

# Internet des Objets (IoT): 6lowpan

***Abderrezak RACHEDI***

Professeur des Universités (Full Professor)

University Gustave Eiffel (UGE)

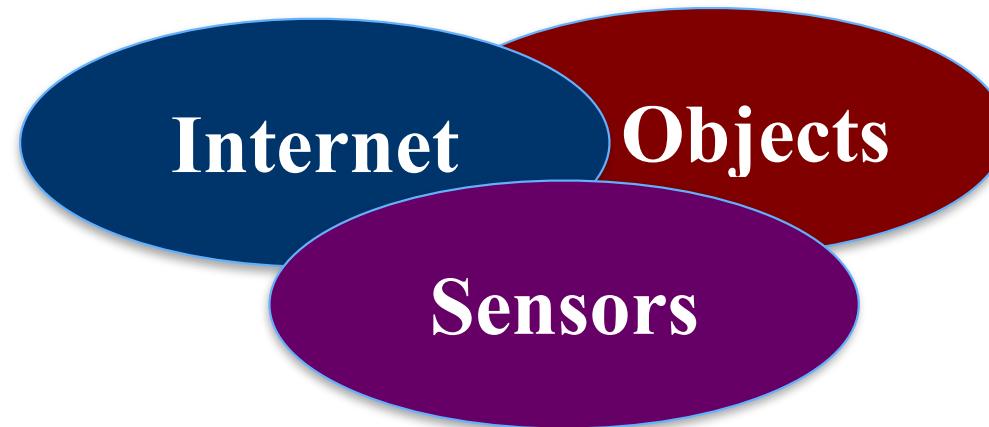
Gaspard Monge computer science (LIGM UMR8049)

**Année 2023/2024**

# Plan

- Introduction
  - Internet of Things (IoT)
  - IoT ecosystem
  - IoT applications
  - Architecture of the communicating object
  - Wireless Sensor Networks (WSN)
  - IoT vs. WSN
  - Standards and technologies associated with the IoT
- 6LoWPAN
  - The interest of 6LoWPAN technology
  - Important elements of IP over 802.15.4
  - Comparison between 6LoWPAN, WiFi Tag and Zigbee
  - Characteristics of 6LowPAN
  - Network architecture of 6LowPAN
  - The 6LoWPAN frame format
- IP header compression and optimization techniques
  - Addressing with 6LoWPAN
  - Neighbor Discovery Protocol
  - Routing - RPL
  - Mobility
  - Security
  - Formats and Application Protocols
  - Operating Systems
    - Contiki-OS
    - COOJA simulator for Contiki

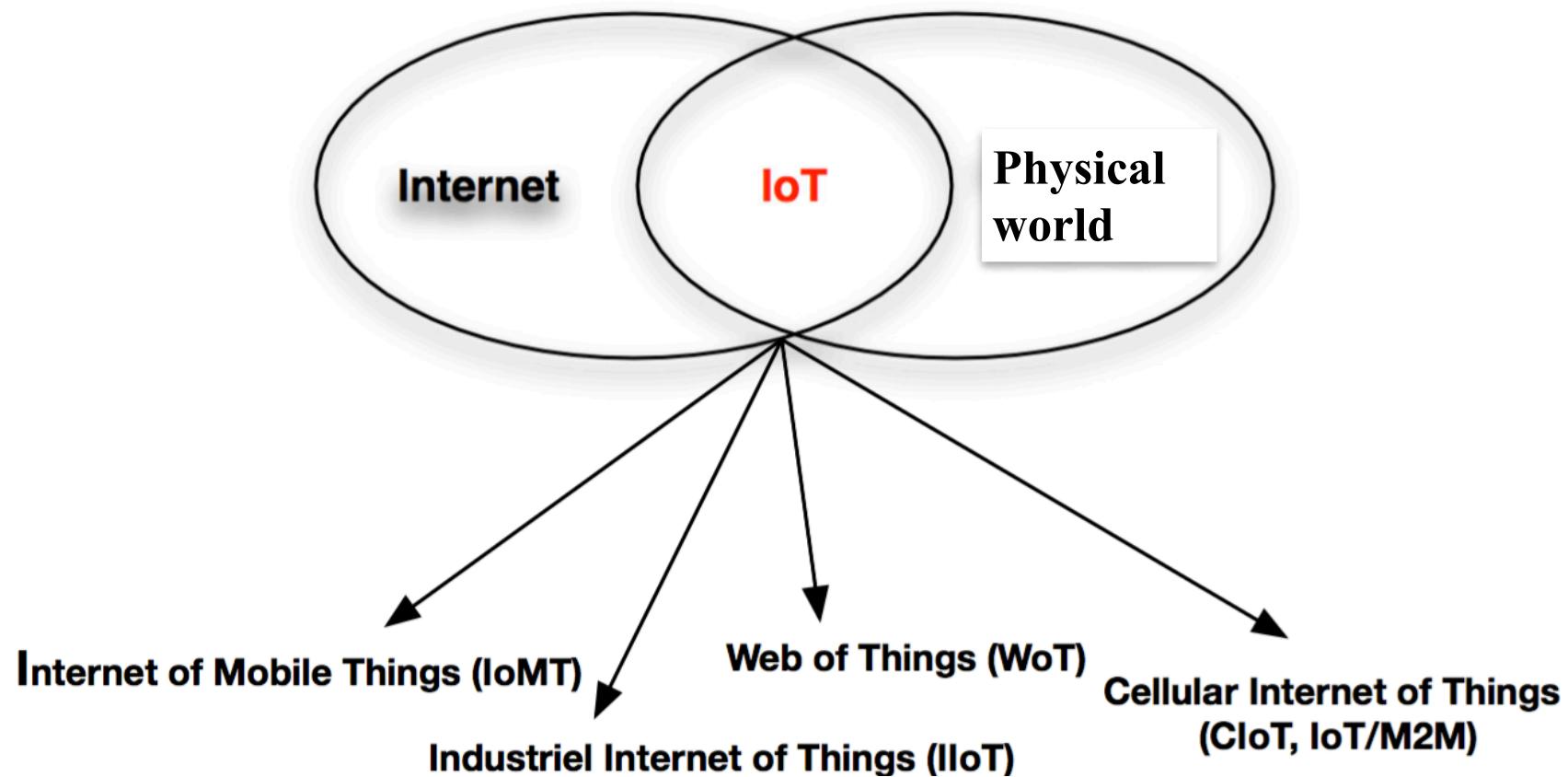
# Definition of Internet of Things (IoT)



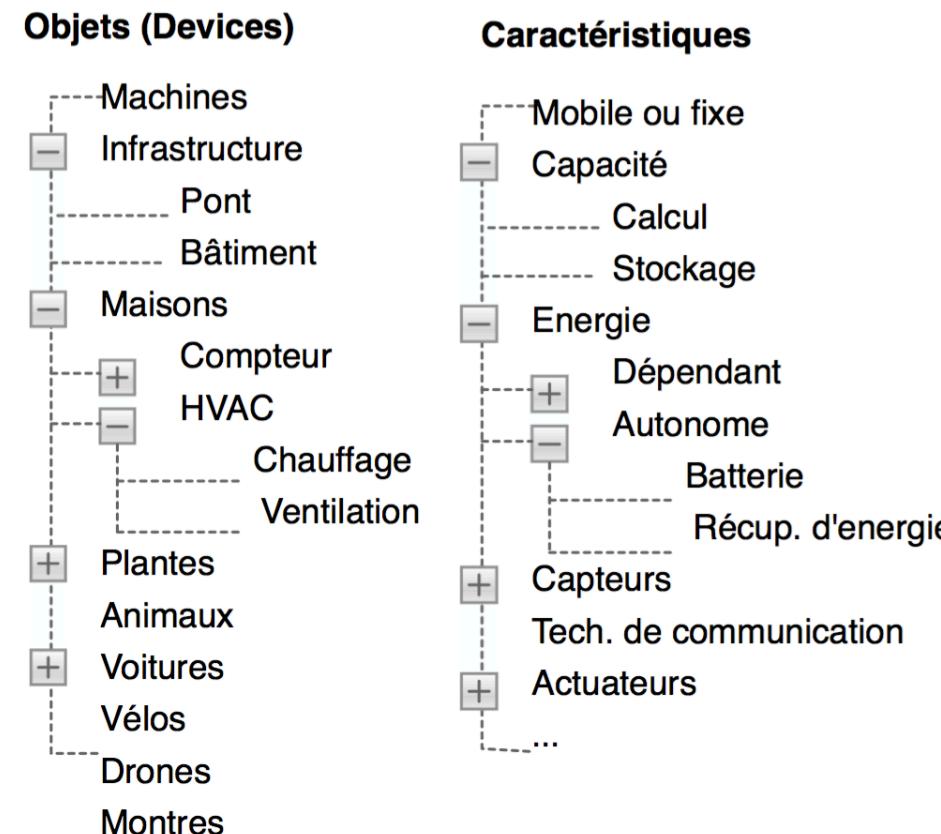
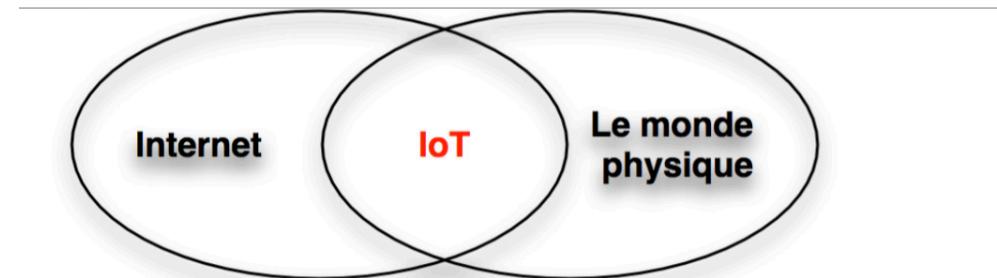
- Internet: Interconnection of IP networks
- Communicating objects : Electronic components equipped with a communication module
- Sensors: allows to capture and transform a physical quantity into another manipulable physical quantity

IoT is part of the Internet of the Future.  
It is defined as the interconnection of objects via the Internet to exchange information for a predefined purpose.

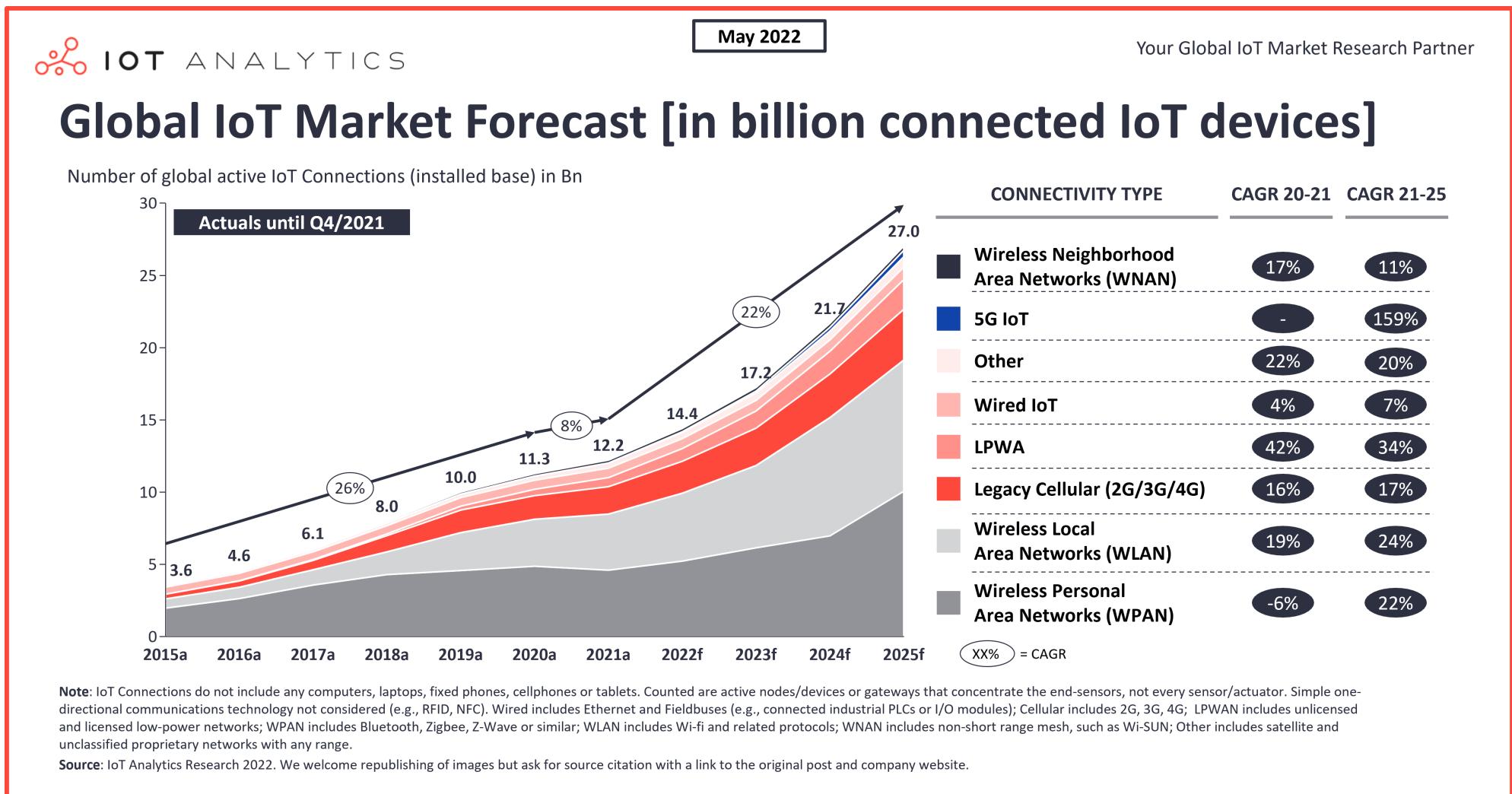
# Definition of Internet of Things (IoT)



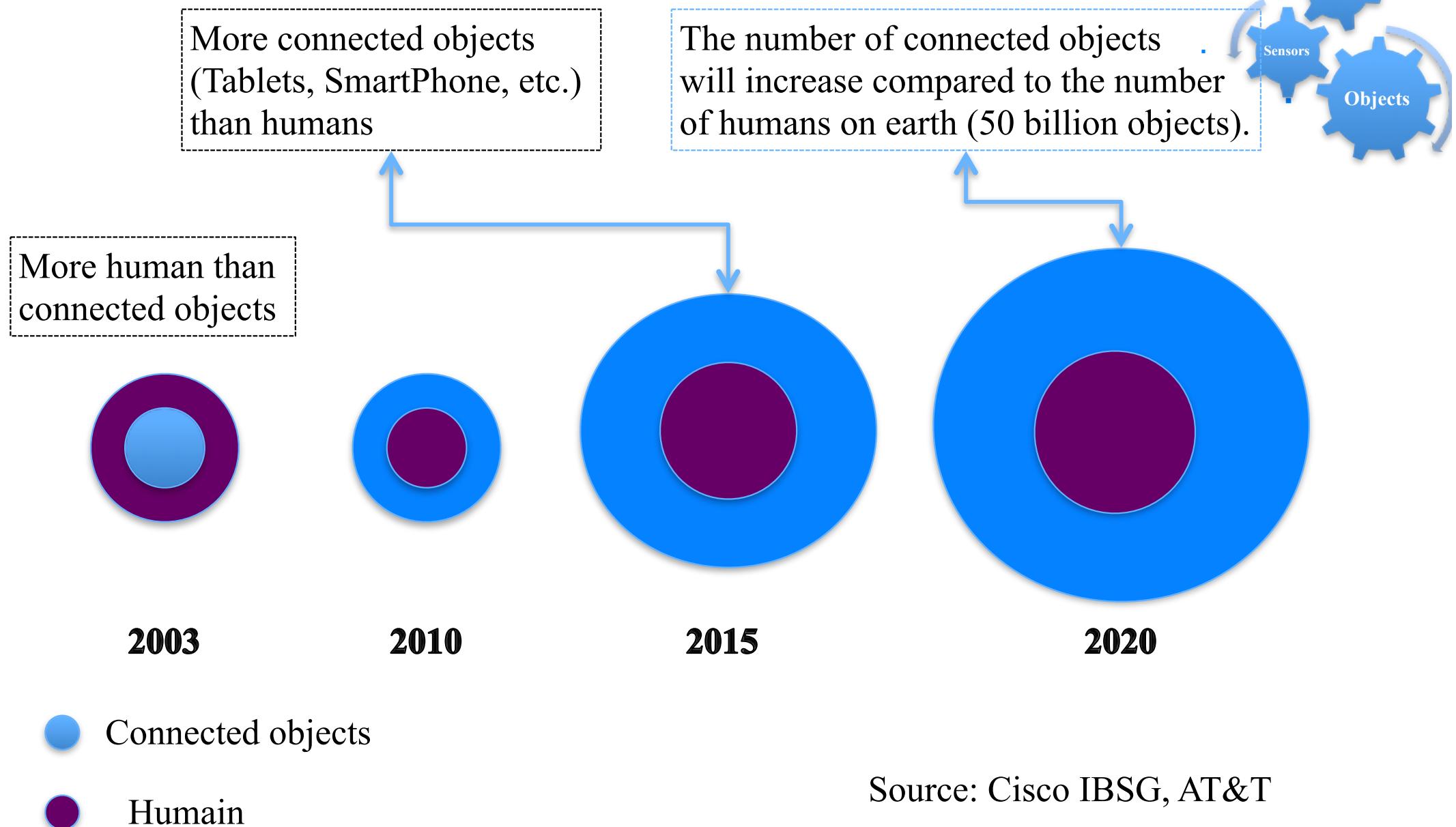
# Definition of Internet of Things (IoT)



# IoT market Forecast (from IoT Analytics)

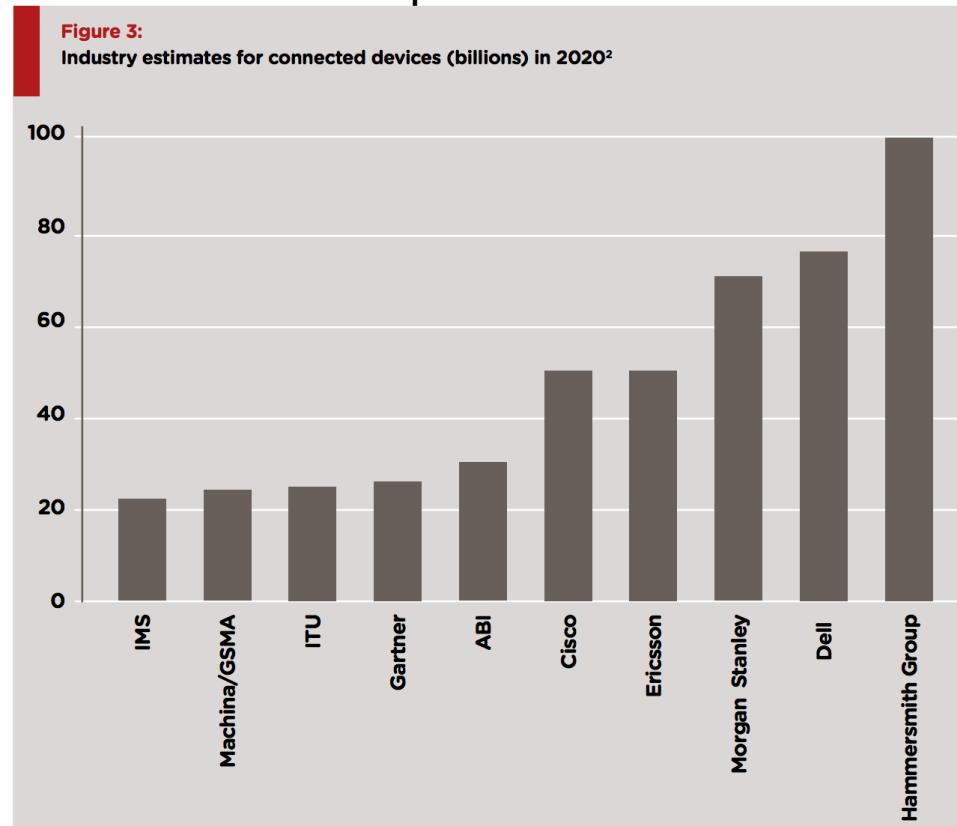


# Evolution of connected objects/machines



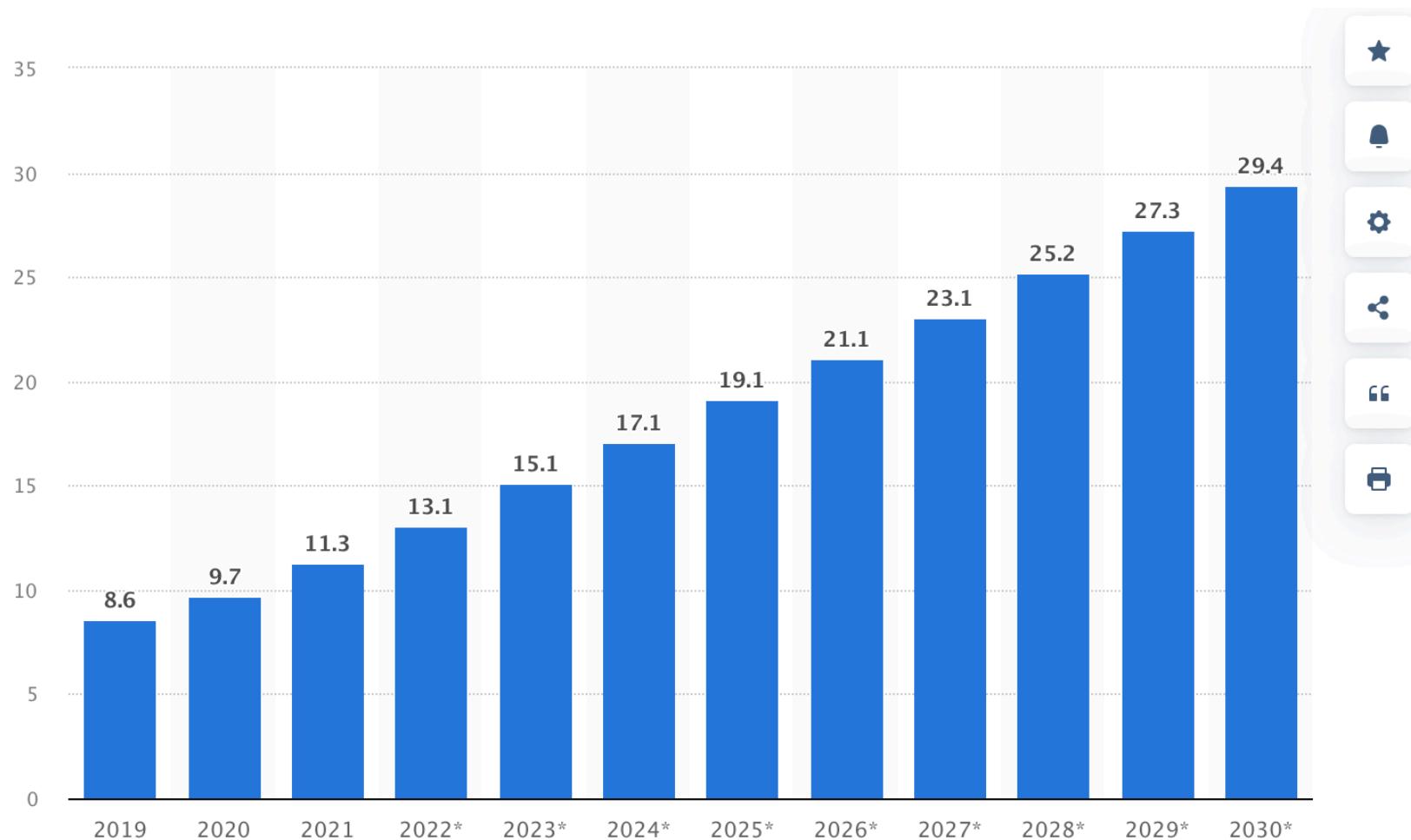
# The Internet of Things in Figures

- **400 million euros**, that's the world's turnover in 2015
- **50 billion**, the number of connected products in circulation in 2020



- **35%** is the percentage of data (all data combined) produced by the Internet of Things in 2020\*
- **44 billion Gigabytes**, the global volume of data in 2020

# Number of IoT connected devices from 2019 to 2021, with forecasts from 2022 to 2030



# The IoT Ecosystem

**Service-oriented**  
(Added value)

**Data-oriented**  
(Big data)

**Network-oriented**  
(cloud & Network)

**Devices-oriented**  
(physical objects)

## **Decision-making process**

(based on the analysis of data collected by sensors)

## **Applications/Services**

(customized and object-data oriented )

## **Data analytics/data engineering**

(Data analysis, data fusion/aggregation, information extraction)

## **Data storage / Database**

(Cloud storage, SQL/NoSQL)

## **Data transportation and connectivity**

(Protocols engineering, networking, communication technologies)

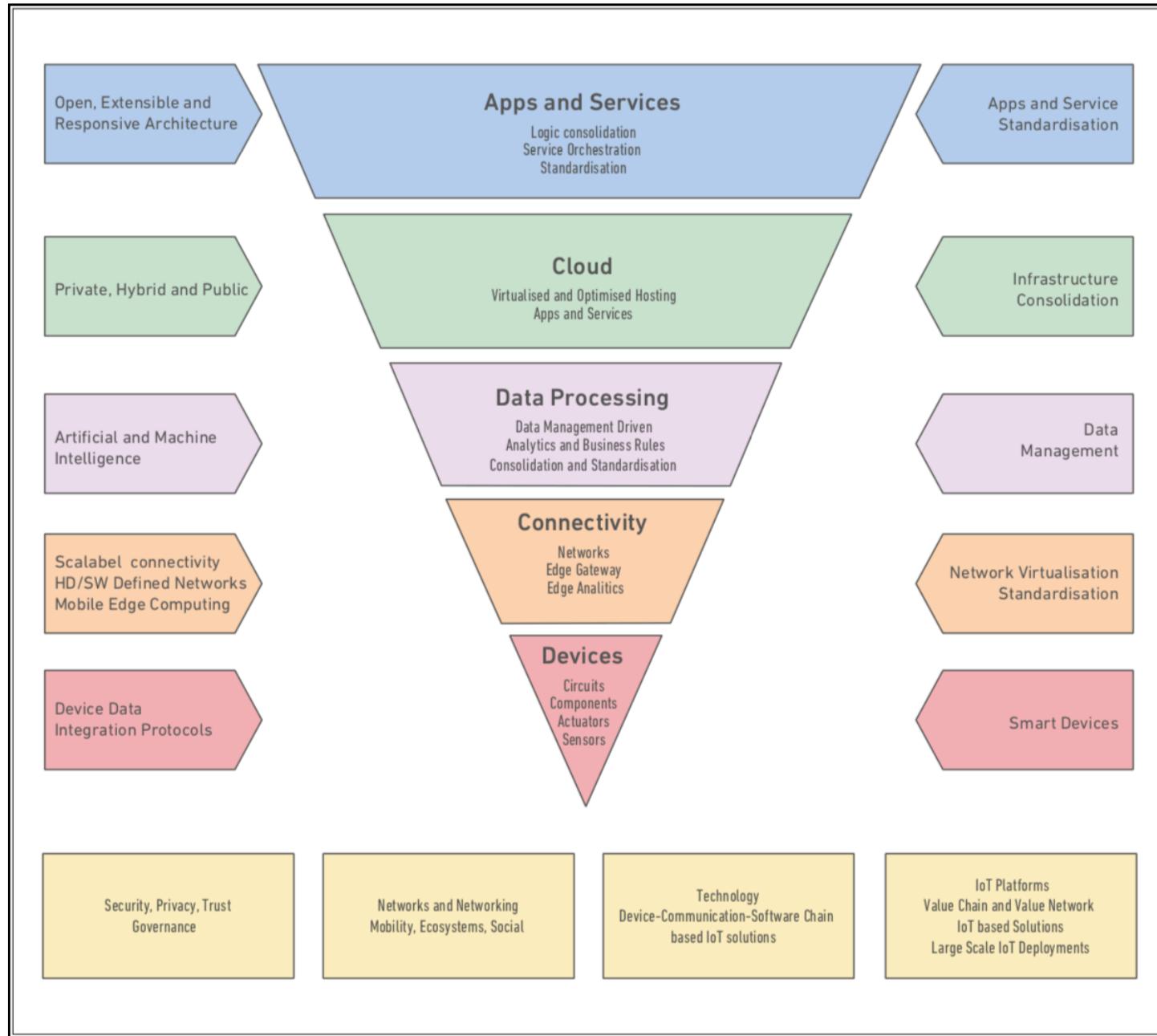
## **Global infrastructure (Systems/Network)**

(Network architecture, cloud computing, virtualisation, etc )

## **Objects (devices)**

(Sensors, actuators, vehicles, drones, SM, etc )

# IoT Platforms covering the data value chain



# IoT applications

## Typical Views of the Internet of Things



Building Automation



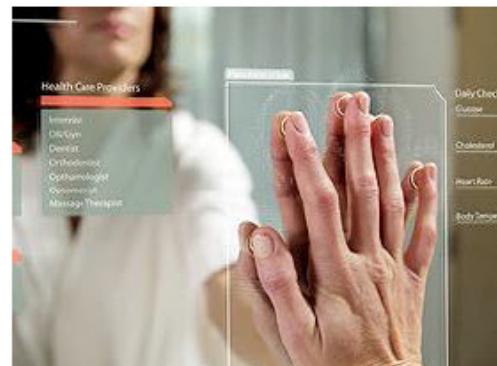
Smart City



Smart Lighting



Smart Grid



Smart Health

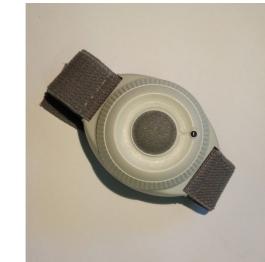


Industrial Automation

# IoT applications

- *Health & Well-being*

- The connected blood pressure monitors
- Connected watches
- Activity trackers
- Fall detectors
- Body Area Networks (BAN)
- ....



