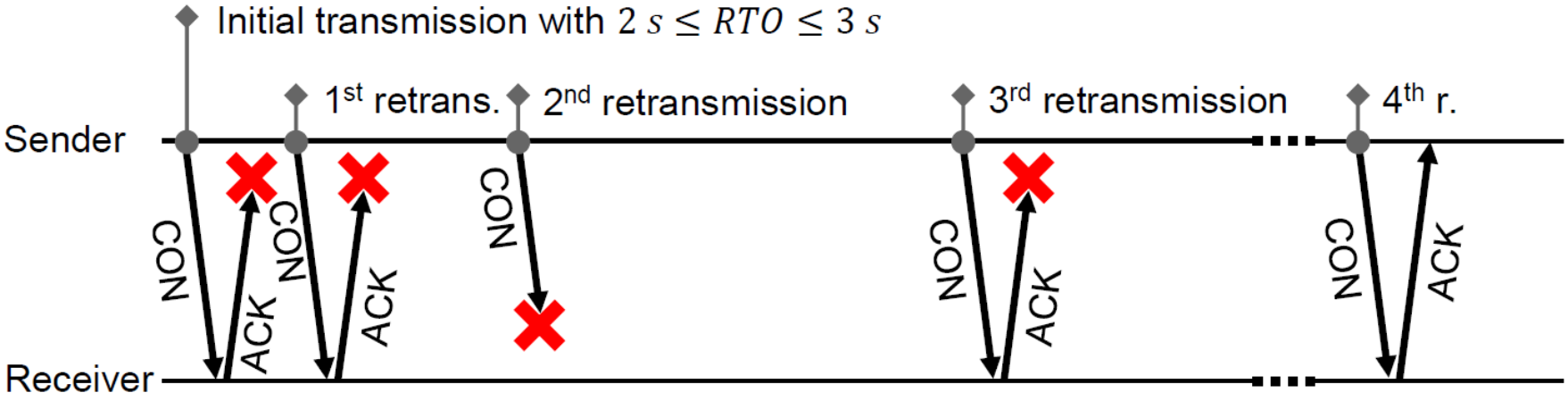


COAP MESSAGE SUB-LAYER

- Provides
  - Duplicate detection
  - Reliable transmission by optional message retransmission
- Message types:
  - CON (Confirmable): the message is retransmitted until the receiver confirms its reception or until time out is reached.
  - ACK: replies to CON messages
    - Allows piggy-back by including the payload corresponding to the CON request.
    - Uses the same MID as the received message.
  - NON (Non-confirmable): best-effort delivery
    - Reply messages to NON messages can be NON or CON
  - RST (Reset): reply messages to CON and NON messages to indicate message processing errors

COAP MESSAGING SUB-LAYER

- Illustration of CON retransmission and acknowledgment



MESSAGE DEDUPLICATION

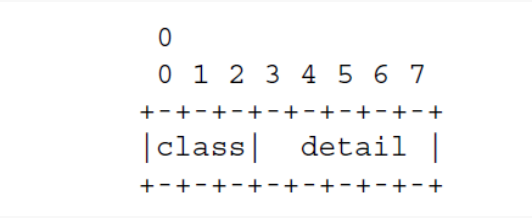
- Duplicate messages are caused when ACK messages are lost and a COAP endpoint keeps retransmitting CON messages.
- Also, UDP as a transport layer can create message duplication by the network itself.
- Solution:
  - Filtering based on the 16-bit MID
  - The MID of every active CON and NON message must be unique within its source endpoint (IP address + UDP port number).
  - The lifetime of a message must be known so that the node understands which messages are active or not.

COAP REQUEST-RESPONSE SUB-LAYER

- Implements a REST architecture
- Common grounds with HTTP 1.1 (not with HTTP 2.0)
- Code field:
  - Is a method code for a request
    - Four methods: GET, PUT, POST, and DELETE
    - Same semantics as in HTTP
    - Allows a stateless, transport mapping from/to HTTP
  - Is a status code for a response
    - Defined with references to HTTP 1.1
    - Some codes are more meaningful for caching and M2M interactions

COAP REQUEST-RESPONSE SUB-LAYER

- Codes are represented internally with 8-bit numbers
  - For requests: CODE = method
  - For responses: CODE = CLASS*32 + DETAIL
    - 3 classes:
      - Success: 2
      - Client error: 4
      - Server error: 5



COAP REQUEST-RESPONSE SUB-LAYER

- Safe (only retrieval), idempotent (can be invoked multiple times with the same effects)

Code	Description	Comment	HTTP
0.xx	**Methods**		
1	GET	safe, idempotent	GET
2	POST	—	POST
3	PUT	idempotent	PUT
4	DELETE	idempotent	DELETE
2.xx	**Success**		
2.01	Created	in response to POST or PUT	201
2.02	Deleted	in response to DELETE or POST	204/200
2.03	Valid	in response to GET with ETag	304
2.04	Changed	in response to POST or PUT	204/200
2.05	Content	in response to GET	200

COAP REQUEST-RESPONSE SUB-LAYER

4.xx	Client Error		
4.00	Bad Request		
4.01	Unauthorized	no WWW-Authenticate header (thus no HTTP 401 Unauthorized mapping)	400
4.02	Bad Option	for unrecognized or malformed options	400
4.03	Forbidden	for general denial independent from authentication	403
4.04	Not Found		
4.05	Method Not Allowed	no Allow header in CoAP	400
4.06	Not Acceptable		406
4.12	Precondition Failed	based on preconditions defined through options	412
4.13	Request Entity Too Large	maximum size can be included in a response option	413
4.15	Unsupported Content-Format	unsupported request payload	415

COAP REQUEST-RESPONSE SUB-LAYER

5.xx	Server Error	
5.00	Internal Server Error	500
5.01	Not Implemented	501
5.02	Bad Gateway	502
5.03	Service Unavailable	503
5.04	Gateway Timeout	504
5.05	Proxying Not Supported	502

COAP RESPONSES

- Can be
  - Piggybacked
  - Separated
- Piggybacked responses:
  - Carried directly in the ACK message that acknowledges the request – meaning that the request was a CON request.
  - Can indicate success or failure.
  - The server decides for piggybacking or not.

COAP RESPONSES

- Separate responses:
  - For all NON requests.
  - When processing for obtaining the response content requires more time than admissible (see Exercise 2.2).
  - When the request was a CON message, the ACK message is then an empty message.
  - Can be sent as a CON or NON message. For CON messages, the client must then send an empty ACK message.

COAP REQUEST-RESPONSE MATCHING

- Is done by means of the token and not by means of the MID (used only by the messaging sub-layer)
- The source endpoint of the response must be the same as the destination endpoint of the request.
- In a piggybacked response, the MID and the tokens of the CON request and of the ACK response must match.
- In a separate response, the tokens of the request and of the response must match.

COAP TOKEN

- Used to match a response with a request
- Every message has a token, may be of zero length
- Every request carries a client-generated token that the server must echo without modification in any resulting response.
- Tokens are or should be generated in such a way that it is unique for active tokens for a given source/destination.

COAP TOKEN

- Empty tokens are or may be used when
  - No other tokens are in use to a destination
  - When requests are made serially
- When not using TLS, tokens should be nontrivial and randomized
  - For protecting against spoofing of responses