# IoT applications

- The Smart Home

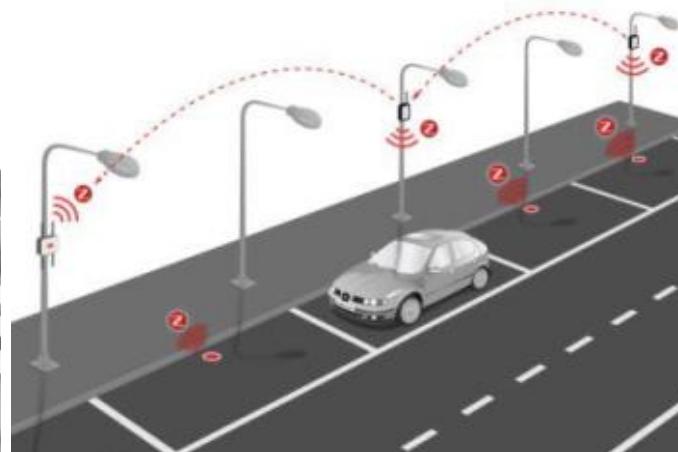


# IoT applications

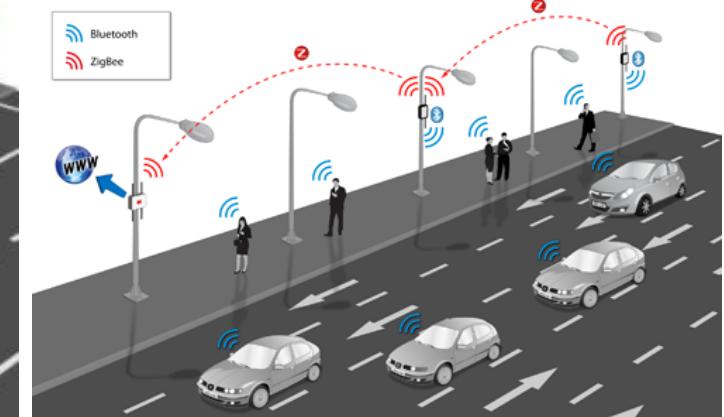
- Smart cities



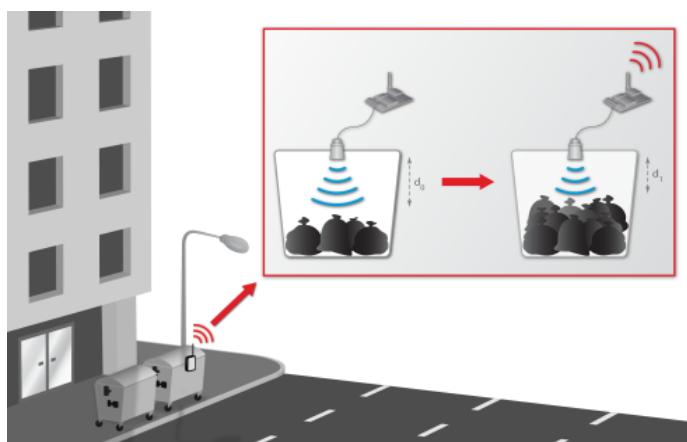
*Smart pollution management*



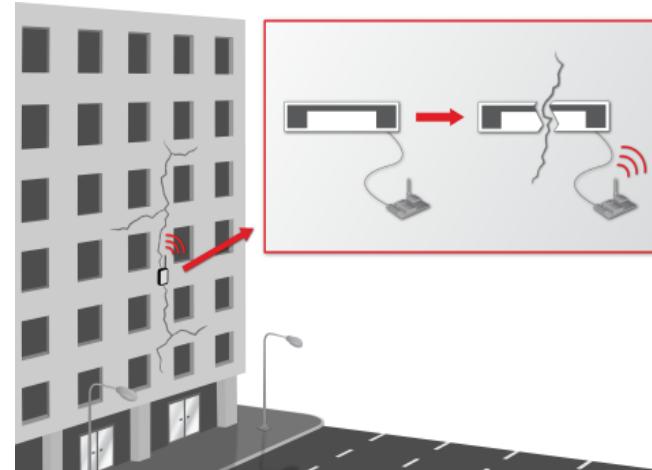
*Smart Parking*



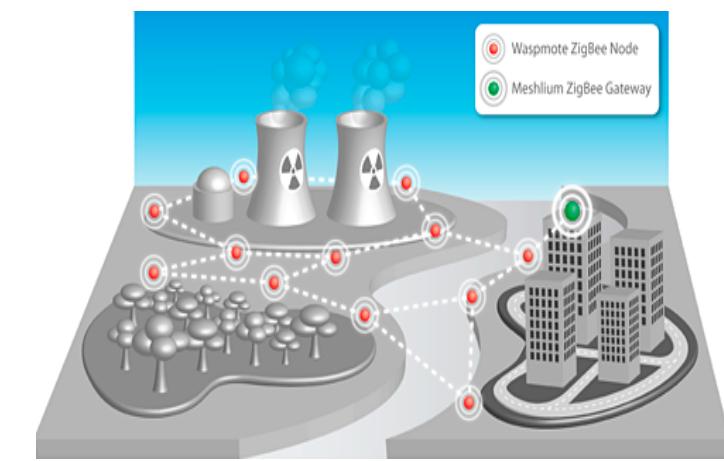
*Cars traffic monitoring*



*Intelligent garbage management*



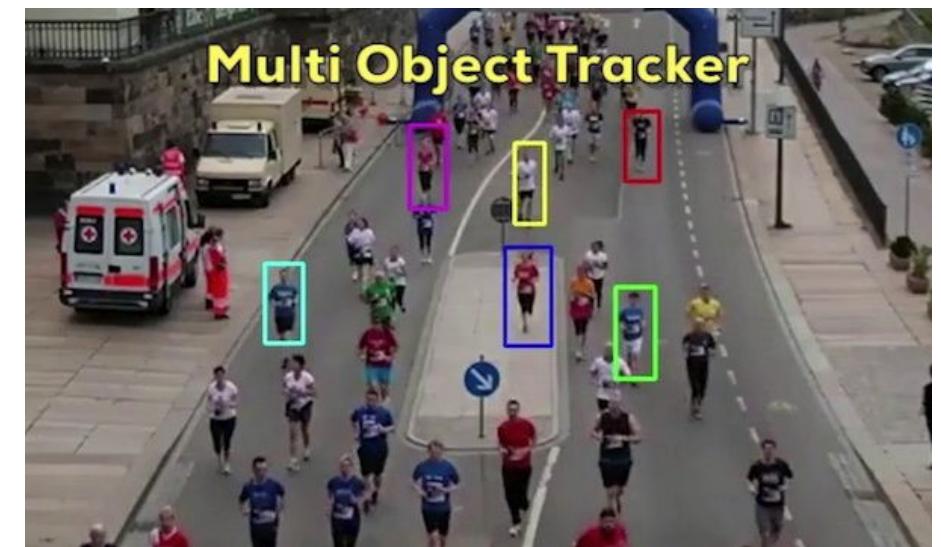
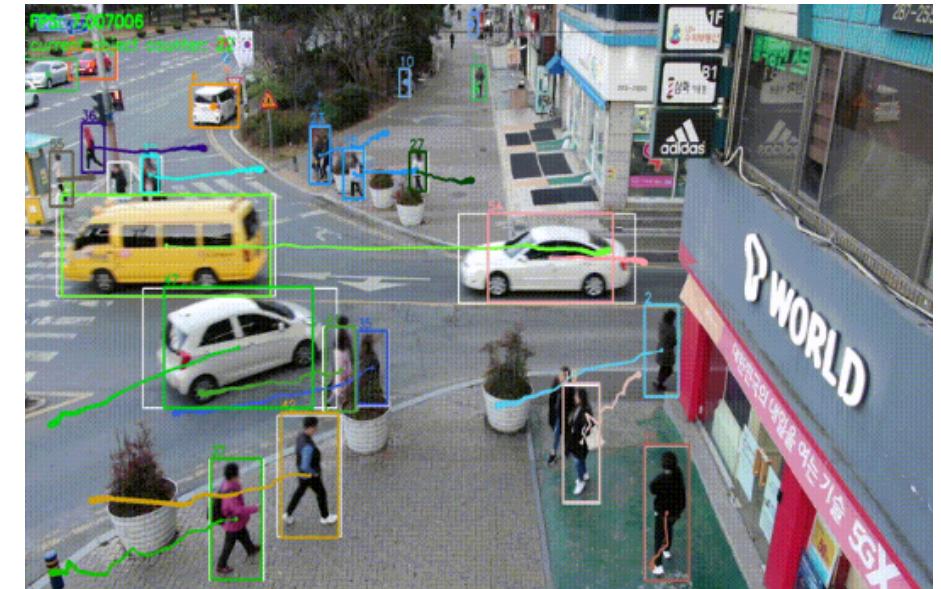
*Building condition monitoring*



*Monitoring of sensitive installations*

# IoT applications

- Security:
  - Video surveillance
  - Objects localisation and objects tracking
  - Access control and identification
  - ....



# IoT applications



- Environmental :
  - Ecology and forest fire monitoring
  - Air and water pollution
  - Smart farme & Agriculture

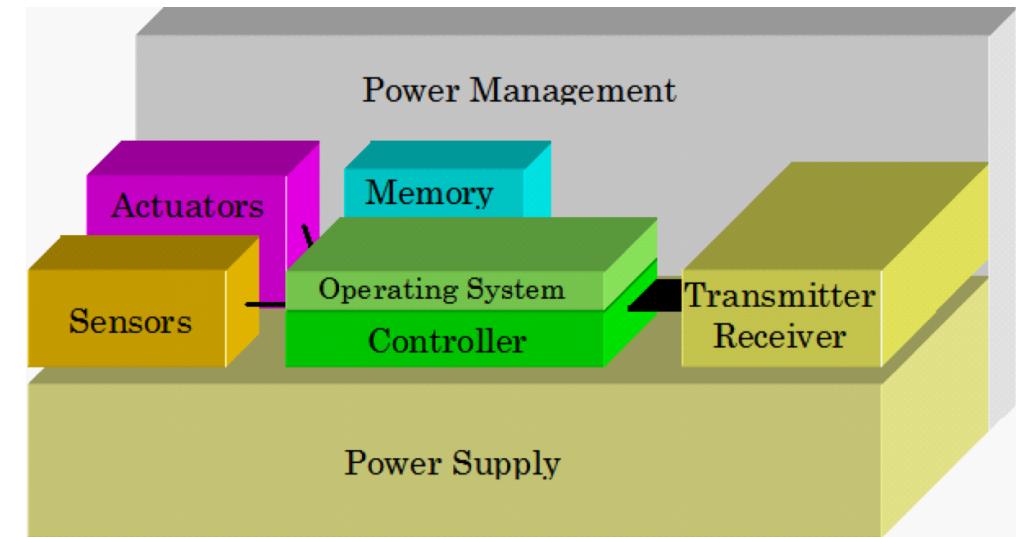


# Architecture of the communicating object

# Architecture of the communicating object

- Each communicating object :

- Processing module
  - Microcontroller and operating system
- Energy management module
  - Battery" power supply
- Memory module (RAM, FLASH)
- Sensing board module
- Actuator module
- Communication module



# Architecture of the communicating object

## ■ Microcontroller

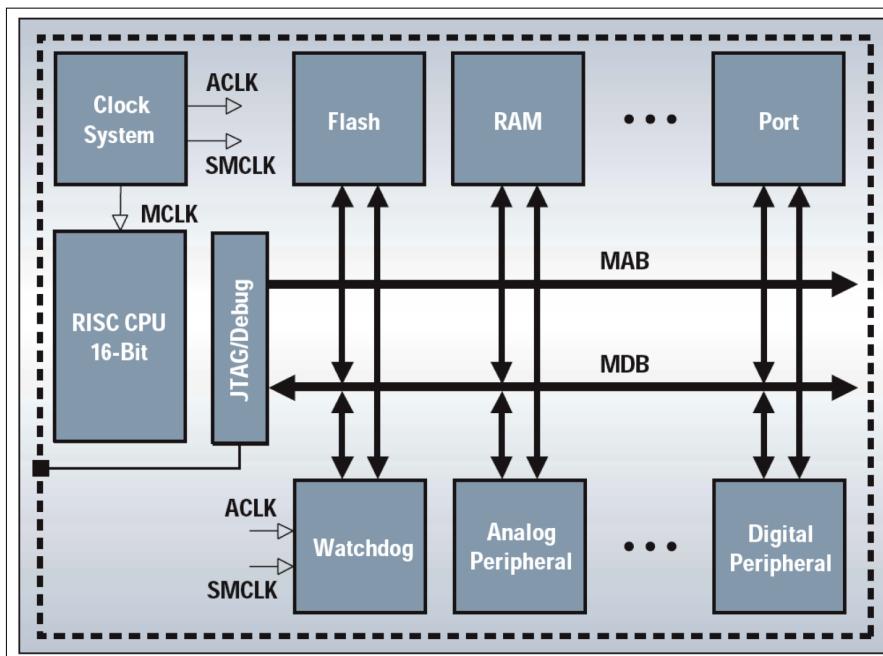
- The central processing unit of the embedded object
- High integration
  - Integrates : RAM, ROM, I/O devices
  - Good trade-off between power and performance rate
  - Cheap, about 0.25 - 10.00 USD
- Runs programs related to on-board system control, calculation and communication
  - Supports time constraints
  - Need for real-time performance
    - Preventive task management
    - Management of queues and semaphores



# Examples:

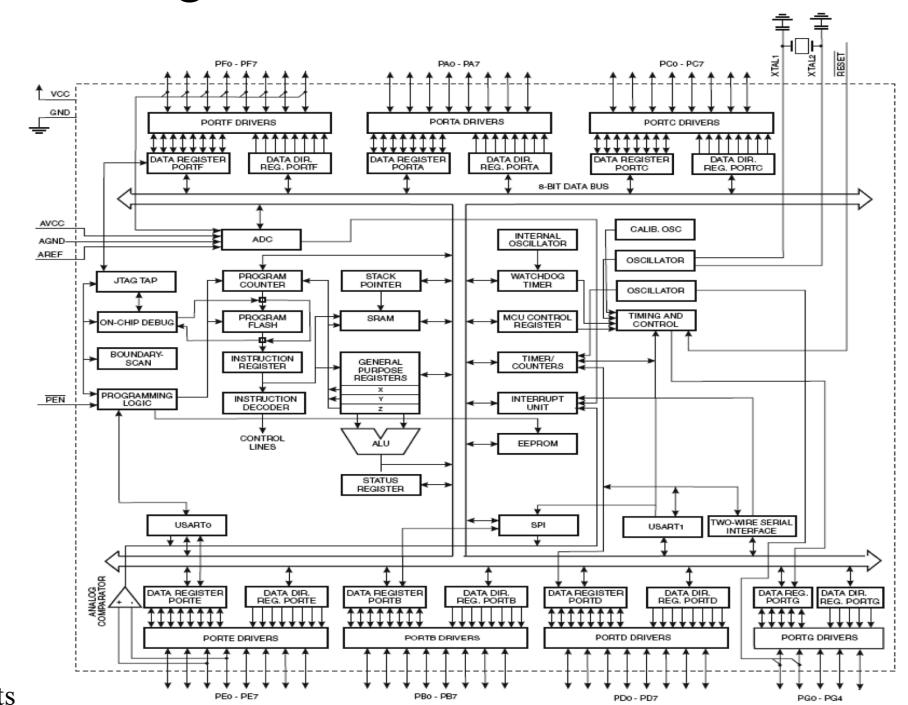
## ■ MSP430

- Texas Instruments mixed-signal uC
- 16-bit RISC - ROM: 1-60 kO
- RAM: jusqu'à 10 kO
- Analogue : 12 bit ADC & DAC - LCD driver
- Digital : USART x 2 - DMA controller Timers



## ■ Atmel AVR

- Famille d'Atmel AVR
- 8-bit RISC
- RAM: up to 4 kO
- ROM: up to 128 kO
- Analogue : ADC , PWM
- Digital : USARTs , Timers



# Architecture of the communicating object

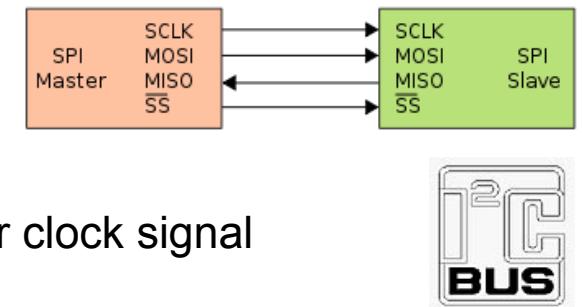
## Memory

- Random Access Memory (RAM)
  - Integrated in the microcontroller
  - Often this is the most important
- Read-Only Memory (ROM)
  - In general, it is implemented with a NOR flash memory
- Flash
  - Programmable and erasable memory
  - Ability to read/write in blocks
  - Slow during the writing process
  - Expensive energy consumption
- External Memory
  - Supported by some microcontrollers

# Architecture of the communicating object

## Interface bus

- Digital and analog I/O
  - Access by port number and pin code
  - Some pins are also connected to the switches
- UART (*Universal Asynchronous Receiver Transmitter*)
  - Asynchronous serial bus
  - After one level of translation it is a RS232 bus
  - Often kbps to mbps data rate
- SPI (serial peripheral interface)
  - Synchronous serial bus
  - Available with Mbps data rates
- I<sup>2</sup>C (inter-integrated circuit) bus
  - 2-wire synchronous serial bus: one for data signal and one for clock signal
- Bus parallèle



# Architecture of the communicating object

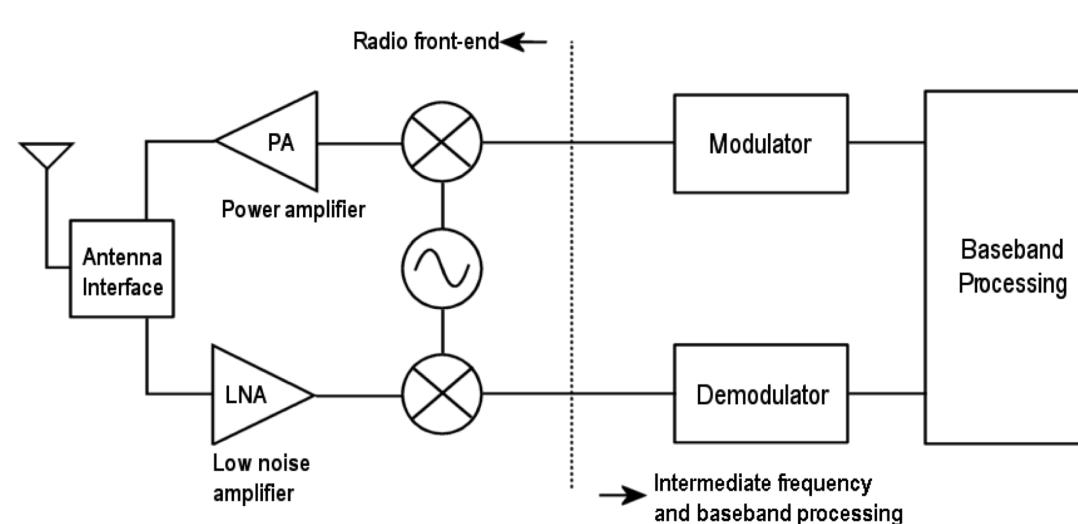
## Communication Module

- Communication interfaces are important in the world of embedded systems.
- Wired interfaces
  - Series: RS232, RS485
  - LAN: Ethernet
  - Industriel: Modbus, Profibus, Lontalk, CAN
- Wireless Interfaces
  - **Low-power: IEEE 802.15.4 (ZigBee, 6LoWPAN, Wireless HART)**
  - WLAN: WiFi, LoRA
  - WAN: GPRS, WiMax, LoRAWAN

# Architecture of the communicating object

- Radio transceiver

- The on-board communication is based on the transceiver:  
It ensures bidirectional communication, in half- or full-duplex through the wireless space.
- Transmit and receive functions are similar to separate functions, but share common resources:  
antenna, power supply, interfaces, frequency generators, etc.



# Architecture of the communicating object

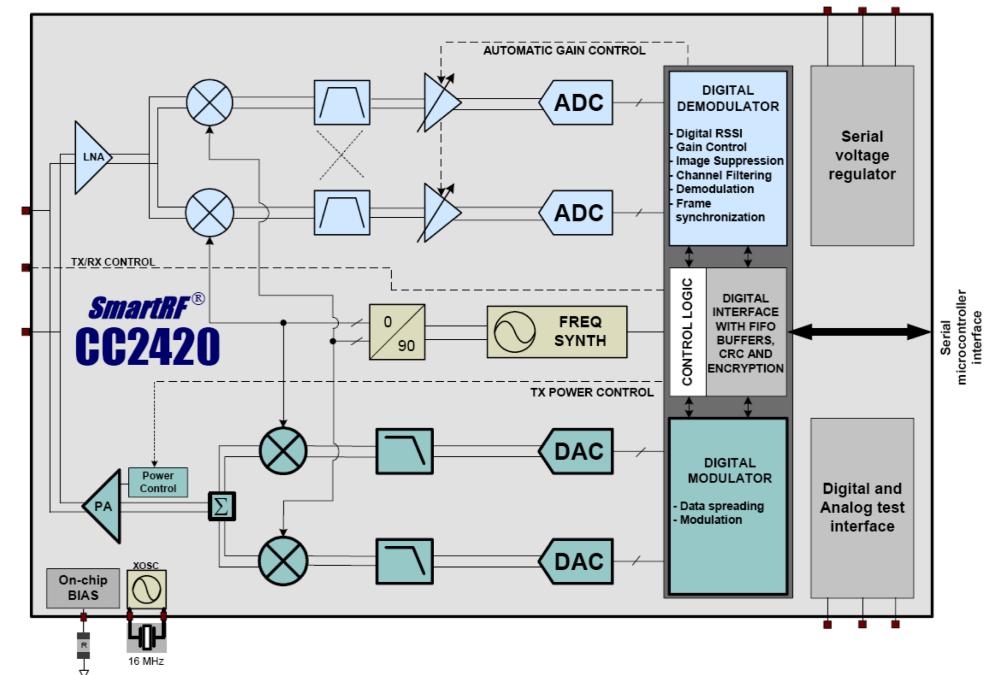
- Important characteristics of "transceivers"
  - Level of digital integration
  - Energy consumption and efficiency
    - Transition and consumption speed
    - Different levels of duty-cycle
  - Carrier frequency and data rate
  - Modulation
  - Error coding capability
  - The noise factor and sensitivity of the receiver
  - Received signal strength indicator (RSSI)
  - Support of the upper layers

# Architecture of the communicating object

- **Example: CC2420**

- Support the physical layer of IEEE 802.15.4
- 2.4 GHz frequency band with DSSS use for 250 kbps
- Integrates the voltage regulator
- Integrates MAC functionalities
  - Clear Channel Assessment (CCA)
  - Energy detection (RSSI)
  - Synchronisation
  - « Framing »
  - Encryption and Authentication
  - Retransmission (CSMA)

Sleep	Idle	Tx	Rx
20 $\mu$ A	426 $\mu$ A	8.5 – 17.4 mA	18.8 mA

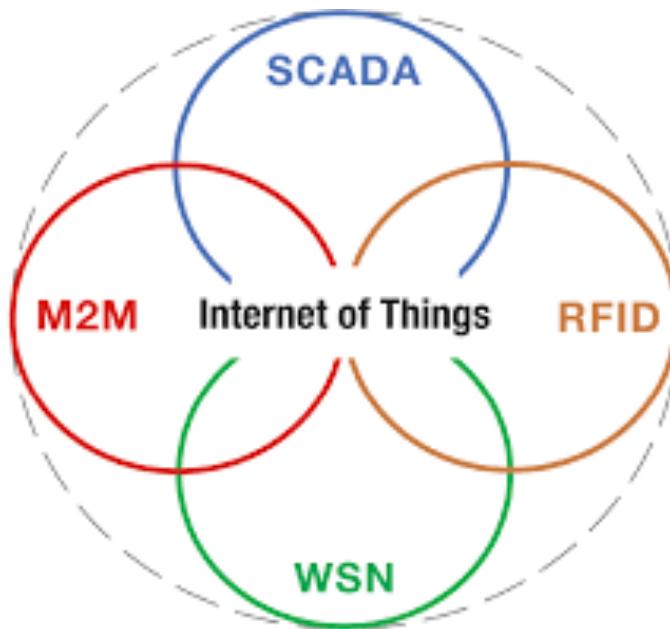


# Example of a radio transceivers

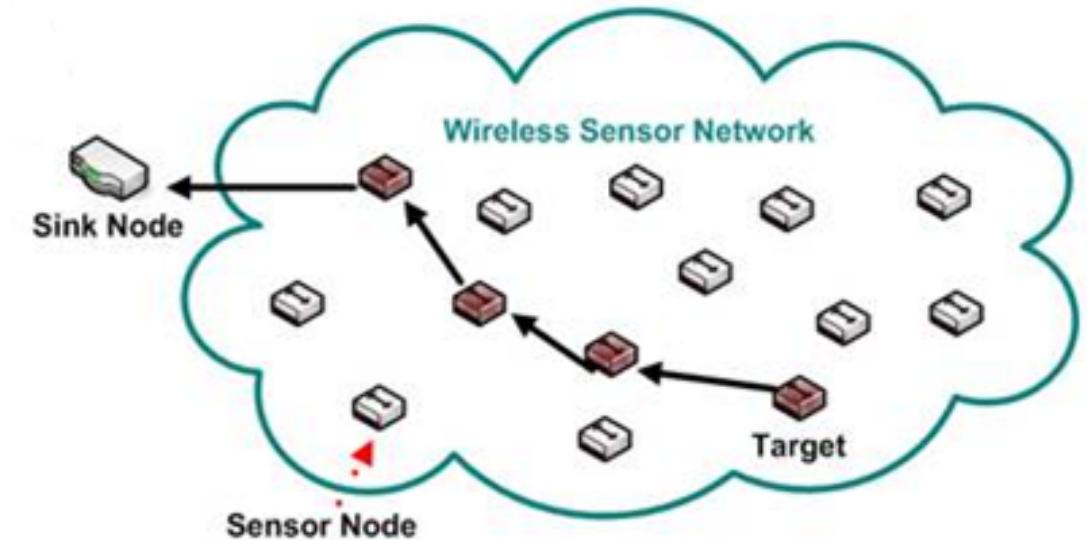
	<b>CC1000</b>	<b>CC1021</b> 	<b>CC2420</b> 	<b>TR1000</b> 	<b>XE1205</b> 
<b>Bit Rate [kbps]</b>	76.8	153.6	250	115.2	1.2 - 152.3
<b>Sleep Mode [uA]</b>	0.2 - 1 (osc. core off)	1.8 (core off)	1	0.7	0.2
<b>RX [mA]</b>	9.3 (433MHz) / 11.8 (868MHz)	19.9	19.7	3.8 (115.2kbps)	14
<b>TX Min [mA]</b>	8.6 (-20dBm)	14.5 (-20dBm)	8.5 (-25dBm)		33 (+5dBm)
<b>TX Max [mA]</b>	25.4 (+5dBm)	25.1 (+5dBm)	17.4 (0dBm)	12 (+1.5dBm)	62 (+15dBm)

# IoT and wireless sensor networks

- Wireless Sensor Networks (WSN) are part of the IoT
- WSNs are a set of communicating objects deployed in an area of interest
- The objective of the WSN is to collect and exchange information from the physical world to an information processing and analysis center.

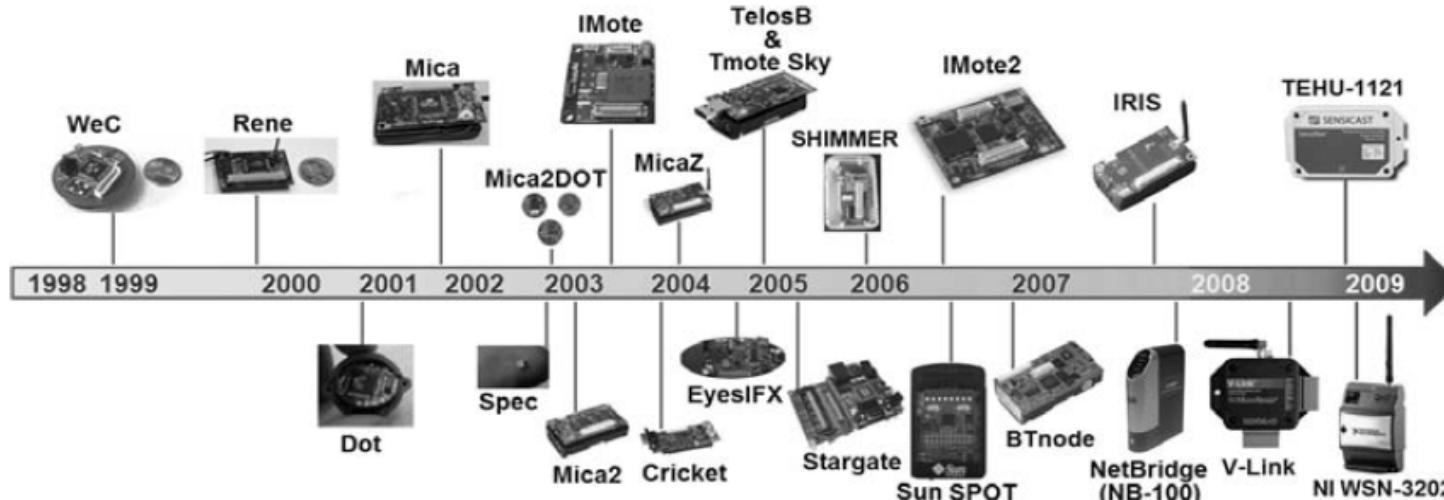


*IoT vs. WSN*



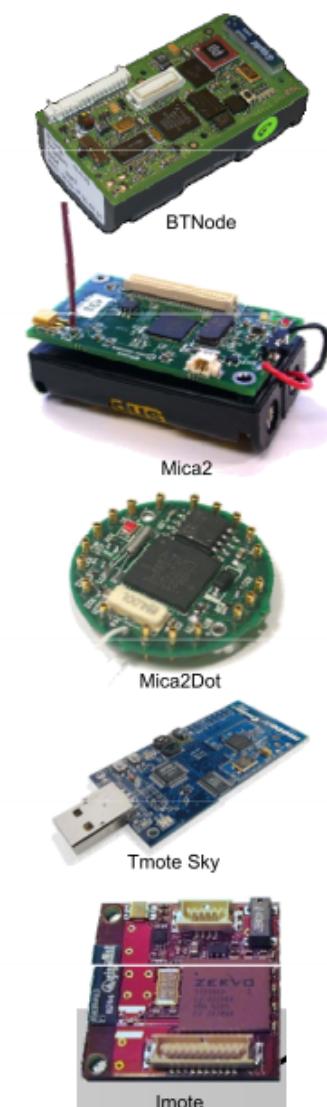
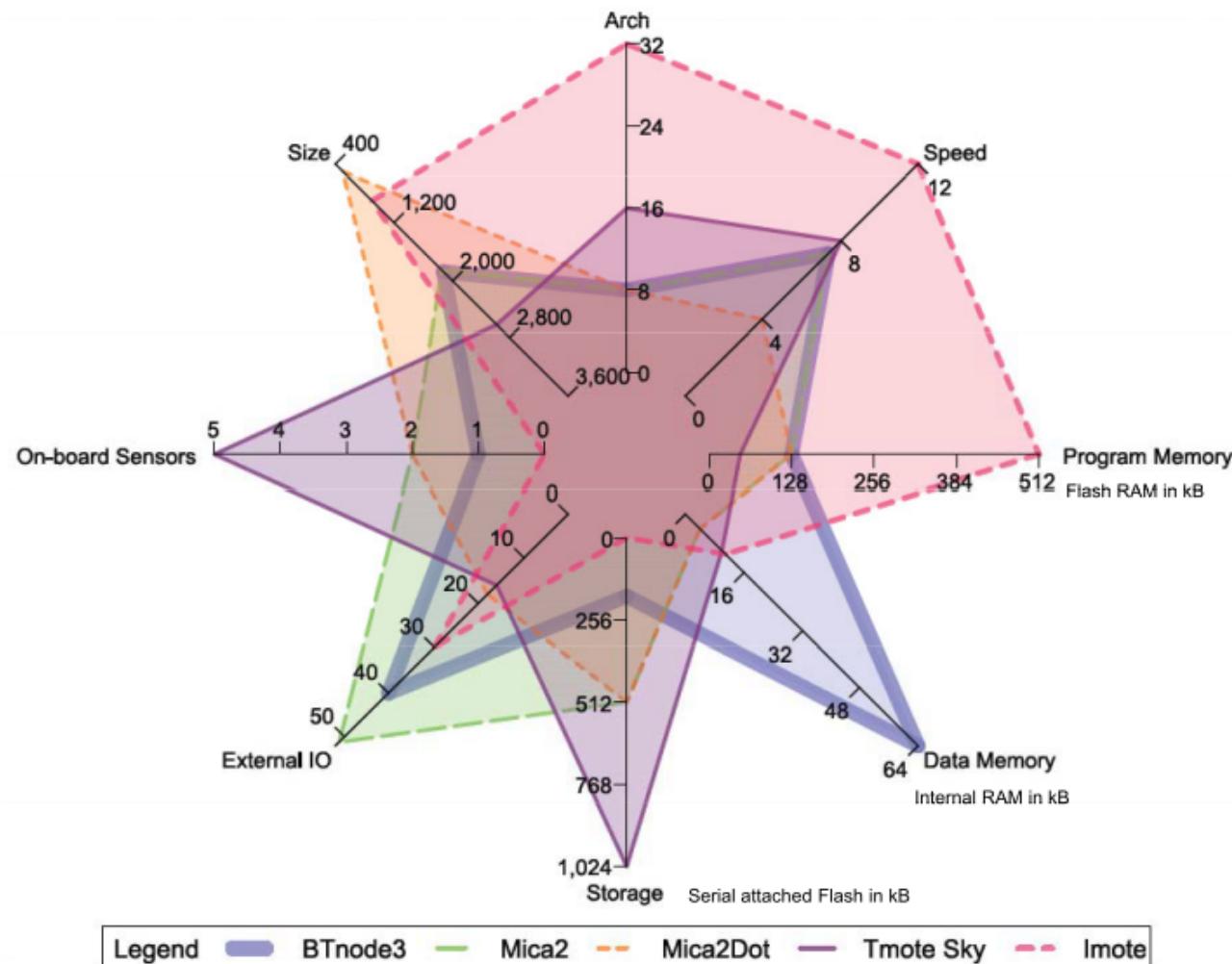
*SCADA (IIoT): Supervisory Control And Data Acquisition*

# History of WSN's platforms

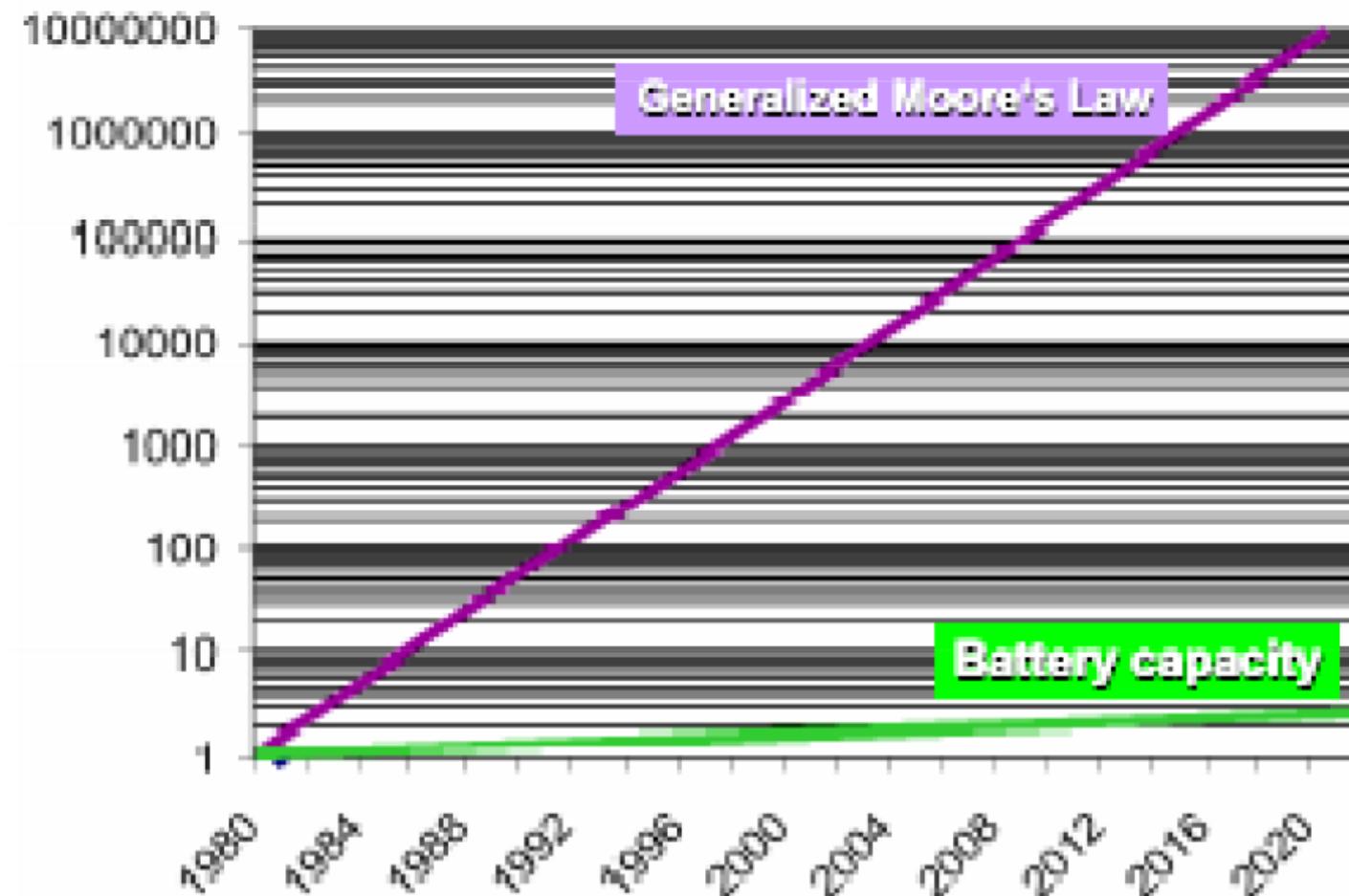


Mote type	CPU speed (MHz)	Prog. mem. (kB)	RAM (kB)	Radio freq. (MHz)	Tx. rate (kbps)
<i>Berkeley</i>					
WeC	8	8	0.5	916	10
rene	8	8	0.5	916	10
rene2	8	16	1	916	10
dot	8	16	1	916	10
mica	6	128	4	868	10/40
mica2	16	128	4	433/868/916	38.4 kbaud
micaz	16	128	4	2.4 GHz	250
Cricket	16	128	4	433	38.4 kbaud
EyesIFX	8	60	2	868	115
TelosB/Tmote	16	48	10	2.4 GHz	250
SHIMMER	8	48	10	BT/2.4 GHz <sup>a</sup>	250
Sun SPOT	16–60	2 MB	256	2.4 GHz	250
BTnode	8	128	64	BT/433–915 <sup>a</sup>	Varies
IRIS	16	128	8	2.4 GHz	250
V-Link	N/A	N/A	N/A	2.4 GHz	250
TEHU-1121	N/A	N/A	N/A	0.9/2.4 GHz	N/A
NI WSN-3202	N/A	N/A	N/A	2.4 GHz	250
Imote	12	512	64	2.4 GHz (BT)	100
Imote2	13–416	32 MB	256	2.4 GHz	250
Stargate	400	32 MB	64 MB SD	2.4 GHz	Varies <sup>b</sup>
Netbridge NB-100	266	8 MB	32 MB	Varies <sup>b</sup>	Varies <sup>b</sup>

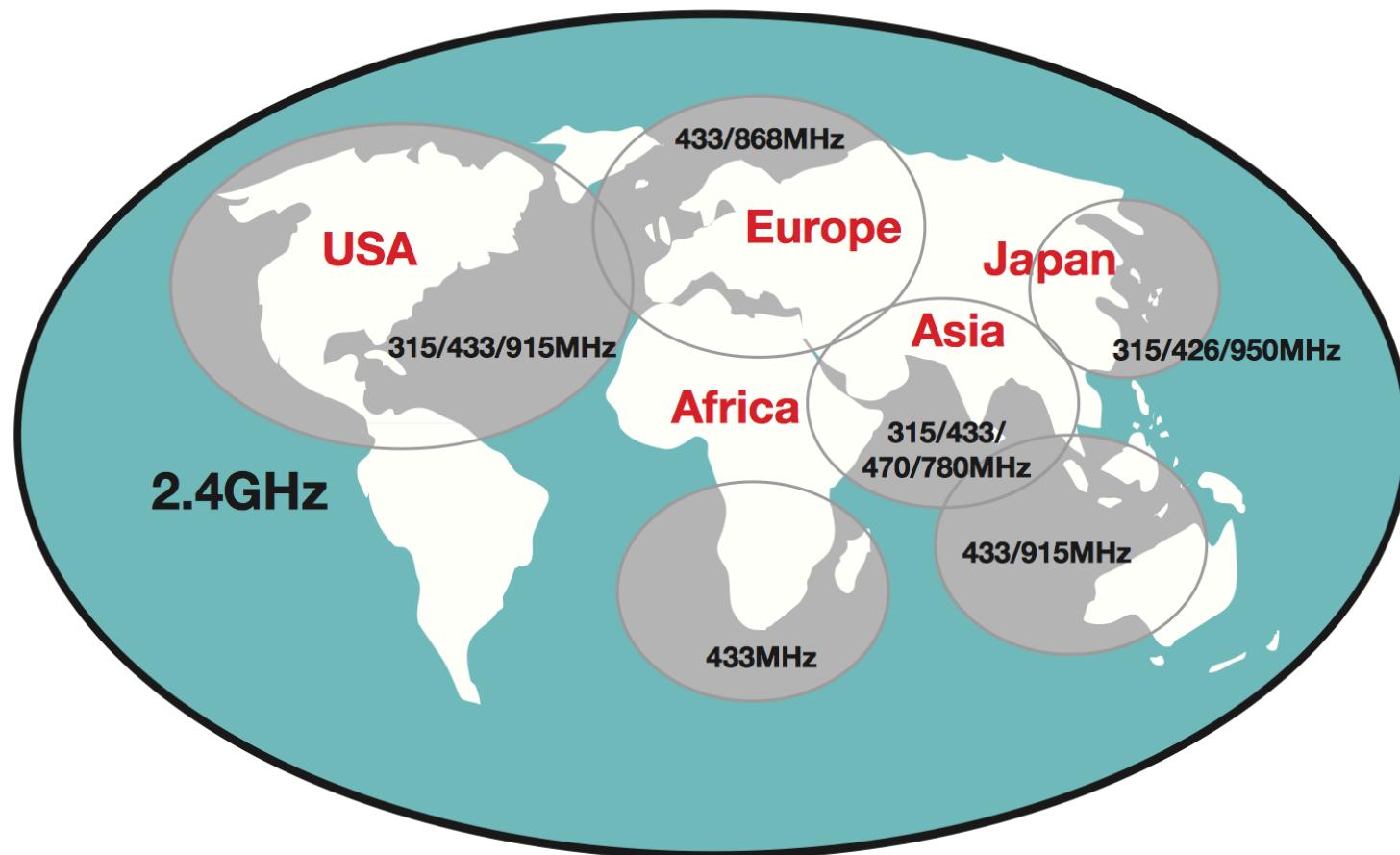
# Example of sensor devices



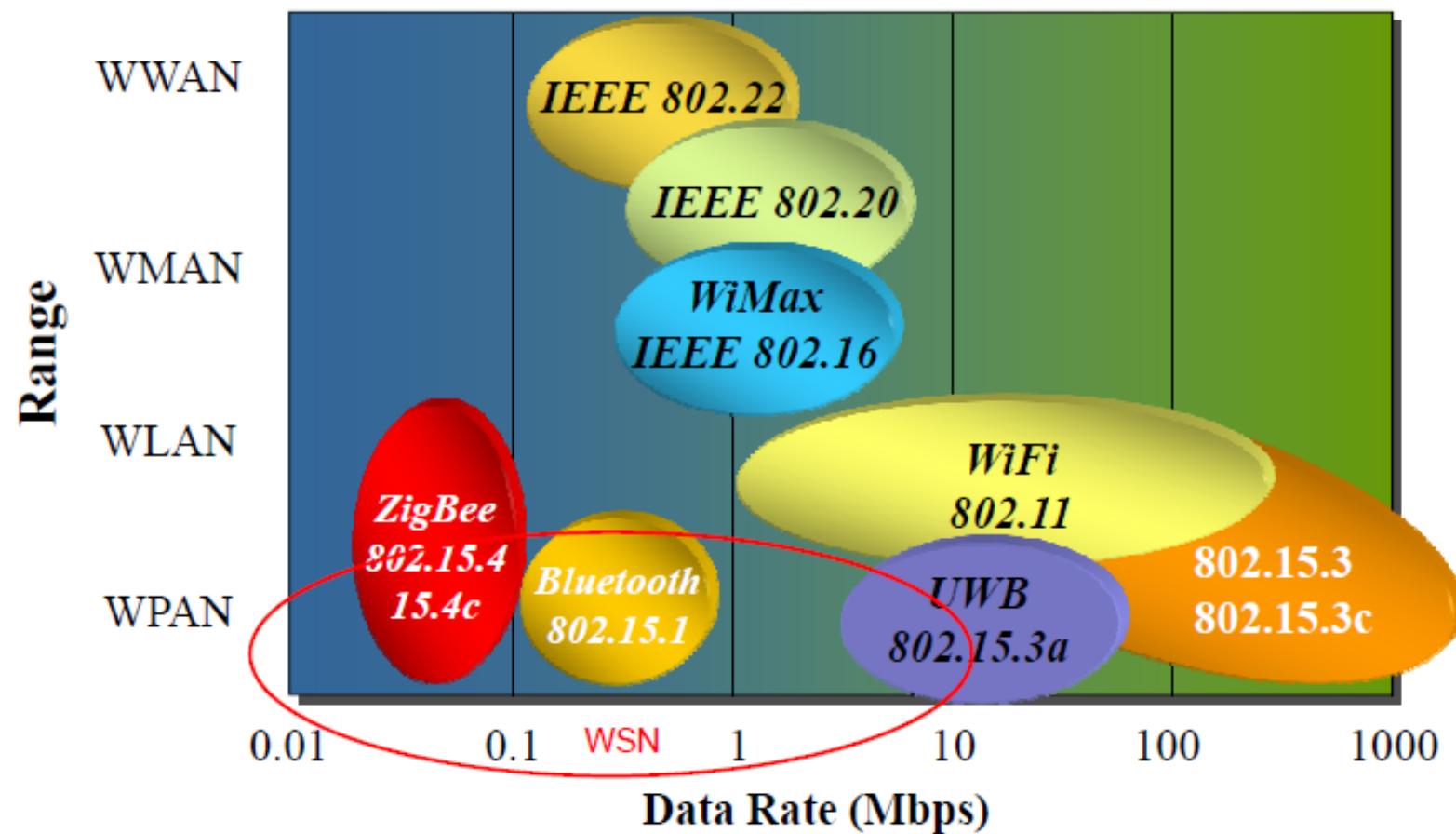
# Battery capacity evolution vs. Complexity



# ISM frequency bands



# Position of WSN in relation to other technologies



# Technologies associated with the IoT



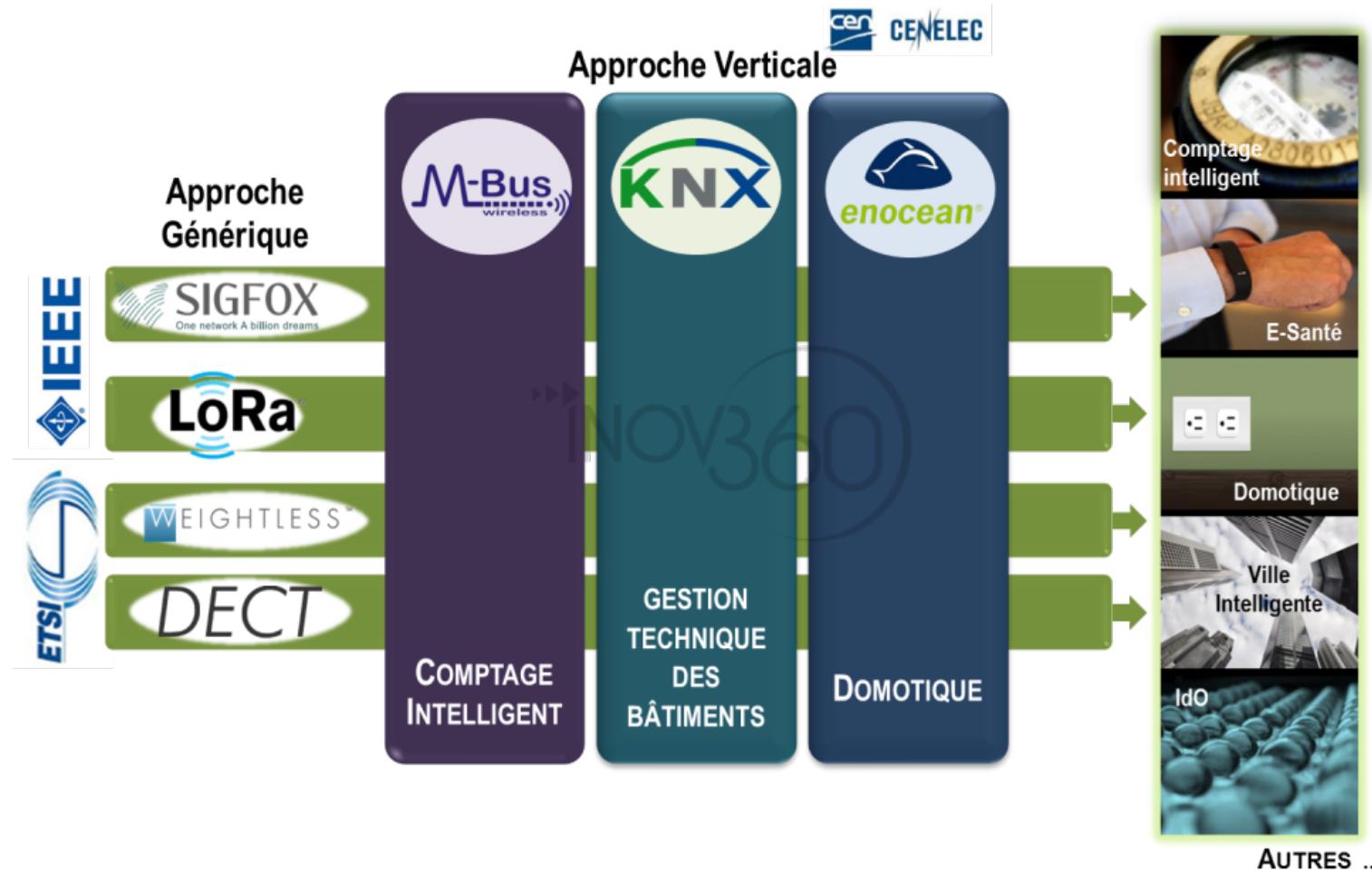
WAN: Wide Area Network  
NAN: Neighbor Area Network  
LAN: Network Area Network  
PAN: Personal Area Network

Different communication range and applications

# Technologies associated with the IoT



# Vertical and generic approaches



**ETSI**-- European Telecommunications Standards Institute  
**IEEE**-- Institute of Electrical and Electronics Engineers

# IoT communication protocols comparison

Protocol Name	Transport Protocol	Messaging Model	Security	Best-Use Cases	Architecture
AMPQ	TCP	Publish/Subscribe	High-Optional	Enterprise integration	P2P
CoAP	UDP	Request/Response	Medium-Optional	Utility field	Tree
DDS	UDP	Publish/Subscribe and Request/Response	High-Optional	Military	Bus
MQTT	TCP	Publish/Subscribe and Request/Response	Medium-Optional	IoT messaging	Tree
UPnP	—	Publish/Subscribe and Request/Response	None	Consumer	P2P
XMPP	TCP	Publish/Subscribe and Request/Response	High-Compulsory	Remote management	Client server
ZeroMQ	UDP	Publish/Subscribe and Request/Response	High-Optional	CERN	P2P

# List of IoT-Related Projects

Name of Project/Product	Area of Focus
Tiny OS	Operating System
Contiki	Operating System
Mantis	Operating System
Nano-RK	Operating System
LiteOS	Operating System
FreeRTOS	Operating System
RIOT	Operating System
Wit.AI	Natural Language
Node-RED	Visual Programming Toolkit
NetLab	Visual Programming Toolkit
SensorML	Modeling and Encoding
Extended Environments Markup Language (EEML)	Modeling and Encoding
ProSyst	Middleware
MundoCore	Middleware
Gaia	Middleware
Ubiware	Middleware
SensorWare	Middleware
SensorBus	Middleware
OpenIoT	Middleware and development platform
Koneki	M2M Development Toolkit
MIHINI	M2M Development Toolkit

# Standards of IEEE 802.15.x

1 – Bluetooth (2005)

2 – Coexistence of WPAN with other wireless devices operating in unlicensed frequency bands (2003)

3 – High rate HR-WPAN (2003)

-b – MAC amendment/enhancement (2006)

-c – Millimeter wave alternative PHY (2009)

4 – Low rate LR-WPAN (2003)

-a – Alternative PHY (2007)

-b – Revision & enhancement for LR-WPAN-2003 (2006)

-c – Alternative PHY to support Chinese frequency bands (2009)

-d – Alternative PHY to support Japanese frequency bands (2009)

-e – MAC amendment & enhancement for LR-WPAN-2006 (in progress)

-f – Active RFID system; new PHY and enhancement to LR-WPAN-2006 for RFID (in progress)

-g – Smart utility networks/neighborhood SUN (2009 ~ in progress)

5 – Mesh topology capability in WPAN (2009)

6 – Body area network (2007 ~ in progress)

7 – Visible light communication VLC (2009 ~ in progress)

*IGthz* – TeraHz Interest Group (2008 ~ in progress)

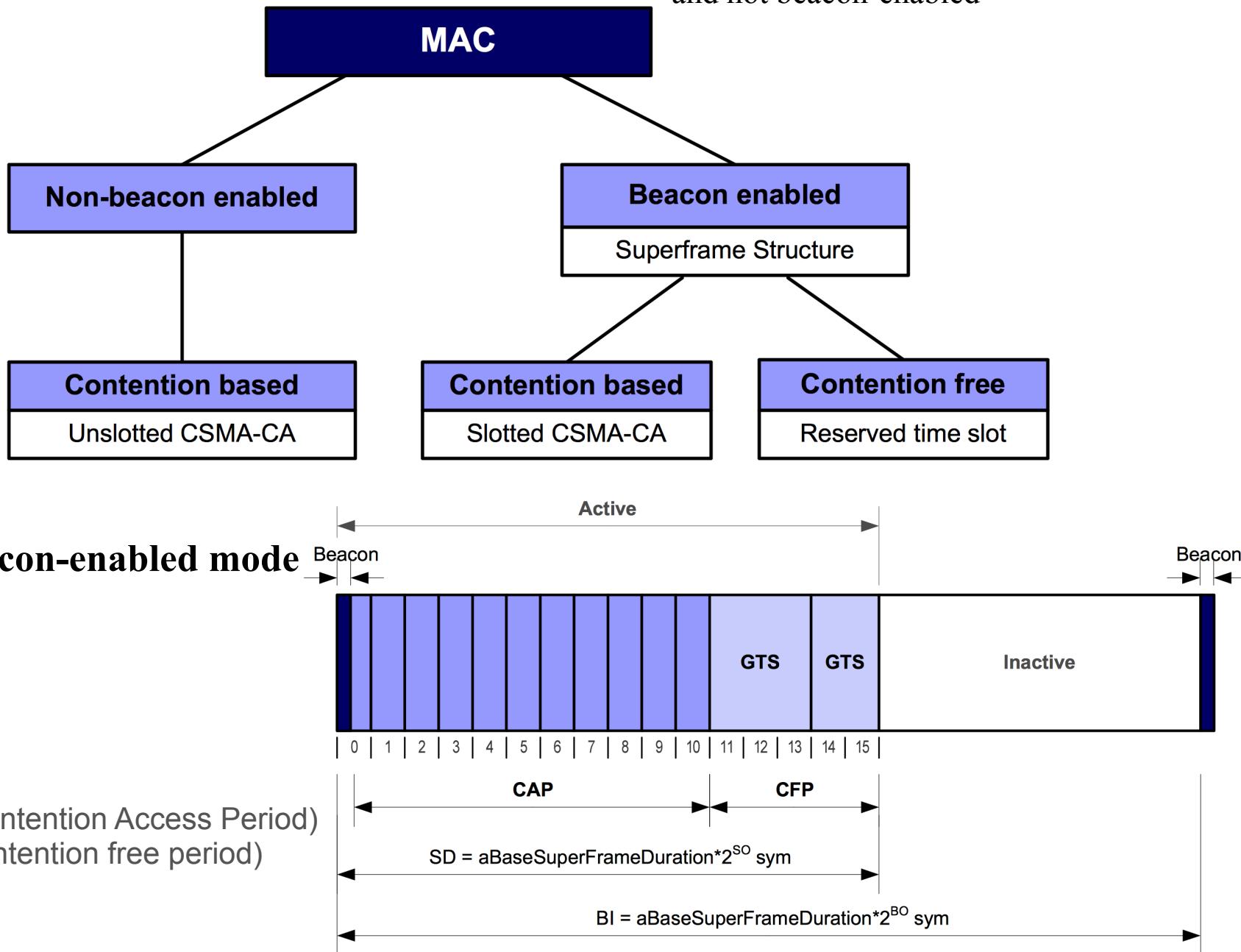
*WNG* – Wireless next generation (2008 ~ in progress)

# IEEE 802.15.4

- Developed by Working Group 4 in 2003 and corrected in 2006
- Defines 2 layers: PHY and MAC for low speed networks
- The characteristics of this standard are:
  - Data rate varies between: **250 Kbps, 40 Kbps and 20 Kbps**
  - Two addressing modes: 16-bit and 64-bit
  - Automatic establishment of the network by the coordinator
  - Optimal management of energy consumption
  - 16 communication channels available in the 2.5GHz band,  
10 channels for the 915MHz band and 1 channel in the 868 MHz band.
  - It operates on the 3 frequency bands:
    - 868-868.6 MHz (eg. Europe) with 20 Kbps
    - 902-928 MHz (eg. North America) with 40 Kbps
    - 2400-2483.5 MHz (worldwide) with 250 Kbps

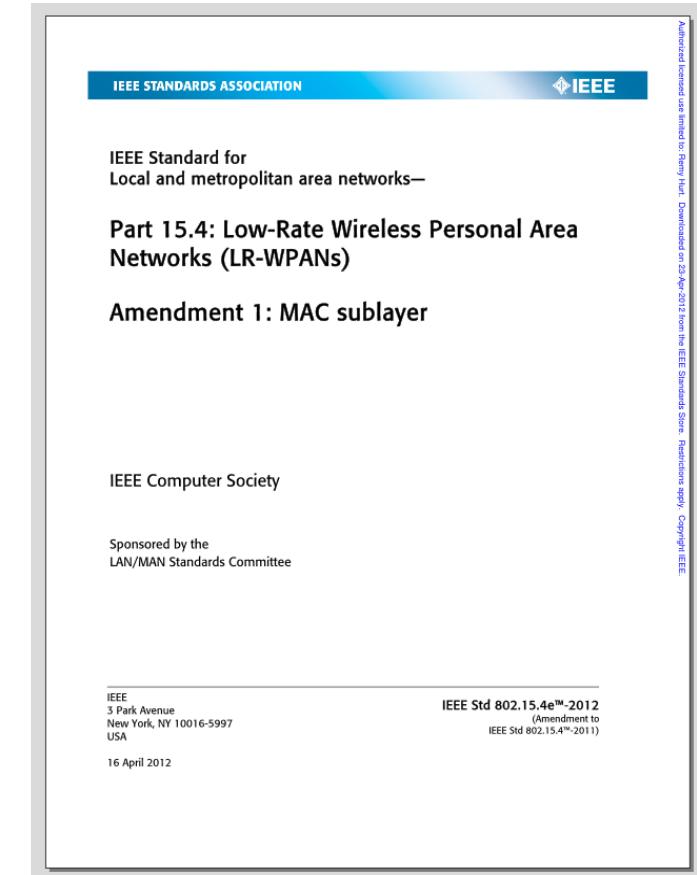
# IEEE 802.15.4

CSMA/CA-based channel access mode 2 access modes:  
 - beacon-enabled (Slotted CSMA/CA)  
 and not beacon-enabled



# IEEE 802.15.4e : TSCH – Time Synchronized Channel Hopping

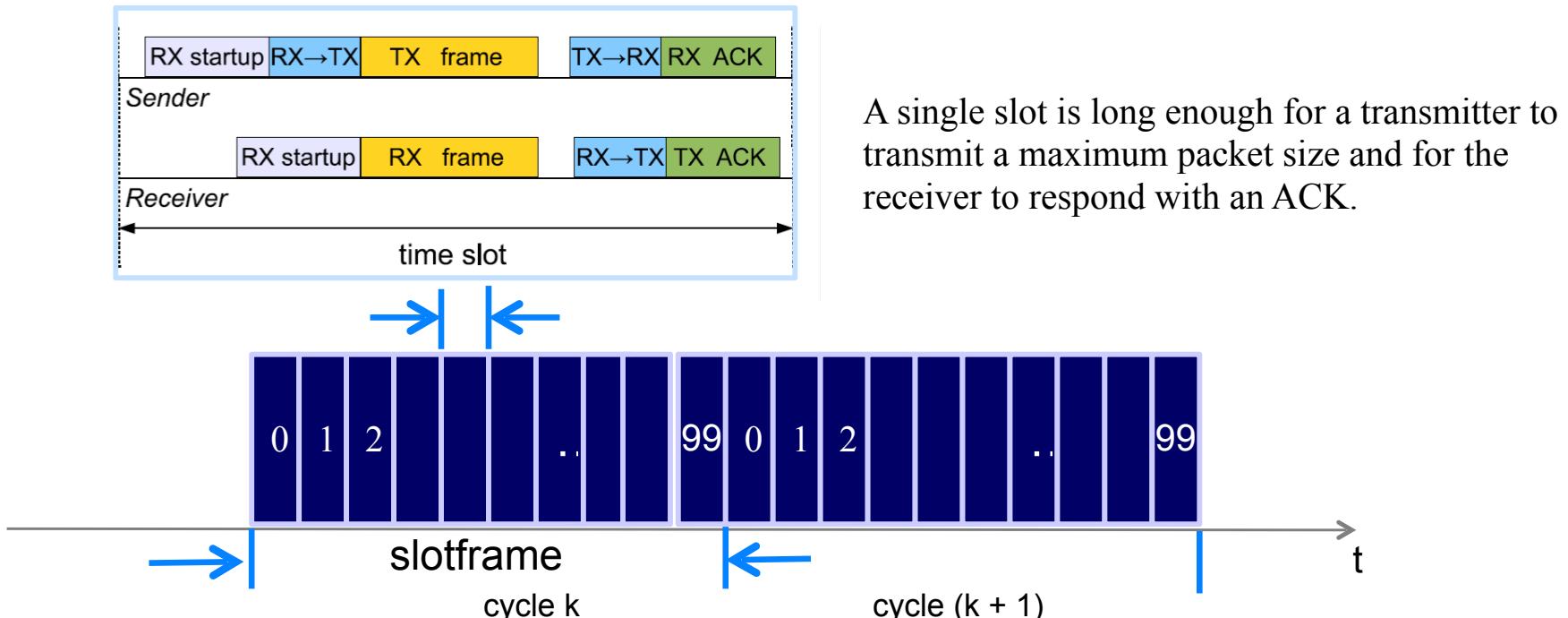
- WG was created in 2008 to review the IEEE802.15.4-2006 standard.
- The objective
  - Deterministic network more adapted to industrial applications
    - Adapted to a priori known data streams
    - Strong synchronization between objects
  - Scalable with distributed resource management
  - More controllable with a compromise between energy, robustness and time delay
  - Flow isolation (Network traffic engineering)
  - Predictable energy consumption
  - Compressed sleep/wake cycle



# IEEE 802.15.4e : TSCH – Time Synchronized Channel Hopping

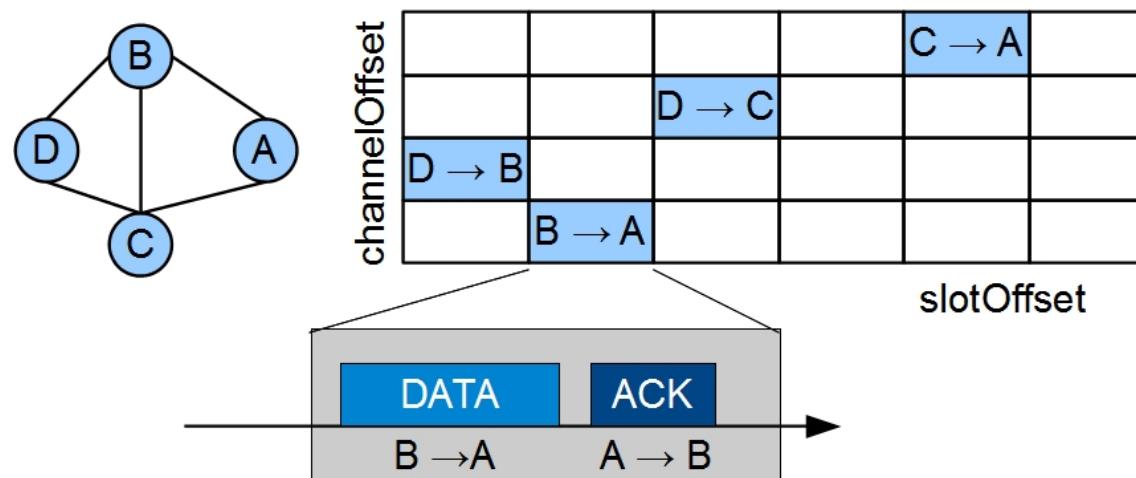
- **TSCH: TimeSlotted (Synchronized)**

- Time is divided into slots
- All sensors are synchronized to the “**slotframe**”
- *Slotframe: is a group of time slots that repeats itself all the time.*
- The number of time slot per slotframe is configurable



# IEEE 802.15.4e : TSCH - Scheduler

- Each communication flow has a scheduler
  - A scheduler is based on a matrix where the columns are indexed by ***slotOffset*** and the rows by ***channelOffset***
  - Each cell can be assigned to a pair of sensors in a given direction
  - The cell can be used by one or more pairs of sensors



**Sensors are only woken up if necessary  
and depending on the scheduler**

# IEEE 802.15.4e : TSCH -Scheduler

- There are different approaches to designing the scheduler:
  - Centralized (eg. PCE-Path Computation Elements based)
    - PCE is responsible for establishing and maintaining the Scheduler
    - Effective for static networks
  - Distributed (eg. MPLS)
    - Each node decides locally which cell should be used to communicate with these neighbors.
    - Adapted for mobile networks with multiple gateways
    - Scalable

**IEEE 802.15.4e defines how the MAC layer runs the scheduler, but it does not indicate how the scheduler should be designed.**

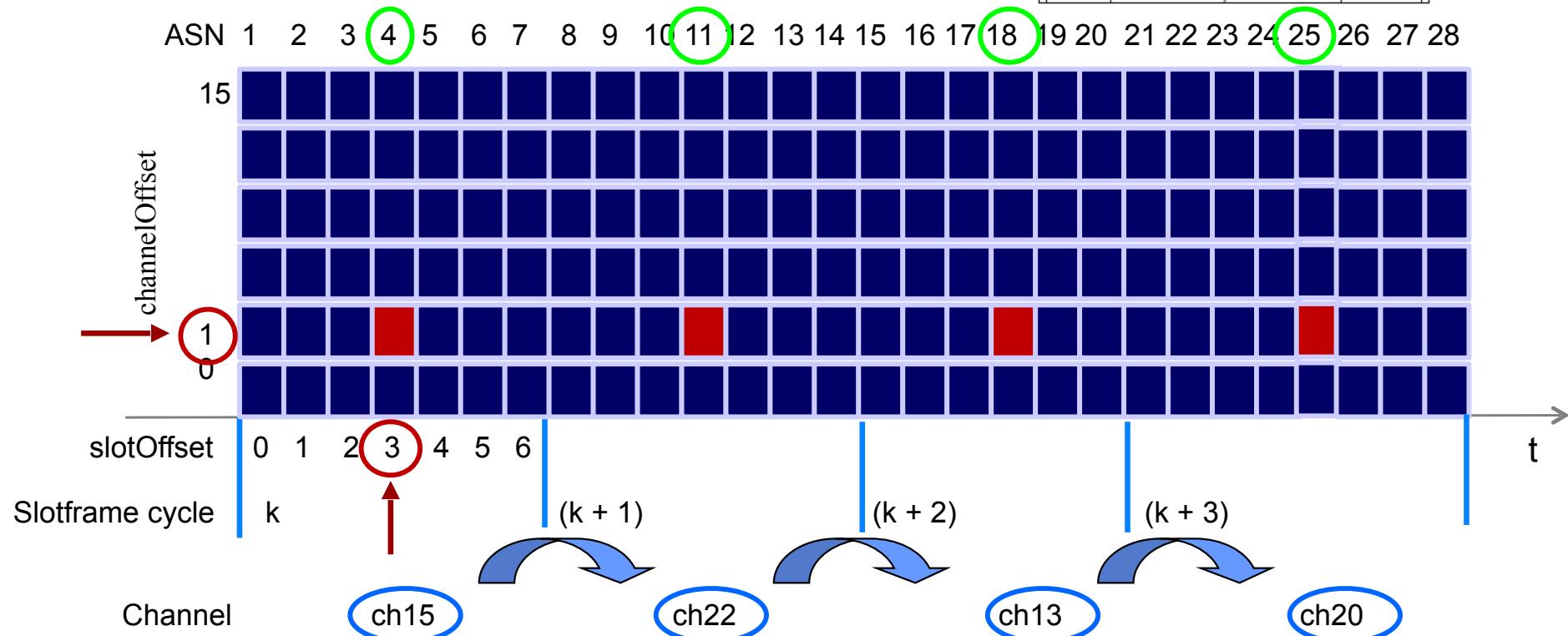
# IEEE 802.15.4e : TSCH - Channel Hopping

- The channel offset is transformed into frequency using the translation function (f)

$$f = F \{ (\text{ASN} + chOf) \mod n_{ch} \}$$

Table I. Frequency Translation

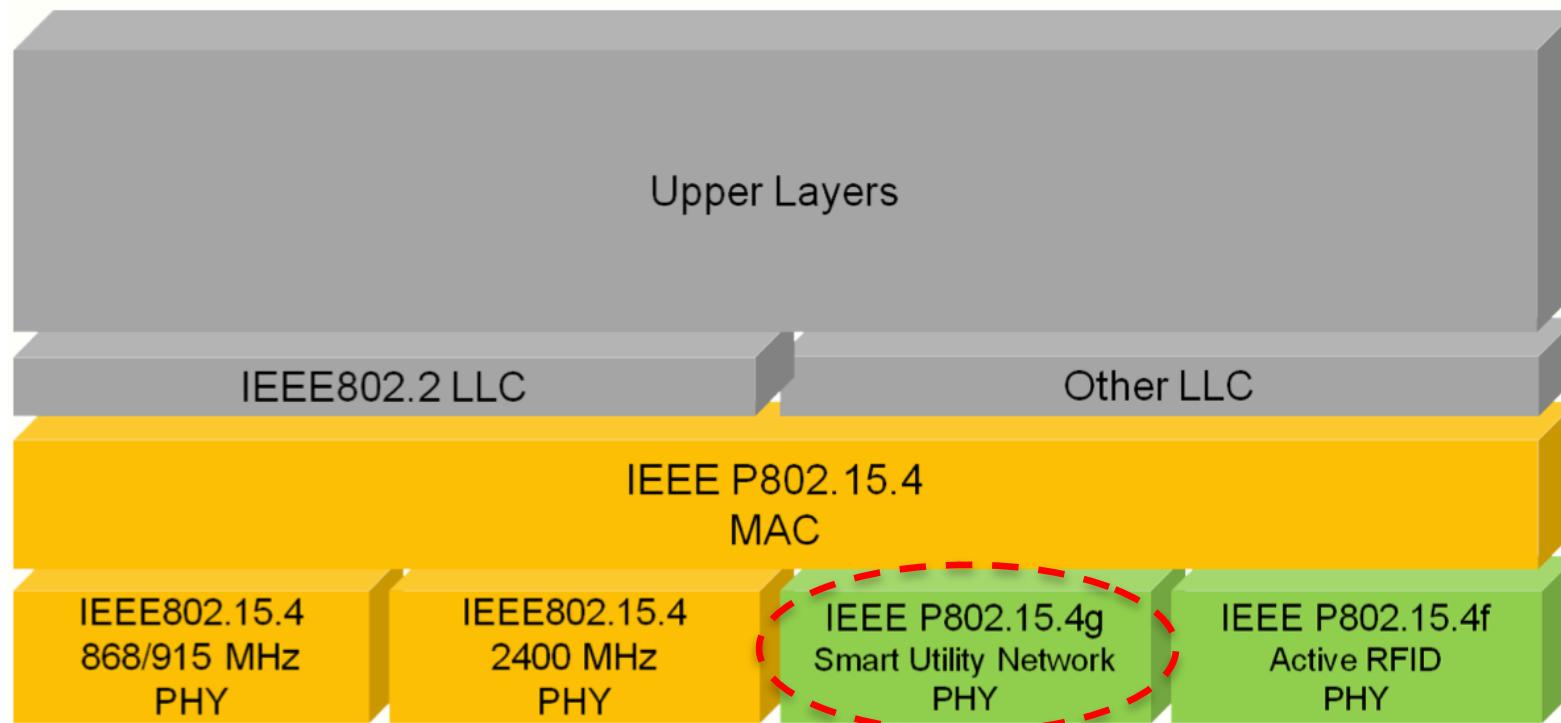
$k$	$\text{ASN}$	$chOf$	$f$
0	4	1	5
1	11	1	12
2	18	1	3
3	25	1	10



# IEEE 802.15.4g : SUN (Smart Utility Networks)

- The SUN PHY supports multiple data rates in bands ranging from 450 MHz to 2450 MHz.
- It has three modes:
  - *Orthogonal Frequency Division Multiplexing (OFDM-MR) PHY* - provides higher data rates with high spectral efficiency
  - *Multi-rate and multi-regional quadrature offset phase-shift keying (MR-O-QPSK) PHY* - shares the features of IEEE 802.15.4-2006, making multi-mode systems more cost-effective and easier to design
  - *Multi-rate and multi-regional frequency shift keying (MR-FSK) PHY* - provides good transmission power efficiency
- It is possible to use several different PHYs operating in the same location and frequency band.
  - To limit interference, a Multi-PHY management (MPM) is defined for SUN networks to allow inter-PHY coexistence.

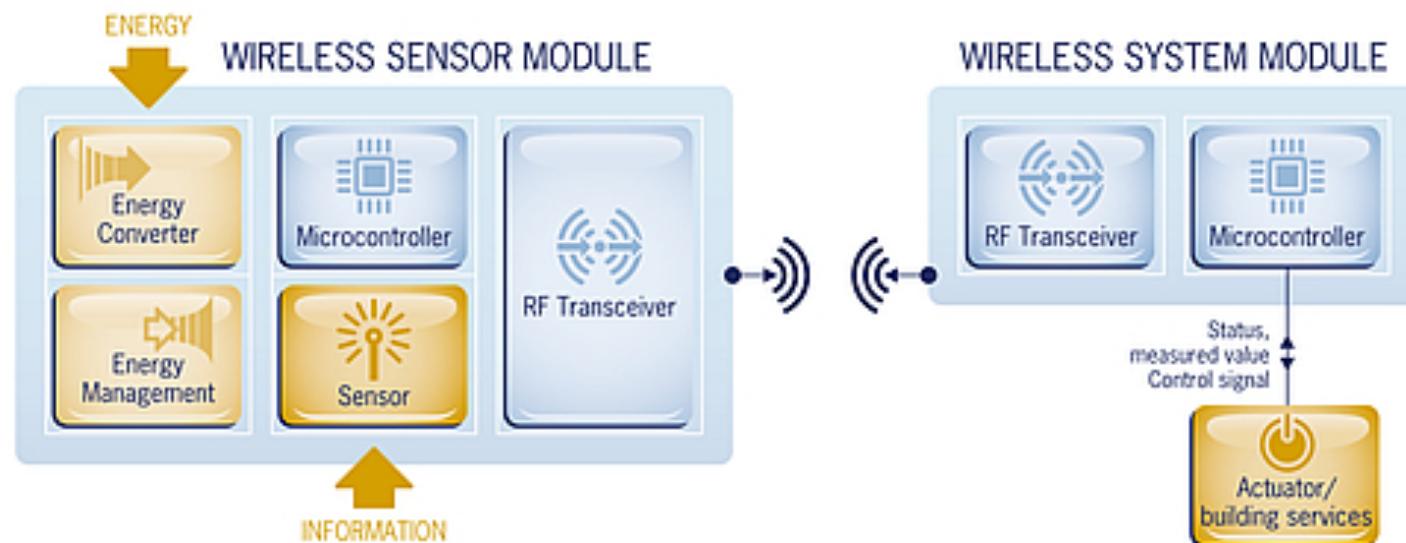
# IEEE 802.15.4g : SUN (Smart Utility Networks)



# EnOcean



- Ultra-low power radio technology based on ***energy harvesting***
- It uses the frequencies :
  - 868Mhz for Europe and 315Mhz for the USA
- Battery-free and wireless technology
  - Radio modules use available energy (mechanical, solar, etc)
  - These different energy sources are sufficient to power each module to transmit the following information



# Z-Wave

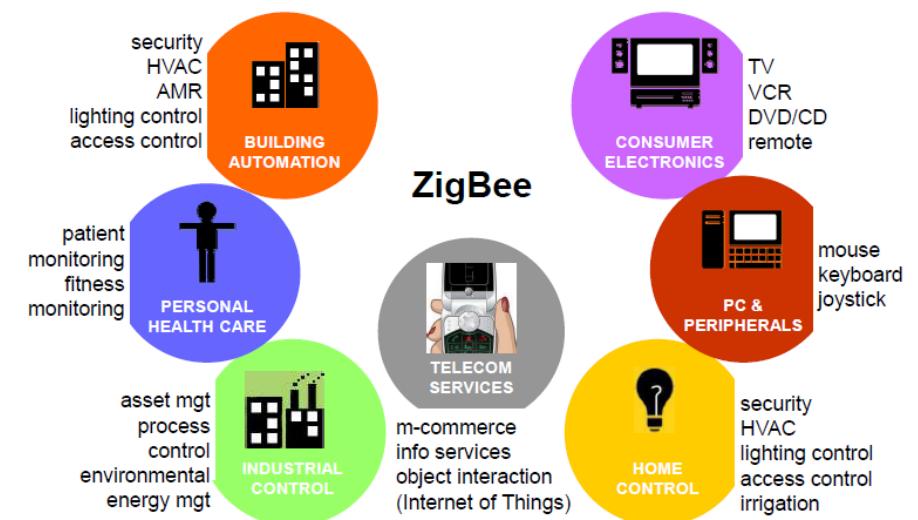
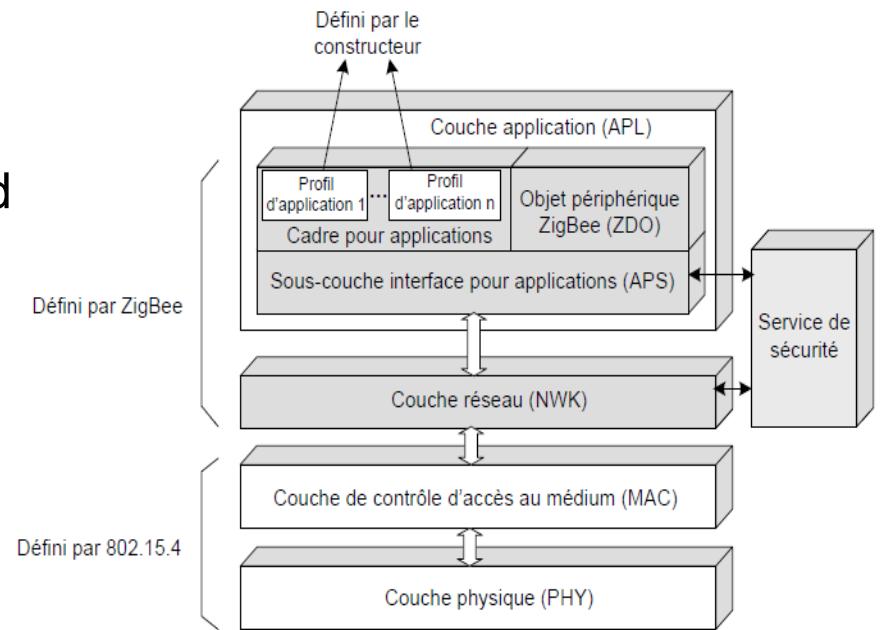


- Radio protocol designed for home automation: lighting, heating, security,....
- Low data rate between 9 and 40 kbps
- Uses frequencies: 868 MHz in Europe, 908 MHz in the US
- Range: around 50 m
- Mesh topology increases range and reliability.
- CSMA/CA access method



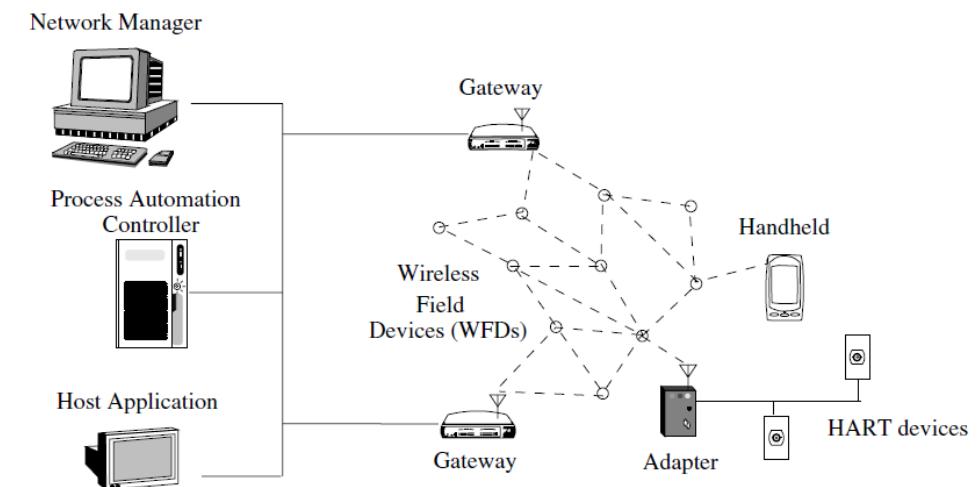
# ZigBee

- Proposed by Zigbee Alliance in 2004
- It is located above the IEEE 802.15.4 standard
- It defines the 2 layers :
  - network and application layers
- Several functionalities at the network level are proposed:
  - Association and disassociation
  - Route discovery and maintenance
  - Introduce of packet security
- Three roles for Zigbee nodes:
  - Coordinateur (ZC)
  - Routeur (ZR)
  - End-Device (ZD)
- Main applications:
  - Home automation
  - Industrial Automation
  - Health



# WirelessHART (Highway Addressable Remote Transducer)

- An extension that introduces wireless communication into the **HART** standard.
- HART defines the set of industrial applications that use communication protocols with the constraint of **real-time** and a node density of up to **20 million nodes**.
- WirelessHART uses the PHY characteristics of the **IEEE802.15.4**
- However, the channel access protocol is based on **TDMA**
  
- It defines 5 roles for the nodes:
  - WirelessHART Field Devices (WFD),
  - Gateway, Network Manager,
  - WirelessHART Adapter, Handhelds
  
- The network topology is mesh



# 6LowPAN (IPv6 over Lowpower wireless area networks)

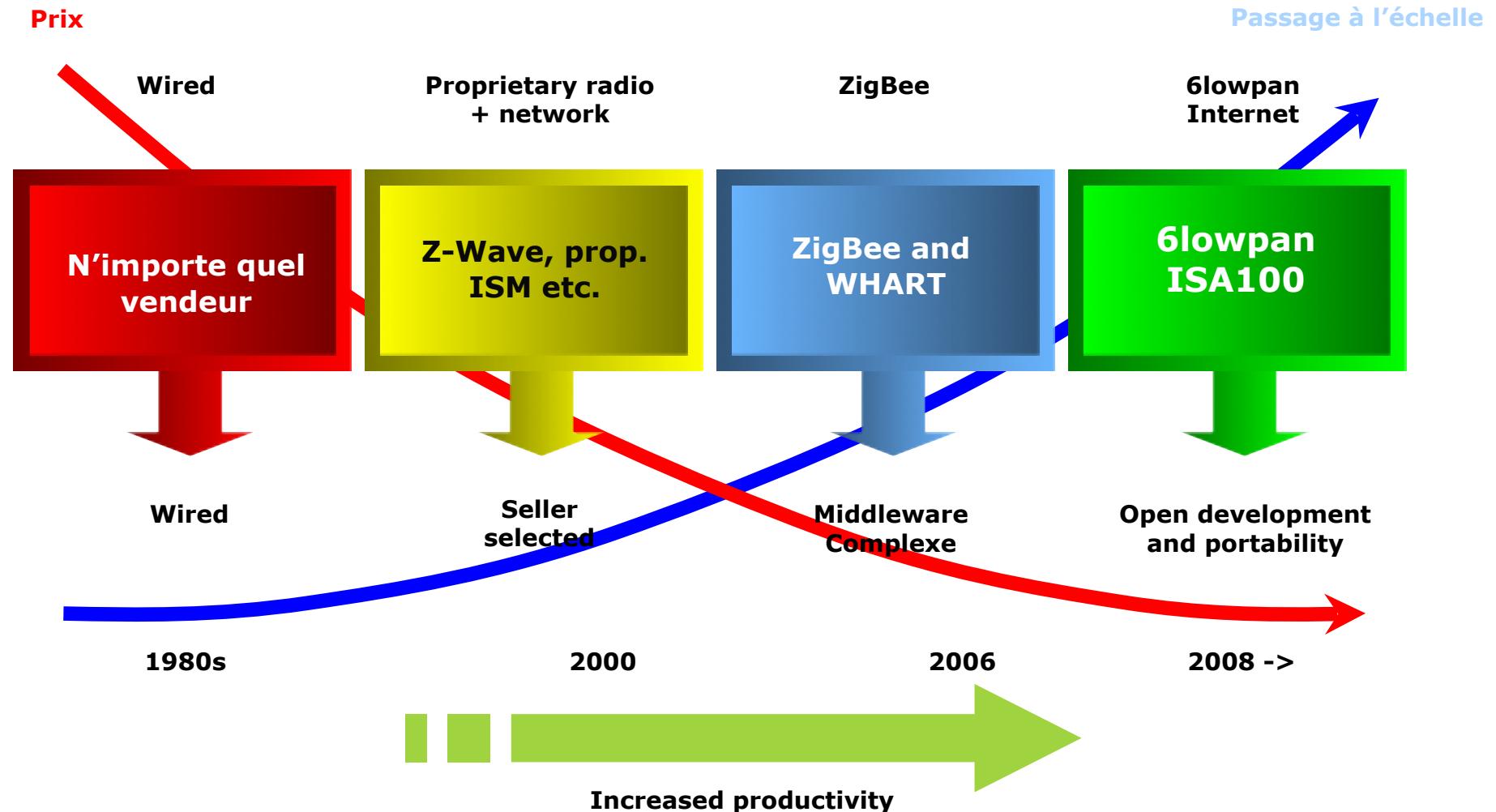
- 6LowPAN is available since 2007 in RFC 4919
- Efficiently uses IPv6 in low-speed networks (802.15.4)
- The challenge is to introduce **40-byte** addresses in a **127-byte** frame defined by IEEE 802.15.4 (the IPv6 frame size is 1280 bytes).
- Compression techniques are used to reduce the address size from **40** to **4 bytes**.
- It proposes a **ROLL** (Routing over Low-power and Lossy Networks) routing protocol with dynamic metrics
- Nodes are directly accessible via the Internet without the need for a gateway (IP-based sensor)
  
- Enables the emergence of the Internet of Things
- It has allowed to create alliances such as :
  - **IP for Smart Objects (IPSO) Alliance, IP500 Alliance, Open Geospatial Consortium (OGC), M2M (lancé par European Telecommunication Standard Institute)**

# Remember: Why IPv6?

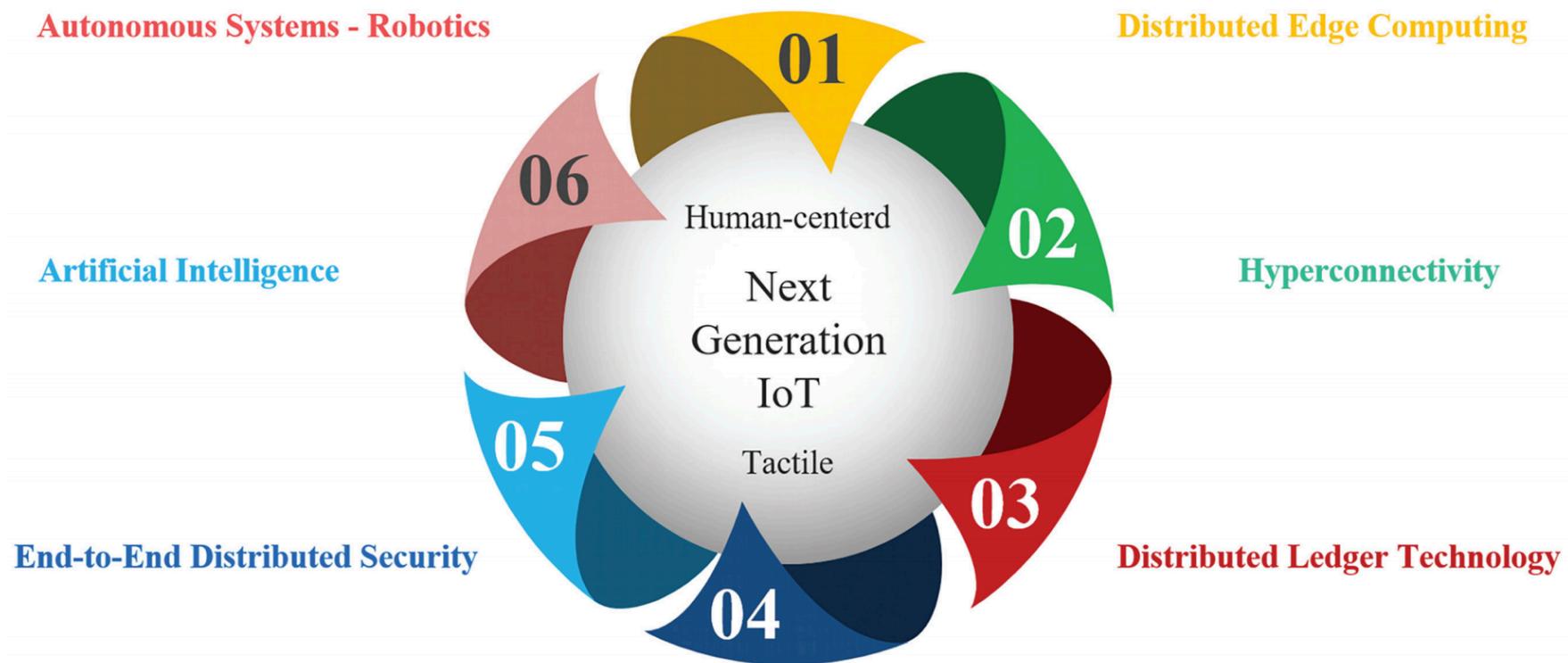
## Main changes brought by IPv6

- Length and format of addresses
  - IPv4 = 32 bits, IPv6 = 128 bits
- Types of addresses
  - `x:x:x:x:x:x:<prefix, site, nœud>`
- IPv6 packet header
- DHCP
  - routers multicast IPv6 network prefixes
- Autoconfiguration
  - NAT : Disappears, we go back to the original IP model
  - No more private addressing, each IPv6 node has a globally unique IPv6 address on the planet.
- Fragmentation
  - Disappears, each node must adjust its packets according to the smallest MTU on the path (Path MTU)
- ARP Protocol
  - Replaced by a mechanism using multicast messages
- Mobility
  - MobileIPv6 allows a node to keep the same IPv6 address even if it changes IPv6 network.
- QoS
  - New Flow Label field (20 bits). Its purpose is to establish a correspondence with level 2 networks (e.g. MPLS) for the classification and prioritization of packets.
- Security
  - Each IPv6 stack MUST integrate IPsec support

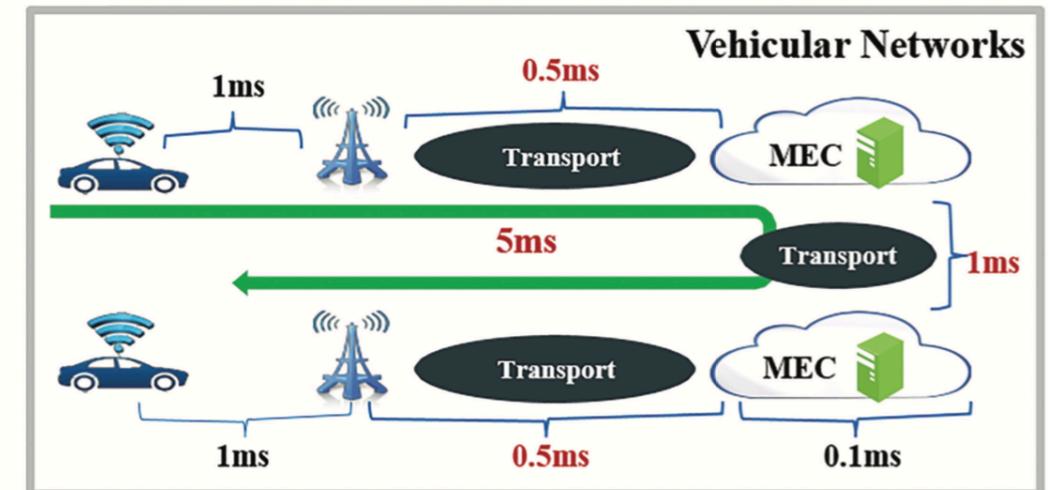
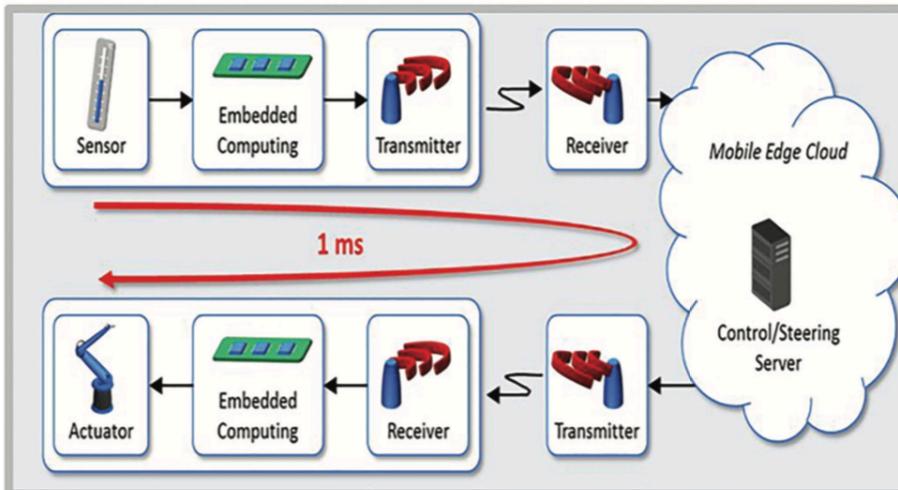
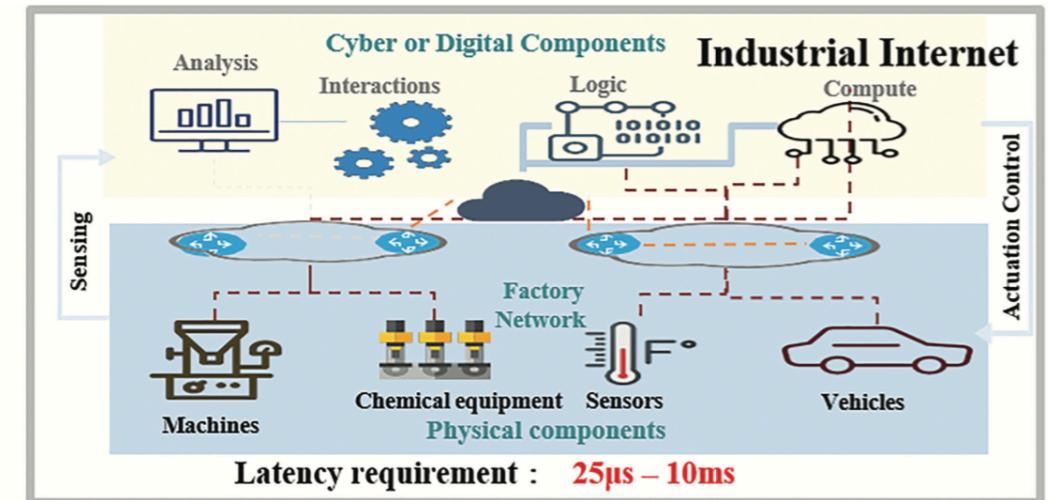
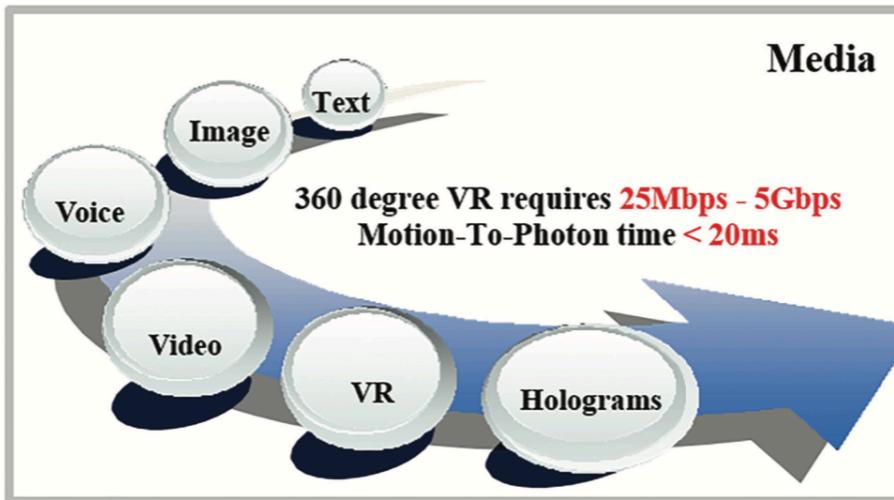
# The evolution of wireless sensor networks



# Next generation IoT technology convergence



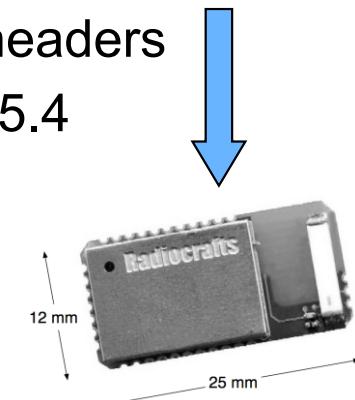
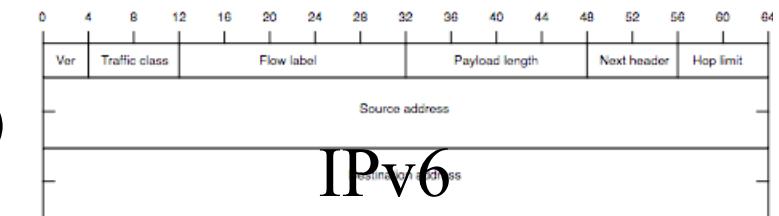
# New applications for NGI and IoT/IIoT



# 6LoWPAN

# What is 6LowPAN?

- 6LowPAN : IPv6 over Low-Power wireless Personal Area Networks
- Defined by IETF standards:
  - RFC 4919, 4944
  - ietf-6lowpan-hc (RFC 6282) and -nd (RFC6775)
  - ietf-roll-rpl (RFC 6550)
- Provides mechanisms for encapsulation and compression of headers
- Allows IPv6 packets to be sent or received via the IEEE 802.15.4 communication protocol
- Supports the Sockets API standard
- Minimal code and memory usage
- Direct Internet and end-to-end integration
  - Several possible network topologies



# The interest of 6LoWPAN technology

- Low-power RF + IPv6 = The Wireless Embedded Internet
- 6LoWPAN allows to implement this solution
  
- The interest of 6LoWPAN is:
  - Open, long life, reliable standards
  - Seamless Internet integration
  - Network stability
  - Scalability
  - End-to-end communication



# Objectives

- Define layer adaptation (fragmentation/assembly) to meet IPv6 MTU constraints
- Specify methods to benefit from IPv6 address auto-configuration
- Propose header compression techniques
- Determine a better implementation for the IPv6 stack
- Define methods to ensure meshing under the IP layer

# Challenges

- No method to run IP on an IEEE 802.15.4 network
  - Worst case .15.4 **81 bytes** PDU, IPv6 requires **1280 bytes** MTU
- The IP layer and upper layers cannot be directly inserted into the 802.15.4 frame.
  - IPv6 **40 bytes**, TCP **20 bytes**, UDP **8 bytes** + other layers (security, routing, etc.) leave little space for data
- Not all ad hoc routing protocols are desired in LoWPAN.
- Service discovery methods for LoWPAN
  - Are based on XML, which requires computing and memory capacity, etc.
- Limited configuration and need for network management
- Multi-hop security required

# Challenges

Impact Analyse	Addressing	Routing	Security	Network management
Low power (1-2 year of battery life)	Backup limitations, low overload	Support of duty-cycle (sleep period), low overload	Simplicity (CPU usage), low overload	Management with sleep period constraint, low overload
Low cost (<\$10/unit)	Stateless address generation	Routing tables of small size or not	Ease of use, simple bootstrapping	Space constraints
Low rate (<300kbps)	Compressed addresses	Low routing load	Low packet overload	Low network overload
High density (<20-40 unit/m <sup>2</sup> )	Wide address space - IPv6	Allows for scaling	Robuste	Easy to use with scaling
Interaction with IP network	Routable address in the IP network	Seamless IP routing	Supports end-to-end security in the IP network	Compatible with SNMP, etc.

# Several IP advantages

- Strong interoperability
  - Other on-board wireless network equipment 802.15.4
  - Any equipment in the IP network with WiFi, Ethernet, GPRS, ...
- Establishing security
  - Authentication, access control, and firewall mechanisms
  - Network design and access policy
- Establishment of naming, addressing, translation, etc.
- Establishing proxy architectures for services at the upper layers
  - Load balancing, mobility
- Establishment of the data and service model at the application level
  - HTTP/HTML/XML/SOAP/REST,
- Establishment of network management tools
  - Ping, Traceroute, SNMP, OpenView, NetManager, Ganglia, ...
- Transport protocols
  - End-to-end reliability
- Most "industrial" standards (wired or wireless) support IP

# Why 802.15.4?

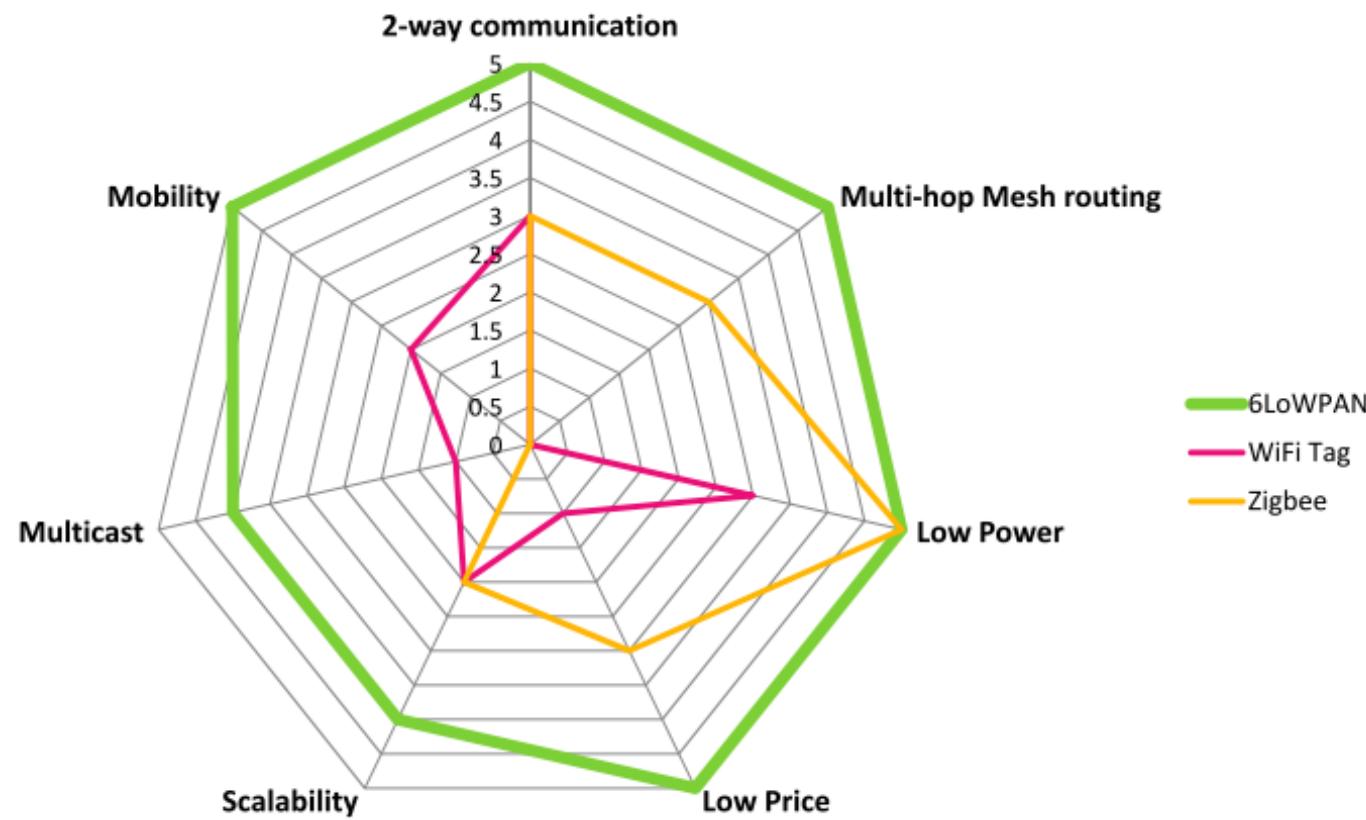


	802.15.4	Bluetooth	UWB	WiFi	Ethernet
Classe	WPAN	WPAN	WPAN	WLAN	LAN
Durée de vie (jours)	100-1000+	1-7	Alimenter	0.1-5	Alimenter
Densité du réseau	65535	7	243	30	1024
BW (kbps)	20-250	720	11,000+	11,000+	100,000+
Portée (m)	1-75+	1-10+	10	1-100	185 (wired)
Objectifs	Low Power, Grande échelle, Faible coût	Remplacer le câble	Remplacer le câble	Débit	Débit

# Important elements of IP over 802.15.4

- The header
  - The IPv6 header size is 40 bytes [RFC 2460]
  - The size of the entire 802.15.4 frame is 127 bytes [IEEE ].
  - Often the size of the Payload is small
- Fragmentation
  - Interoperability: applications do not need to know the constraints of physical links
  - IP packets can be large compared to the frame size 802.15.4
  - IPv6 needs links that support 1280 byte packet size [RFC 2460].
- Enables mesh routing at level 2 under an IP topology
  - 802.15.4 subnet can use multiple radio hops for one IP hop
  - Similar to the operation of switches (LAN) within the IP routing domain in Ethernet
- Allows IP routing over a mesh topology 802.15.4
- Impact of the cost of power consumption in 6LoWPAN

# Comparison between 6LoWPAN, WiFi Tag and Zigbee



# The 6LowPAN standard

Network Working Group  
Request for Comments: 4944  
Category: Standards Track

G. Montenegro  
Microsoft Corporation  
N. Kushalnagar  
Intel Corp  
J. Hui  
D. Culler  
Arch Rock Corp  
September 2007

 I E T F®

Transmission of IPv6 Packets over IEEE 802.15.4 Networks



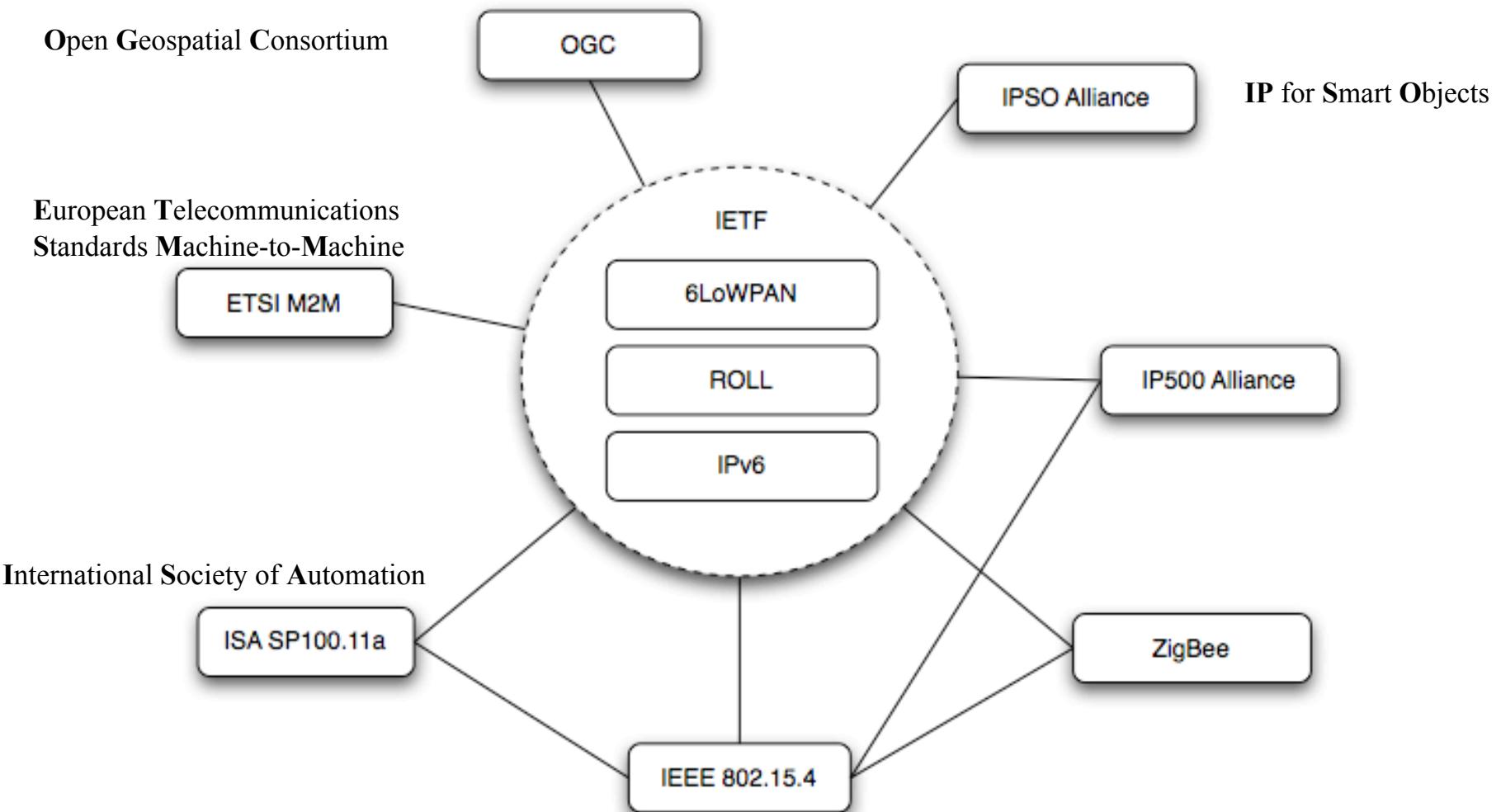
Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

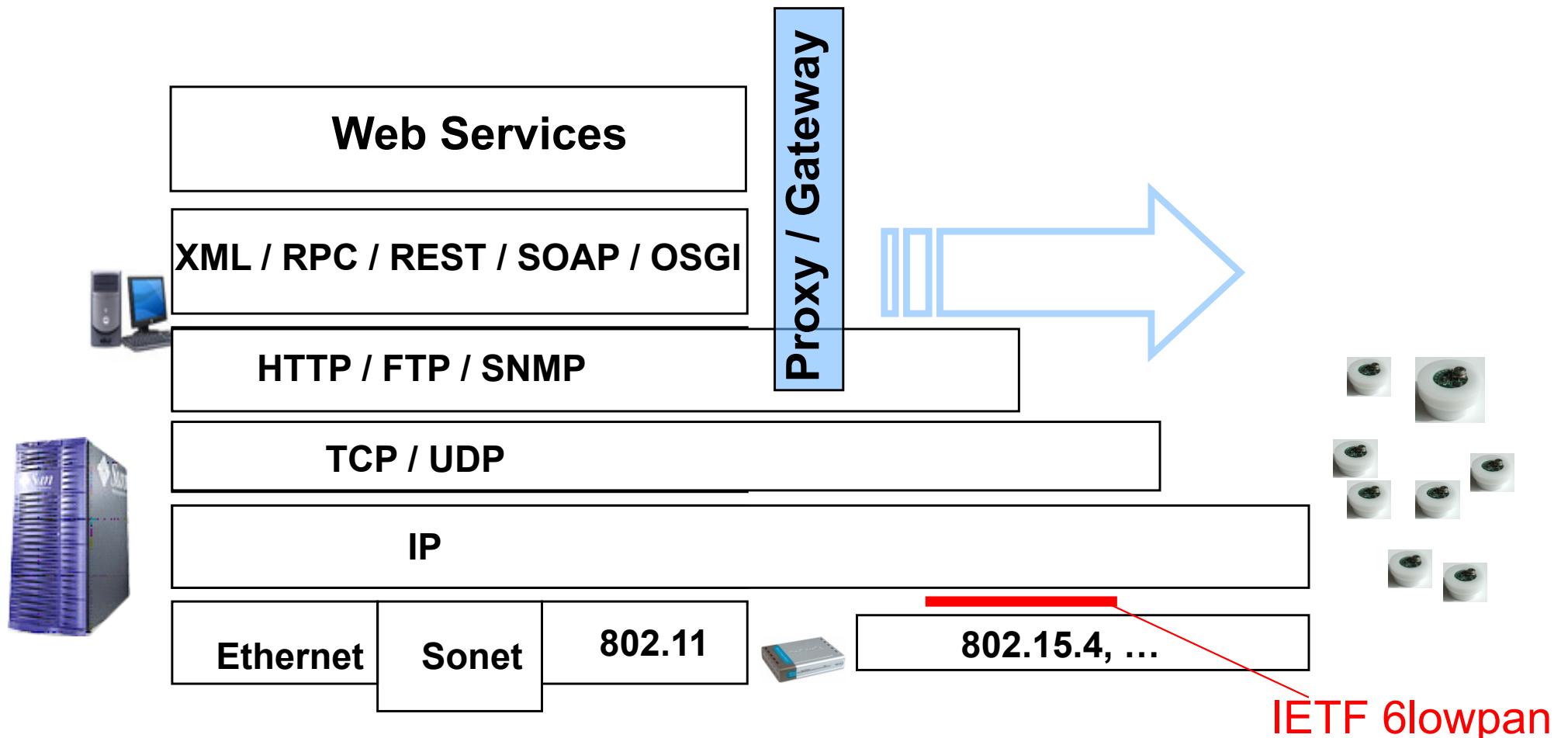
Abstract

This document describes the frame format for transmission of IPv6 packets and the method of forming IPv6 link-local addresses and statelessly autoconfigured addresses on IEEE 802.15.4 networks. Additional specifications include a simple header compression scheme using shared context and provisions for packet delivery in IEEE 802.15.4 meshes.

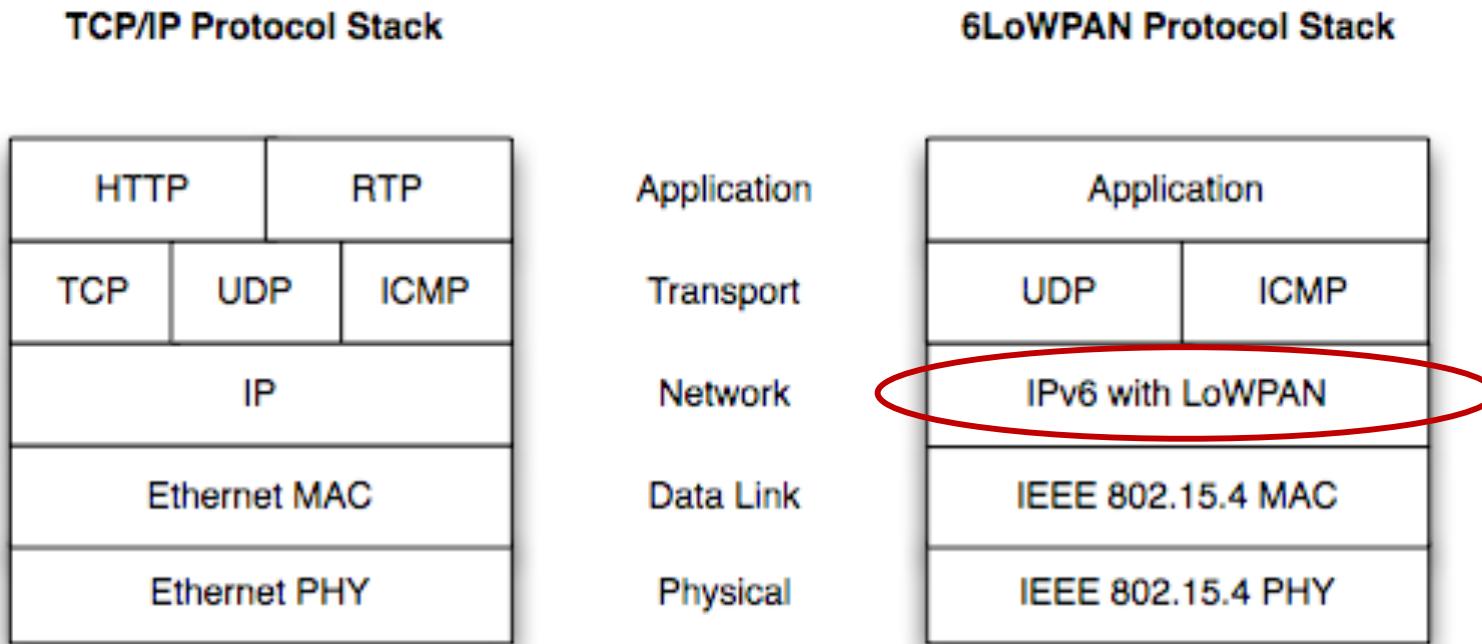
# Relationship between different standards



# Positioning in the protocol stack



# Positioning in the protocol stack



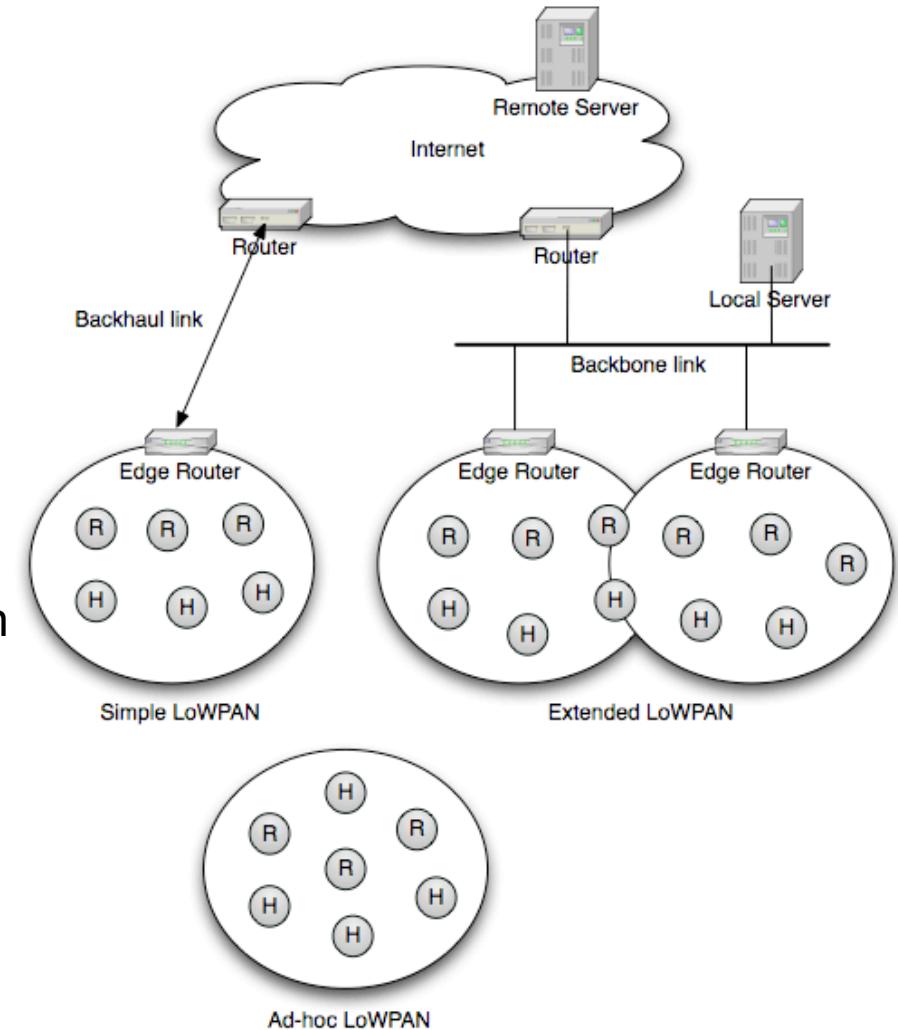
# The characteristics

- Low energy consumption
- Possible network topology: star and mesh
- Large number of devices => need for IPv6 to allocate addresses
- Efficient packet header compression
  - Based on IPv6 and extension headers, UDP header
- Auto-configuration of the network (neighborhood discovery mechanism)
- Supports transmission modes: Unicast, multicast and broadcast
- Fragmentation
  - 1280 byte IPv6 MTU -> 127 byte frames 802.15.4
- Supports IP routing (e.g. IETF RPL)
- Supports the use of the mesh bond layer (e.g. 802.15.5)

# Architecture

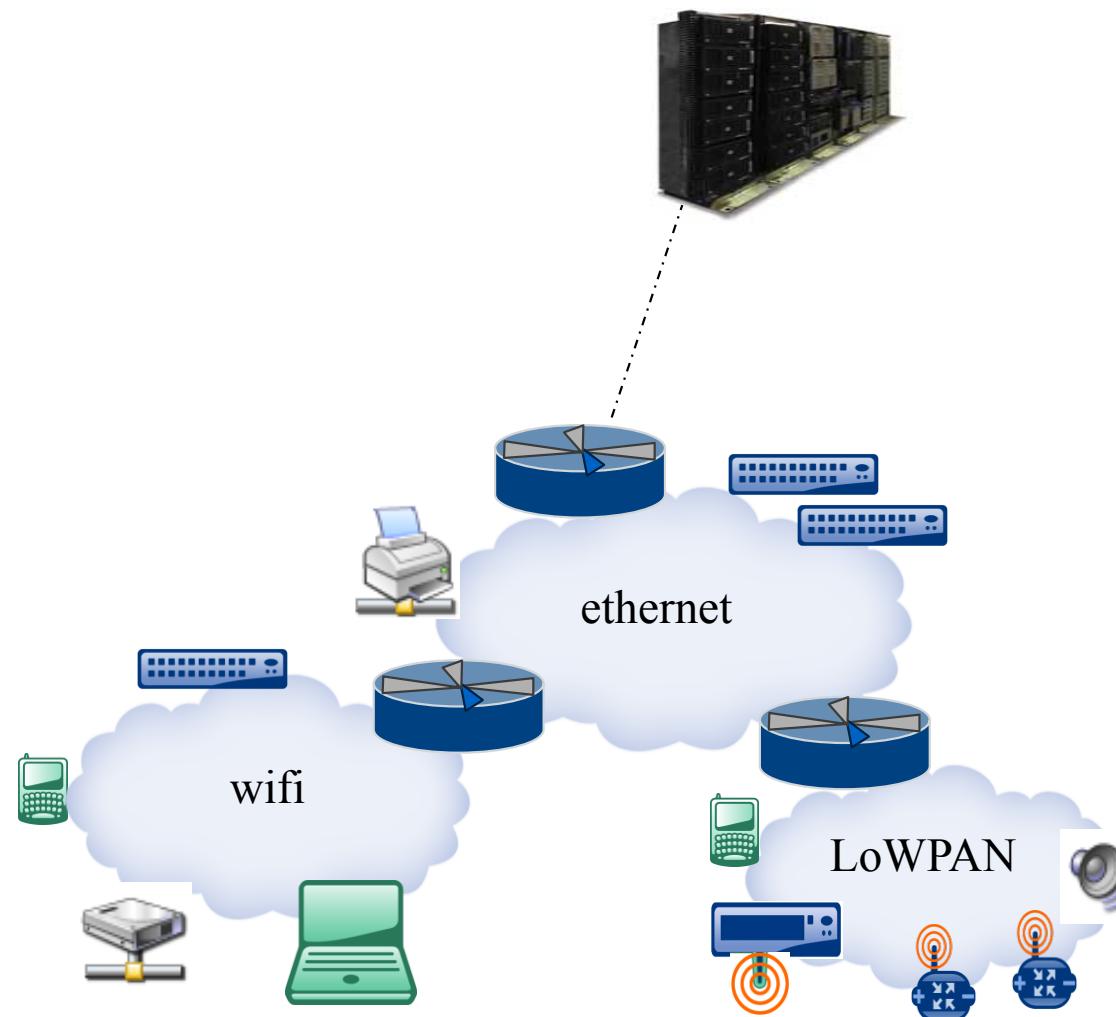
LoWPAN can be deployed in different formats:

- **Simple LoWPAN**
  - A single edge router
  
- **Extended LoWPAN**
  - Multiple edge routers with a common backbone link
  
- **Ad-hoc LoWPAN**
  - No road outside LoWPAN



# Architecture

Example of WiFi, Ethernet and LoWPAN coexistence

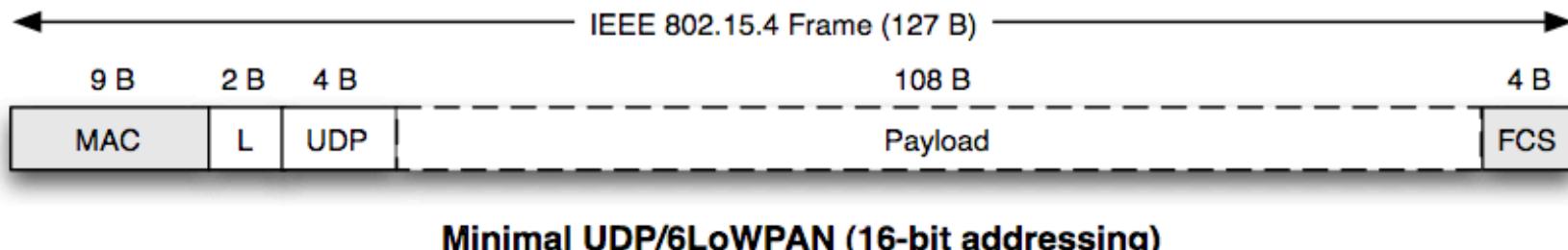
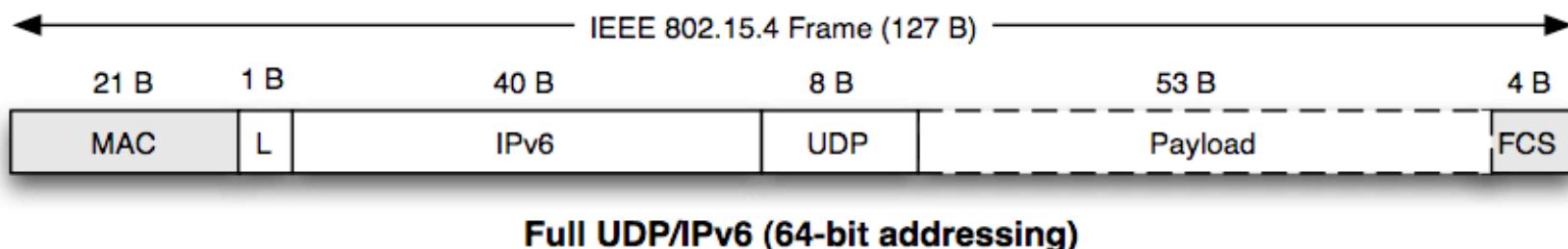


# Architecture

- There are three types of communication:
  - many-to-one (to the root/sink) ;
  - one-to many (from the root/sink) ;
  - point-to point (successively to and from the root/sink).
- The problems of 6LowPAN:
  - Maximum Transmission Unit
  - Application protocols
  - IPv4 interconnectivity
  - Firewalls and NATs
  - Security

# The 6LowPAN header

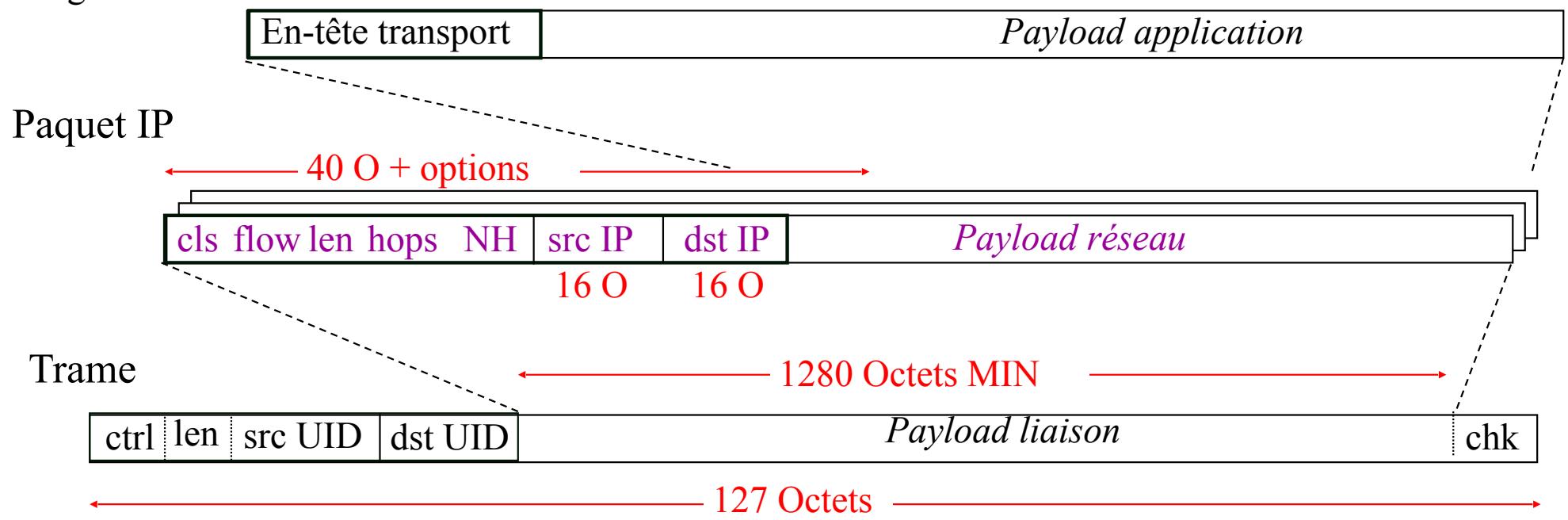
- For better efficiency, the header format is orthogonal.
- Header compression



# Challenges of 6LowPAN

Datagramme UDP ou  
Segment TCP

..., modbus, BacNET/IP, ... , HTML, XML, ..., ZCL



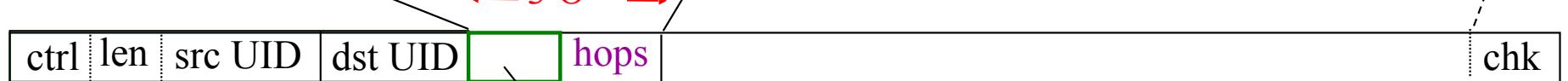
- IP address size Address & large header => 16 bit short address / 64 bit EUID
- Minimum transfer unit => Fragmentation
- Short range radio & Embedded => Multi-hops

# IP header optimization

IP Packet

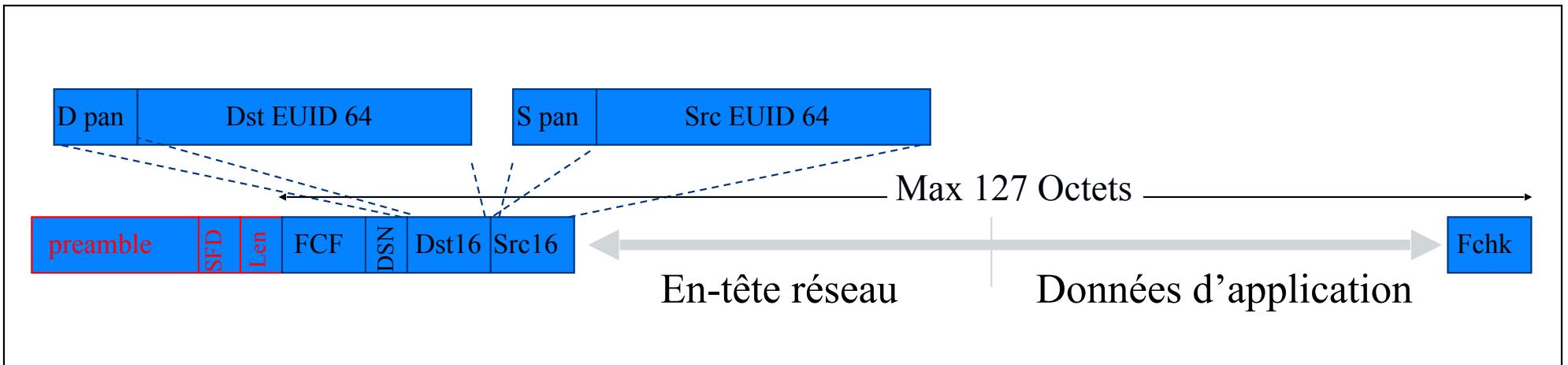


Frame



- Remove all fields from the IPv6 header that can be derived from the 802.15.4 header.
  - Source address : deduced from the link address
  - Destination address : deduced from the link address
  - Size : deduced from the frame size at link level
  - Traffic Class & Flow Label : zero
  - Next header : UDP, TCP, ou ICMP
- The additional IPv6 options are defined as options

# Frame format 802.15.4



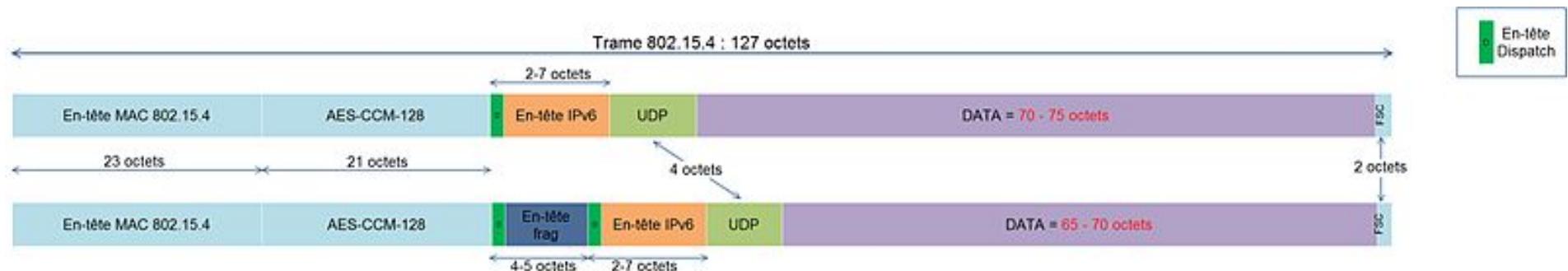
- A simple MAC layer allows to use
  - Several protocols based on TinyOS (MintRoute, LQI, ...), TinyAODV, Zigbee, SP100.11, Wireless HART, etc.
  - 6LoWPAN => IP
- The packet size is small for
  - Keeping the error rate low
  - Allow to share the channel

# Header compression (HC)

- Two IPv6 headers compression mechanisms for LowPAN:
  - **LOWPAN\_HC1**: It also includes 4-byte UDP header compression, but does not allow checksum compression.
    - This method restricts the range of **UDP ports** from 61616 to 61631 in order to compress this value to **4 bits**.
    - This IPv6 header compression can only be applied to **local link addresses**.
  - **LOWPAN\_HC1g**: It applies to global addresses for multi-hop IP communications.
- These two compression mechanisms are **complementary**
- The 6LoWPAN working group proposes to use **LOWPAN\_IPHC**.
- **LOWPAN\_IPHC** allows to replace LOWPAN\_HC1 and LOWPAN\_HC1g.
- The IPHC bytes result from the compression of the IPv6 header.
- They mainly integrate the quality of service information.
  - (DSCP and ECN) of the next headers, the number of jumps and the compressed source/destination addresses.

# Header compression (HC)

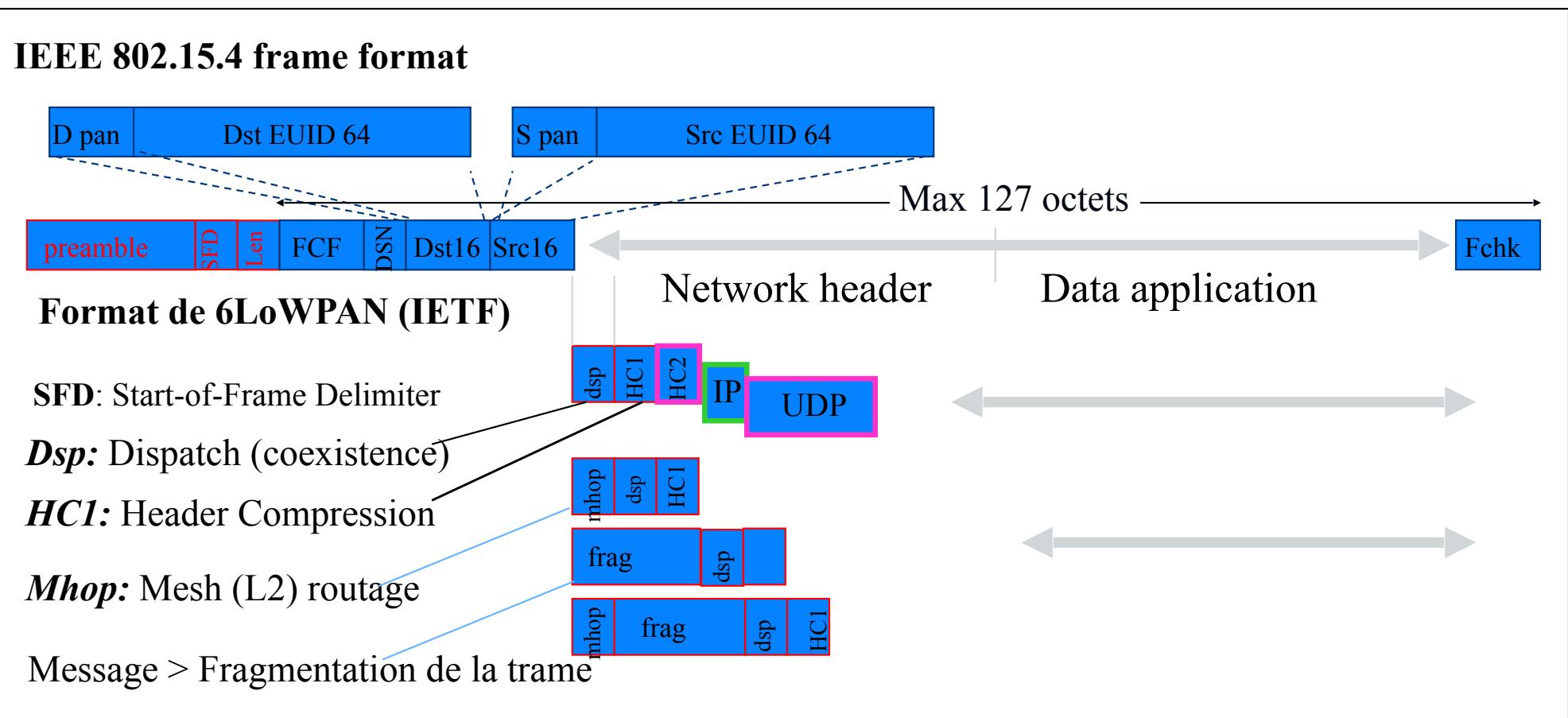
- The compression rate of the LWPAN\_IPHC method depends on the type of communication :
  - For communications on a local link, the IPv6 header can be reduced to **2 bytes**.
    - 1-byte Dispatch and 1-byte LWPAN\_IPHC
  - For multi-hop IP communications the header can be compressed to **7 bytes**.
    - 1-byte Dispatch, 1-byte LWPAN\_IPHC, 1-byte Hop Limit, 2-byte Source Address and 2-byte Destination Address



- The payload is:
  - 70 to 75 bytes in the best case.
  - 65-70 bytes in the case of adding fragmentation information

# 6LoWPAN header design

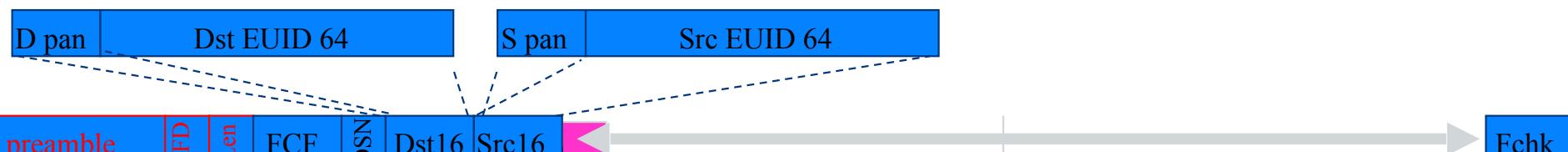
- Header format must be stackable and orthogonal
- Virtually no overload for interoperability and scaling



# 6LoWPAN – The first byte

- The "Dispatch" is used to determine the type of packet above 802.15.4.
- Coexistence with other protocols on the same link

Format de la trame IEEE 802.15.4



Frame Format 6LoWPAN

*Examples of possible values for the Dispatch byte*

Valeur	signification
01000001	paquet IPv6 non compressé
01010000	Broadcast LoWPAN
11000***	premier fragment
11100***	fragment
11110CPP	UDP Header

00 Not a LoWPAN frame

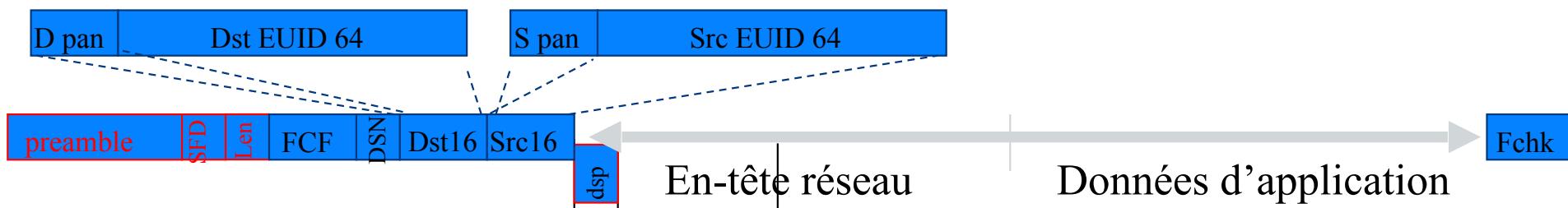
01 LoWPAN IPv6 addressing header

10 LoWPAN mesh header

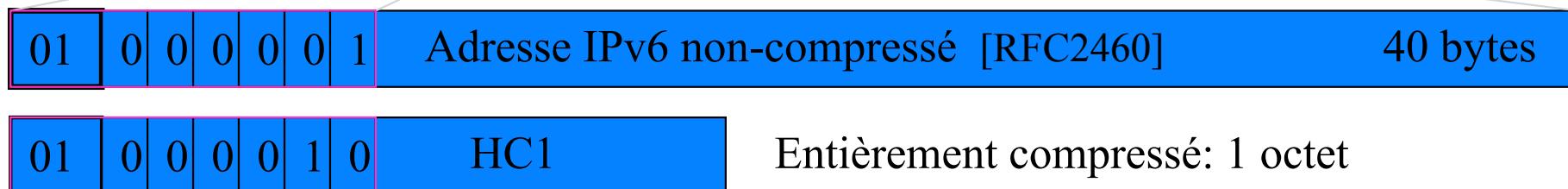
11 LoWPAN fragmentation header

# 6LoWPAN vs IPv6

## Format de la trame IEEE 802.15.4



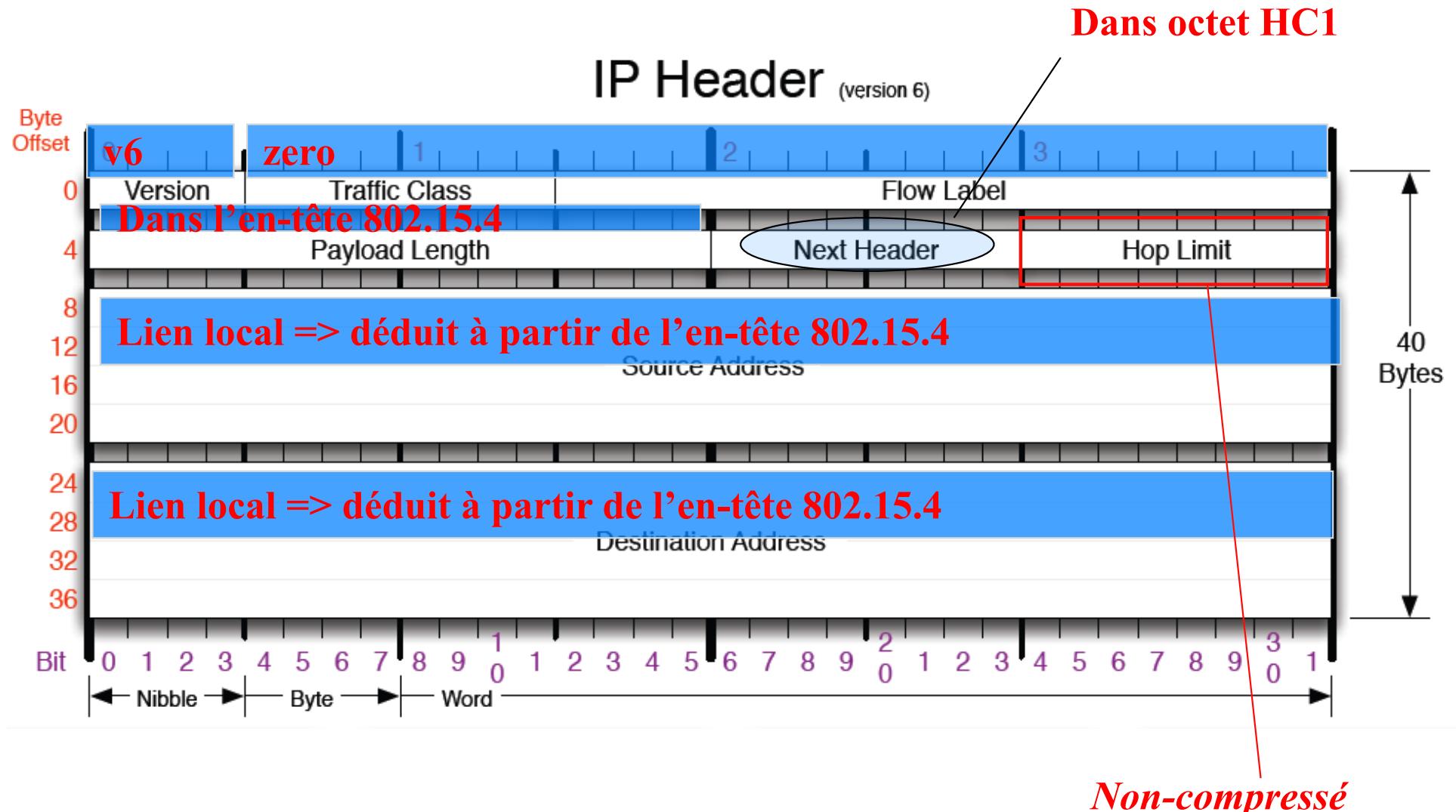
## Format de la trame 6LoWPAN



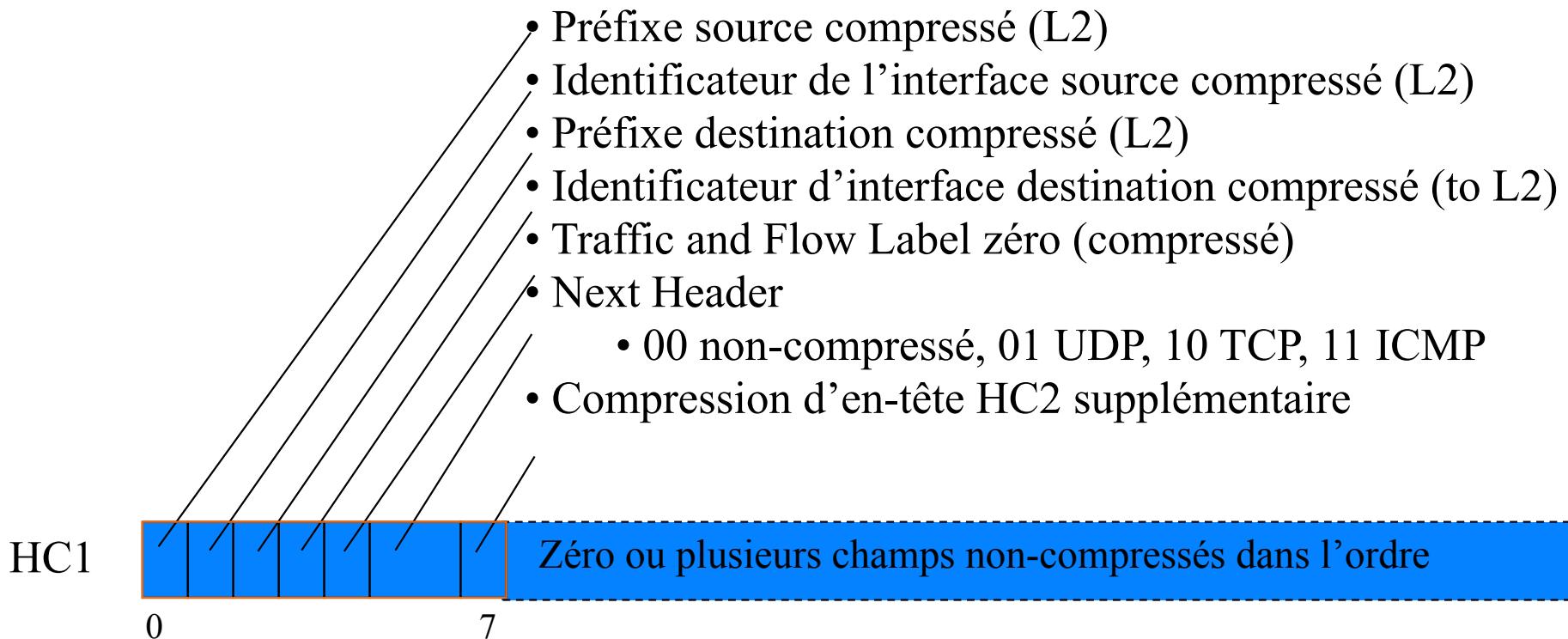
Adresse Source  
 Adresse Destination  
 Traffic Class & Flow Label  
 Next header

- : déduit à partir de l'adresse du niveau 2
- : déduit à partir de l'adresse du niveau 2
- : zéro
- : UDP, TCP, or ICMP

# Compression d'en-tête IPv6



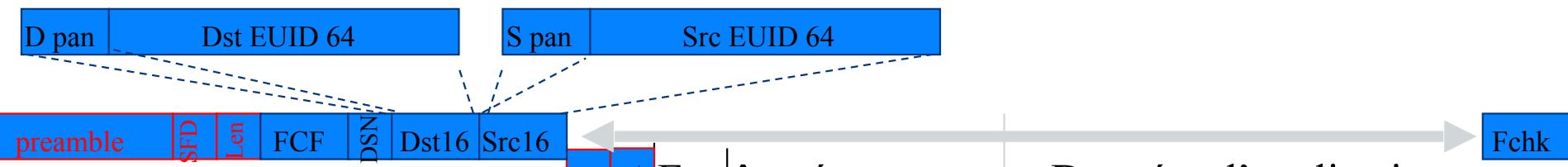
# Niveau de compression HC1



- Adresse IPv6 <prefix64 || interface id> pour les noeuds dans un sous-réseau 802.15.4 est déduite de l'adresse au niveau 2
  - Chaque PAN ID avec un préfixe IPv6 unique
  - Interface identifier généré à partir de EUID64 ou Pan ID et short address
- Hop Limit est le seul champ incompressible dans l'en-tête IPv6

# 6LoWPAN: En-tête IPv6 compressée

## Format de la trame IEEE 802.15.4



## Format de la trame 6LoWPAN

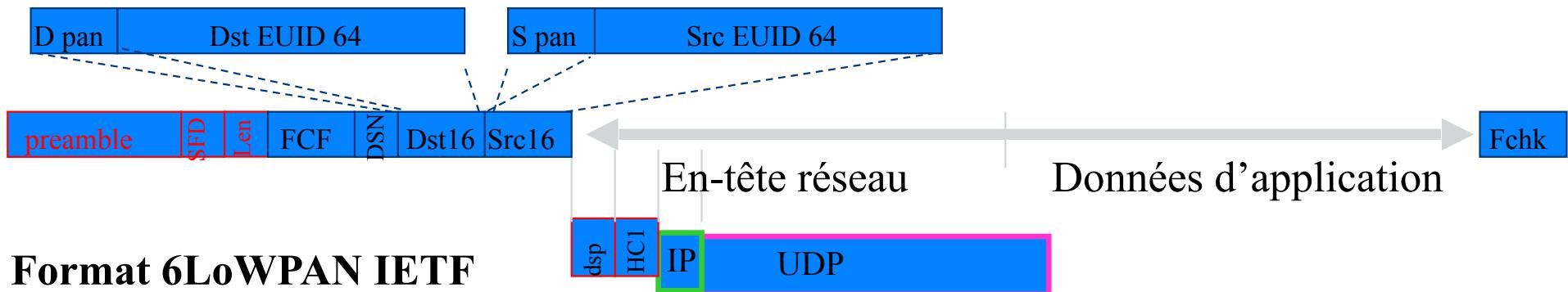


*Comment est-il compressé,*  
“IPv6 Compressé”

- Pas d'adresse locale 802.15.4
- non-zero traffic & flow
  - rare et optionnel

# 6LoWPAN – Compressé / UDP

## Format de la trame IEEE 802.15.4



Dispatch: IPv6 compressé

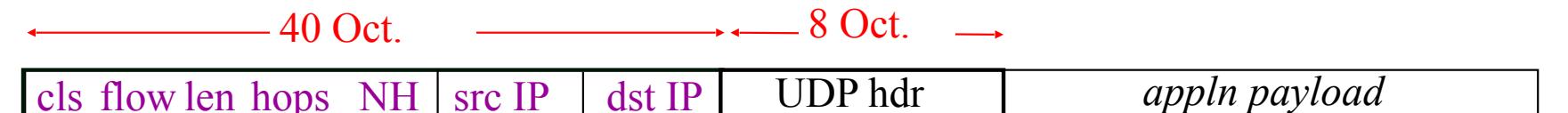
HC1: Source & Dest Local, next hdr=UDP

IP: Hop limit

UDP: 8-byte header (incompressé)

# 6LoWPAN – Optimisation UDP/IP

Paquet au niveau 3



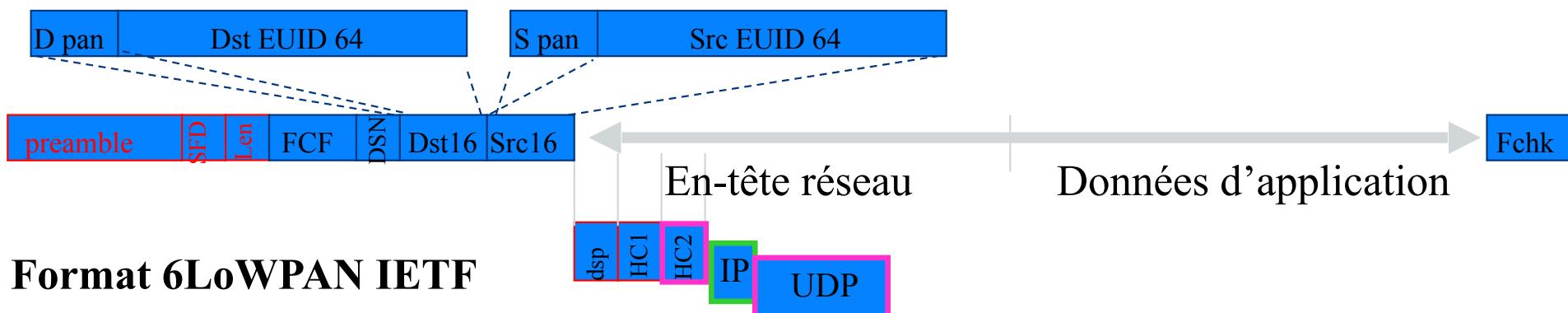
Trame au niveau liaison



- La taille du niveau transport est dérivée du niveau liaison
- Les numéros de ports sont compressés

# 6LoWPAN – Compressé / Compressé UDP

## Format de la trame IEEE 802.15.4



Dispatch: IPv6 compressé

HC1: Source & Dest Local, next hdr=UDP

IP: Hop limit

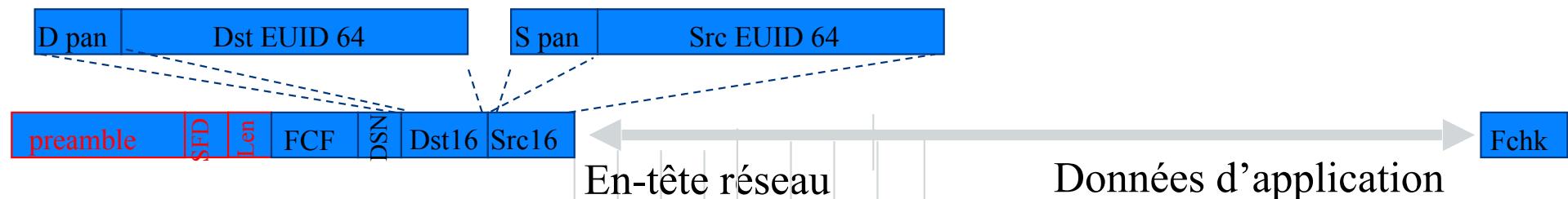
UDP: En-tête HC2+3-byte (compressé)

Port source = P + 4 bits, p = 61616 (0xF0B0)

Port destination = P + 4 bits

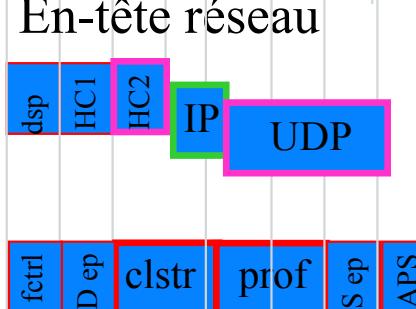
# Comparaison 6LoWPAN / Zigbee

## Format de la trame IEEE 802.15.4



## Format 6LoWPAN

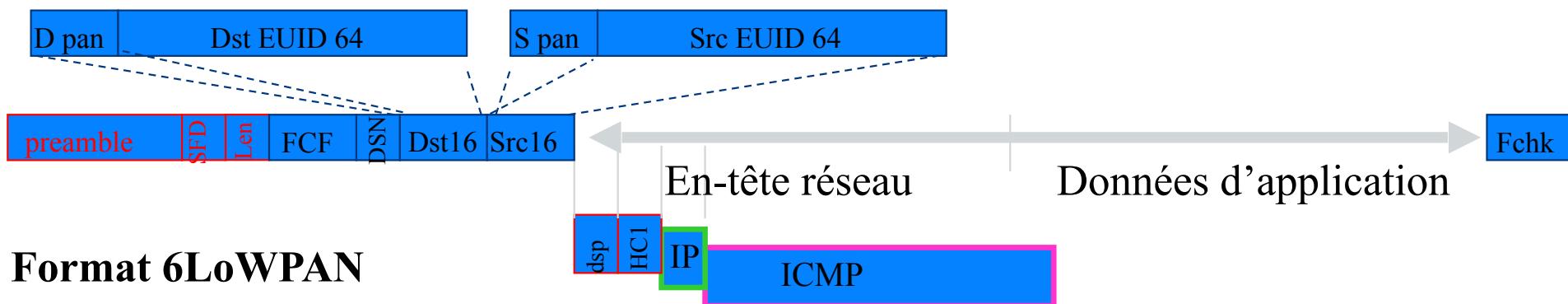
## Format de la trame Zigbee APDU (APDU, Application Protocol Data Unit)



- fctrl: Bits de contrôle de la trame
- D ep: Destination Endpoint (comme le port UDP)
- clstr: Identificateur du cluster
- prof: profil identifier
- S ep: Source Endpoint
- APS: APS counter (sequence to prevent duplicates)

# 6LoWPAN – Compressé / ICMP

## Format de la trame IEEE 802.15.4



## Format 6LoWPAN

Dispatch: IPv6 compressé

HC1: Source & Dest Local, next hdr=ICMP

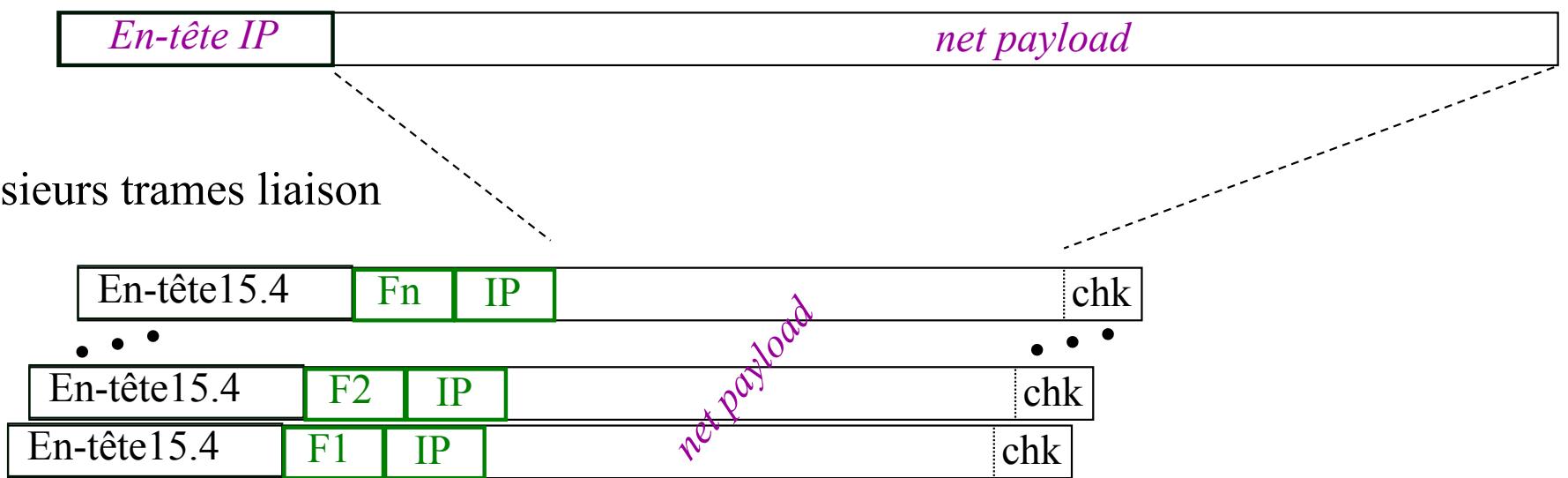
IP: Hops Limit

ICMP: En-tête 8-octets

# Fragmentation 6LoWPAN

- Interopérabilité IP => l'utilisateur n'est pas limité par la taille de la trame
- Les datagrammes IP sont souvent de grande taille comparés à la trame 802.15.4 ce qui oblige une fragmentation en plusieurs trames

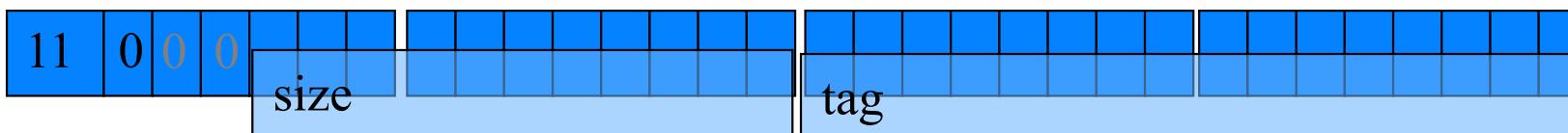
Paquet au niveau 3



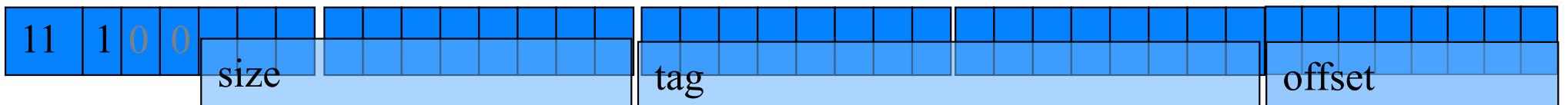
# Fragmentation

- Tous les fragments du même paquet IP possède le même “tag”
  - Assigné séquentiellement par la source de la fragmentation
- Chaque fragment spécifie : tag, taille, et position
- Pas besoin de respecter l'ordre des fragments à l'arrivée
- La limite du temps pour l'ensemble des fragments est 60s

Premier fragment

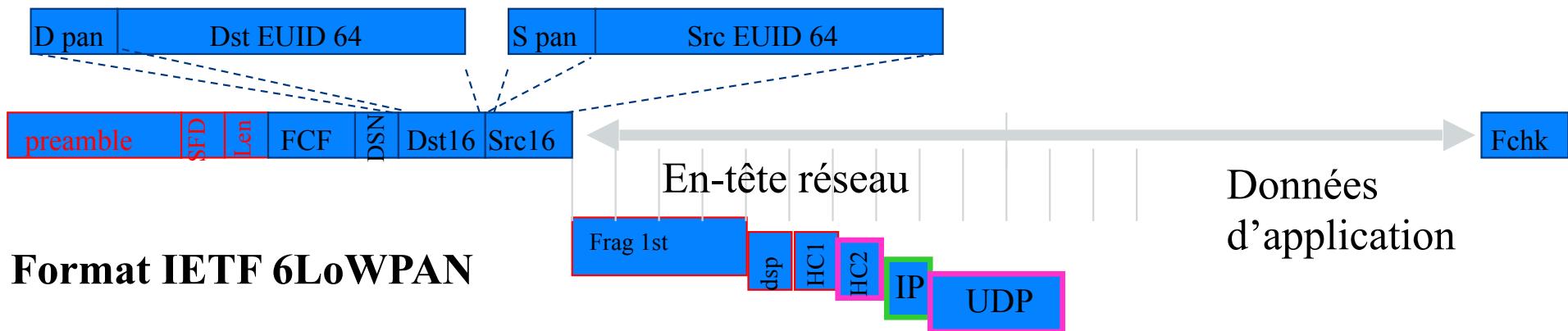


Le reste des fragments



# Exemple : Format 6LoWPAN Fragmenté / Compressé / UDP Compressé

## Format de la trame IEEE 802.15.4



## Format IETF 6LoWPAN

Dispatch: Fragmenté, Premier Fragment, Tag, taille

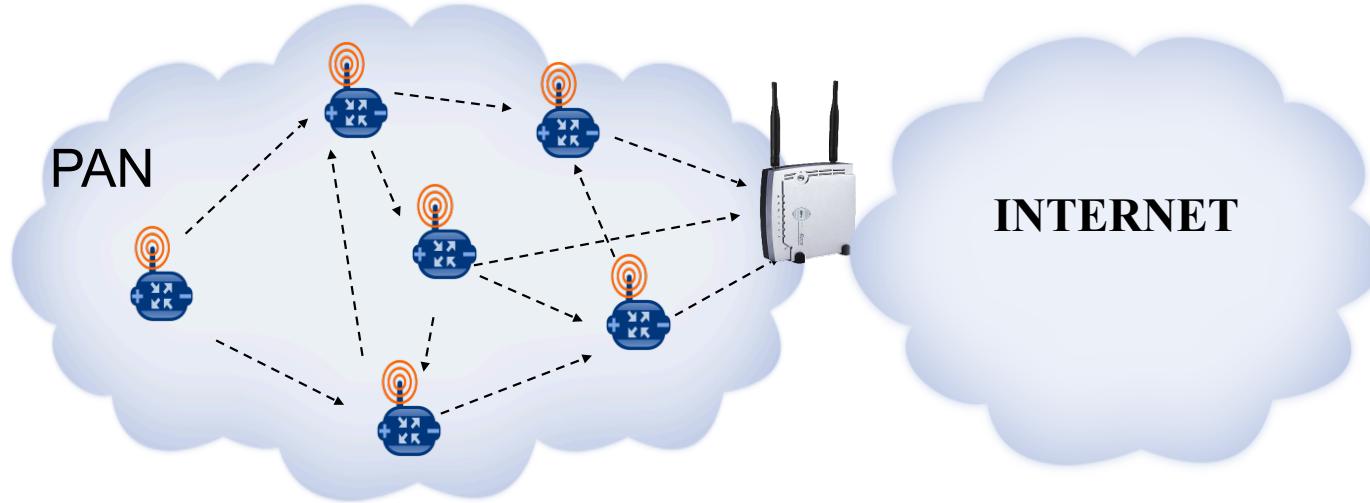
Dispatch: IPv6 Compressé

HC1: Source & Dest Local, next hdr=UDP

IP: Hop limit

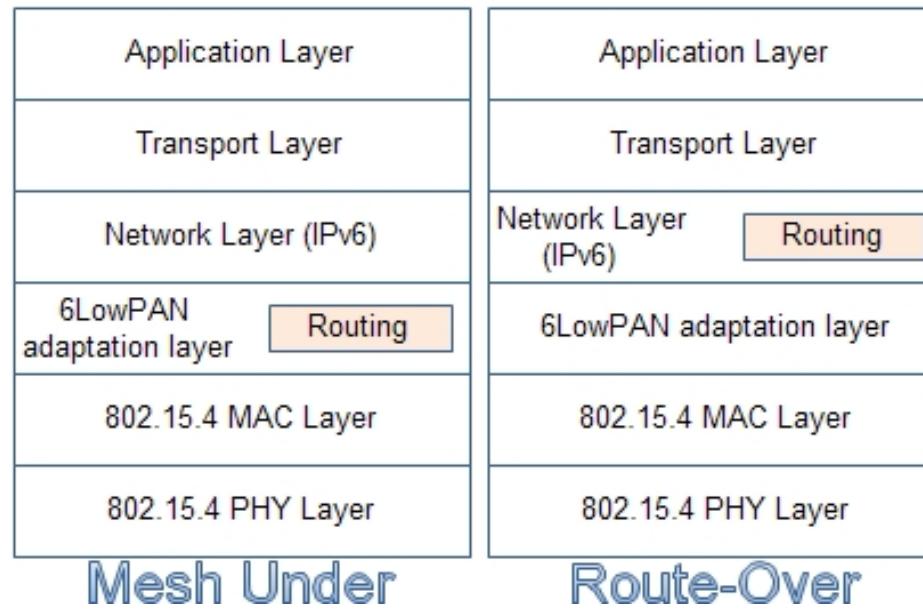
UDP: En-tête HC2+3-Octets (compressé)

# Communication à plusieurs sauts



- Portée radio petite et Obstacles => La communication à plusieurs sauts est souvent nécessaire
  - i.e. Routage et relierage (Forwarding)
- “**Mesh-under**”: Communication à plusieurs sauts au niveau de la couche liaison
  - Mais il a besoin d'un autre type de routage dans le cas **de plusieurs PANs**
- “**Route-over**”: Routage au niveau IP avec le PAN
- 6LoWPAN supporte les deux types de routage

# Routage : Mesh Under vs Route-Over



# Communication à plusieurs sauts basée IP

- IP a toujours fonctionné en “**multi-hop**”
  - Les routeurs connectent les sous-réseaux avec d’autres réseaux
  - Les sous-réseaux peuvent avoir les mêmes ou différents liens physiques
- Les routeurs utilisent des **tables de routage** pour déterminer quel est le prochain saut “next hop” vers la destination
- Les protocoles de routage permettent d’établir et de maintenir les tables de routage
  - Plusieurs protocoles de routage sont utilisés dans différentes situations
  - RIP, OSPF, IGP, BGP, AODV, OLSR, ...
- Le routage IP sur les liens 15.4 **n'a pas besoin des en-têtes supplémentaires au niveau 6LoWPAN**
- Plusieurs outils et logiciels supportent le routage IP
  - Diagnostic, traçage, gestion, visualisation
  - En revanche, ces outils doivent être **réinventés pour le routage maillé « meshing »**

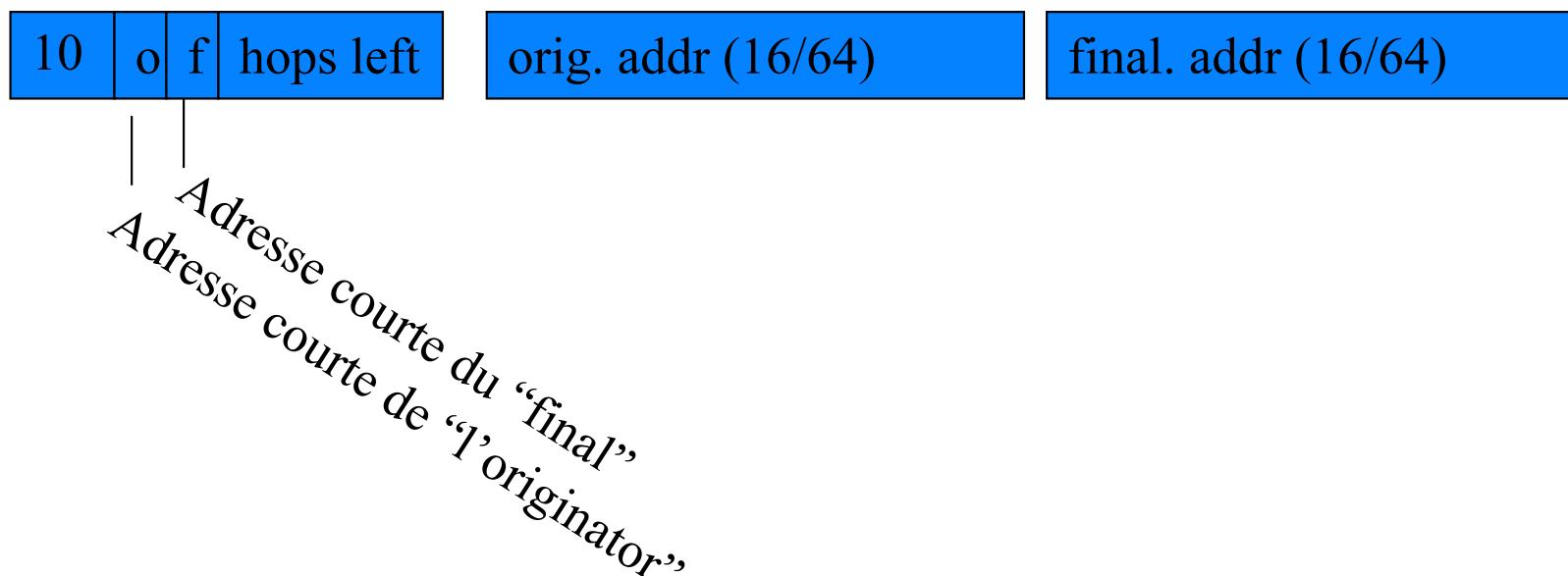
# Routage : IP vs Maillé (meshing)

- Le lien IP conventionnel est **un domaine de diffusion**
  - Routage connecte les liens (ex. : réseaux)
- De nombreux liens IP sont passés du domaine de diffusion à celui de couche liaison « maillé- mesh » avec une émulation de diffusion
  - Ethernet => Switched Ethernet
  - 802.11 => 802.11s (WLAN maillé)
- Le routage IP utilise une grande bande passante **sans la contrainte d'énergie** pour maintenir les liens/routes
- Les réseaux 802.15.4 **sont limités** en bande passante et en énergie
- Le routage à deux niveaux différents peut entraîner des conflits
- Un groupe de travail d'IETF propose le protocole **ROLL**
  - Routing Over Low-Power and Lossy networks

# L'en-tête «Mesh Under»

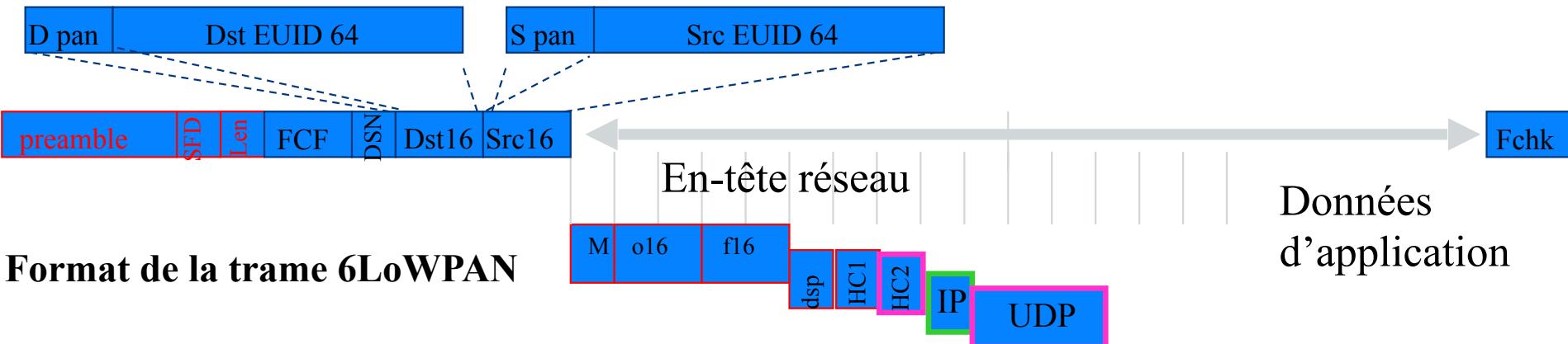
- Les nœuds source et destination sont définis par des adresses .15.4 courtes (16 bits) ou longues EUID (64 bits)
  - En plus des adresses IP source et destination
- Hops Left (jusqu'à 14 sauts, sinon il faut ajouter des Octets)
- Le protocole Mesh détermine le nœud à chaque saut

En-tête mesh LoWPAN mesh



# 6LoWPAN – Exemple Mesh / Compressé / UDP Compressé

## Format de la trame IEEE 802.15.4



## Format de la trame 6LoWPAN

Dispatch: Mesh under, orig short, final short

Mesh: orig addr, final addr

Dispatch: IPv6 compressé

HC1: Source & Dest Local, next hdr=UDP

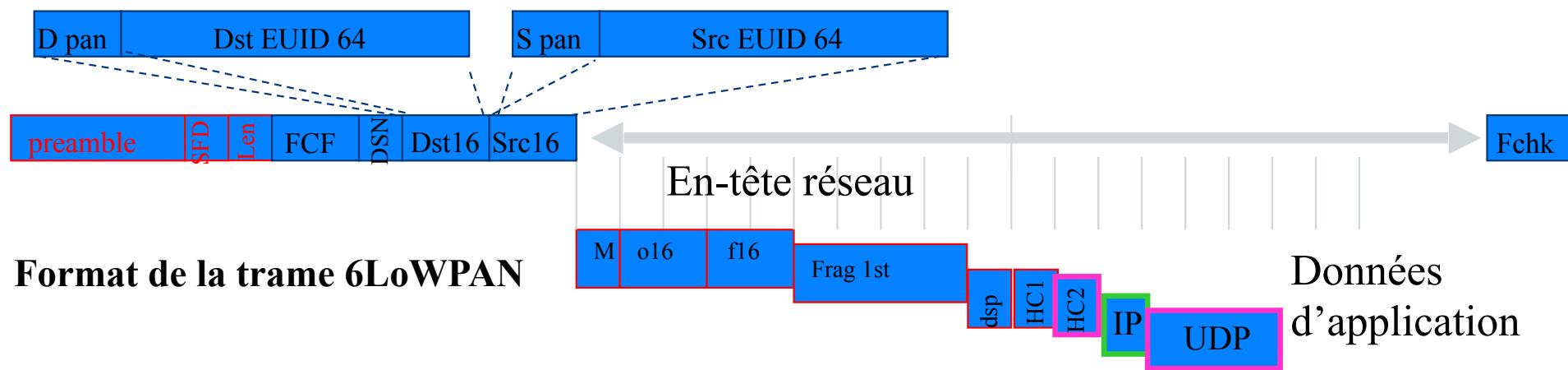
IP: Hop limit

UDP: En-tête HC2+3-Octets

# 6LoWPAN – Exemple

## Mesh / Fragmenté / Compressé / UDP

### Format de la trame IEEE 802.15.4



Dispatch: Mesh under, orig short, final short

Mesh: orig addr, final addr

Dispatch: Fragmenté, Premier Fragment, Tag, Size

Dispatch: IPv6 compressé

HC1: Source & Dest Local, next hdr=UDP

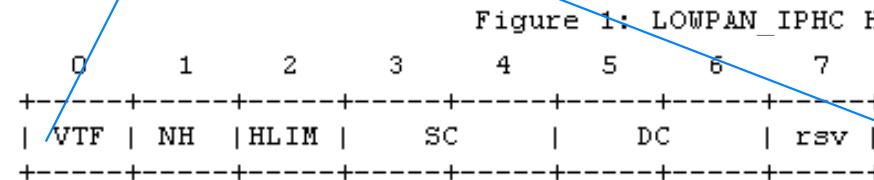
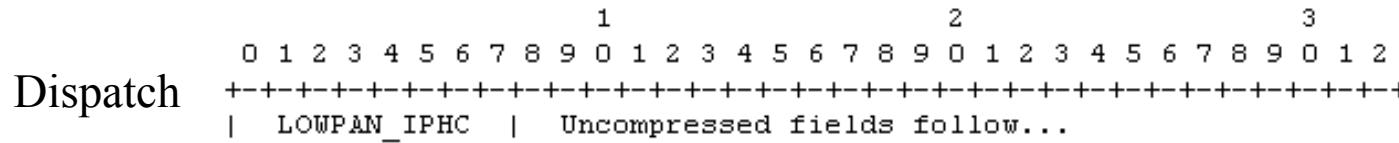
IP: Hop limit

UDP: En-tête HC2 + 3-Oct.

# IP Header Compression (IPHC)

- Ajoute **deux nouveaux** “dispatch” type IPHC – pour compresser l’en-tête IPv6
  - 0x03 (TBD) LOWPAN\_IPHC avec adresse de **lien locale**
  - 0x04 (TBD) LOWPAN\_IPHC avec **préfixe commun routable**
- Toutes les formes de compression d’en-tête au même format
- Mise en place de la prochaine en-tête
- Mise en place de la compression de l’en-tête au niveau (L4)
- Permet la compression d’adresses multicast

# IP Header Compression (IPHC)



- **VTF: Version, Traffic Class, and Flow Label (bit 0):**
  - 0: Full 4 bits for Version, 8 bits for Traffic Class, and 20 bits for Flow Label are carried in-line.
  - 1: Version, Traffic Class, and Flow Label are elided. Version is 6. Traffic Class and Flow Label are 0.
- **NH: Next Hop (bit 1):**
  - 0: Full 8 bits for Next Hop are carried in-line.
  - 1: Next Hop is elided and the next header is compressed using LOWPAN\_NHC
- **HLIM: Hop Limit (bit 2):**
  - 0: All 8 bits of Hop Limit are carried in-line.
  - 1: All 8 bits of Hop Limit are elided.
    - receiving interface => 1, otw 64 (TBD).
- **SRC: Source Address (bits 3 and 4):**
  - 00: All 128 bits of Source Address are carried in-line.
  - 01: 64-bit Compressed IPv6 address.
  - 10: 16-bit Compressed IPv6 address.
  - 11: All 128 bits of Source Address are elided.
- **DST: Destination Address (bits 5 and 6):**

# IPv6 Unicast address header compression

- **SRC/DST** peut être compressé à 64, 16 ou 0 bits
  - 0: dépend entièrement de l'en-tête L2
- Le format de compression à 16 bits est aussi utilisé pour la compression des adresses multicast IPv6
- L'espace d'adresse de 16 bits est divisé en plusieurs portées
- Pour les adresses unicast, le premier bit **porté à la ligne** doit être égal à 0

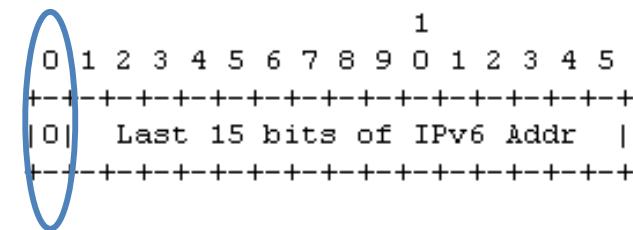


Figure 3: 16-bit Compressed IPv6 Unicast Address Encoding

# Multicast address compression

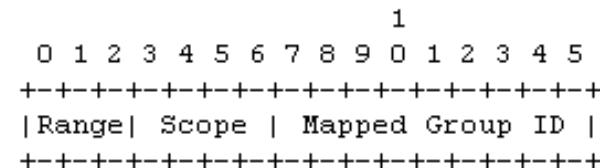


Figure 4: Compressed IPv6 Multicast Address Encoding

- Range (bits 0-2):
  - Fixé à '101' (TBD) – la portée d'adresse courte 6LoWPAN pour les adresses compressées IPv6 multicast
  - 0, 0xxxxxxxxxxxxxx: comme spécifié dans RFC 49441,
  - 1, 101xxxxxxxxxxxxx: les 13 bits restants représentent une adresse multicast IPv6 compressée
  - 2, 100xxxxxxxxxxxxx: comme spécifié dans RFC 4944
- Scope (bits 3-6):
  - portée multicast de 4 bits comme spécifié dans RFC 4007
- Mapped Group ID (bits 7-15)
  - Identifiant de groupe multicast mis au point à 9 bits

# Next Header Compression

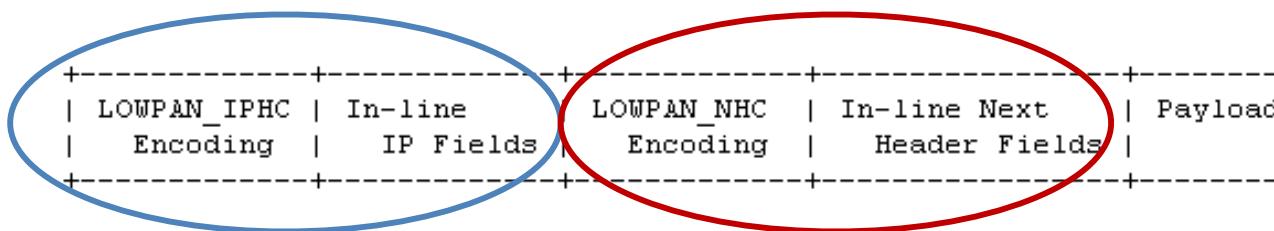


Figure 5: Typical LOWPAN\_IPHC/LOWPAN\_NHC Header Configuration

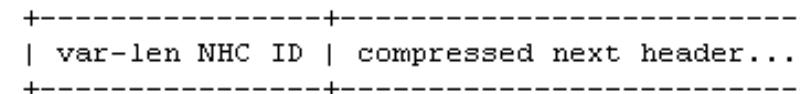
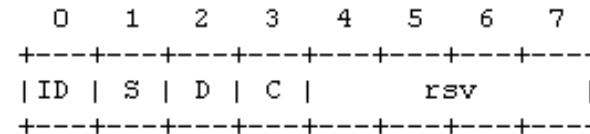


Figure 6: LOWPAN\_NHC Encoding

# Compression de l'en-tête UDP (L4)



- **ID: Identifier (bit 0):**
  - 0: IPv6 Next Header = 17, compressé
  - 1: IPv6 Next Header != 17, non compressé
  
- **S: Source Port (bit 1):**
  - 0: les 16 bits du port source sont portés à la ligne
  - 1: Les premiers 12 bits du port source sont éliminés et les 4 bits restants sont portés à la ligne.
  - Le port source est restauré avec: P + *short\_port*, où P est égal à 61616 (0xF0B0).
  
- **D: Destination Port (bit 2):**
  
- **C: Checksum (bit 3):**
  - 0: les 16 bits de Checksum sont portés à la ligne. Le Checksum DOIT être inclus s'il n'y a aucune autre vérification de l'intégrité de bout en bout qui soit plus efficace que celle fournie par le checksum UDP.
  - 1: les 16 bits du Checksum sont éliminés. Pour restaurer le checksum il faut le recalculer

# Format adaptation

One hop\*

**802.15.4**

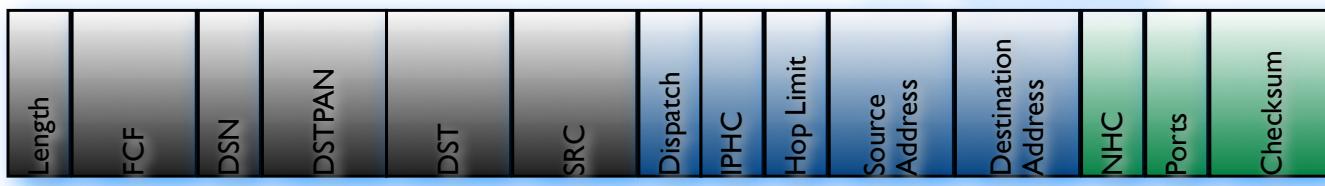
**7 Octets**



- IPv6 Compressed
- UDP Compressed

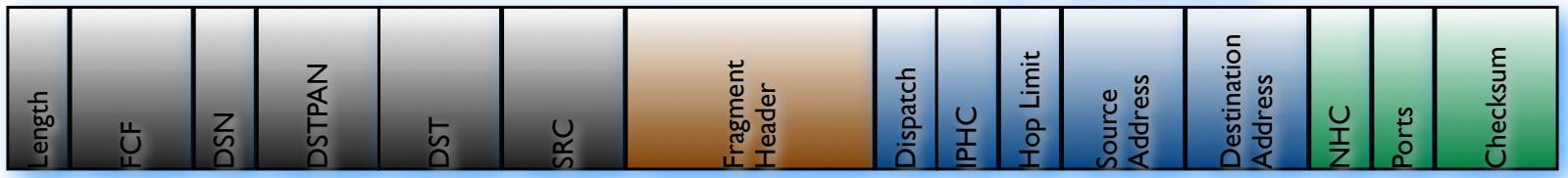
Mesh Hop

**11 Octets**



Mesh Hop Fragmenté

**15 Octets**



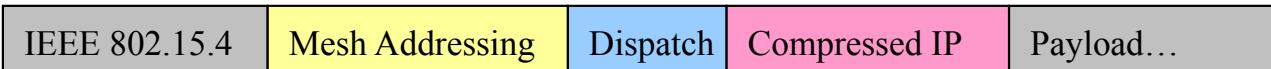
Source and destination  
MAC address

\* including each of the multiple IP hops

Un seul saut , sans Fragmentation



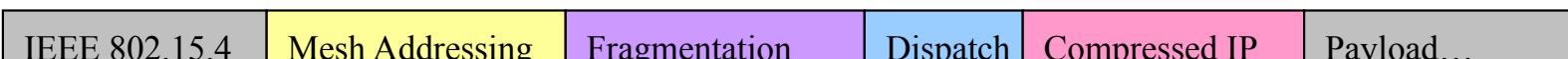
Multi-sauts, sans Fragmentation



Un seul saut, Fragmentation



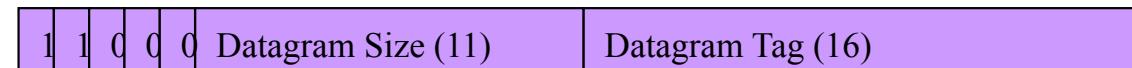
Multi-Sauts, Fragmentation



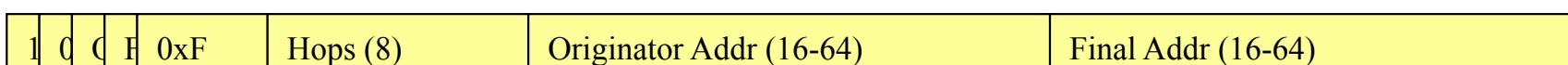
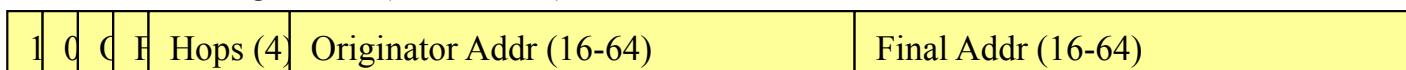
En-tête Dispatch (1-2 Octets)



En-tête de Fragmentation (4-5 Octets)



En-tête d'adressage Mesh (5-18 Octets)

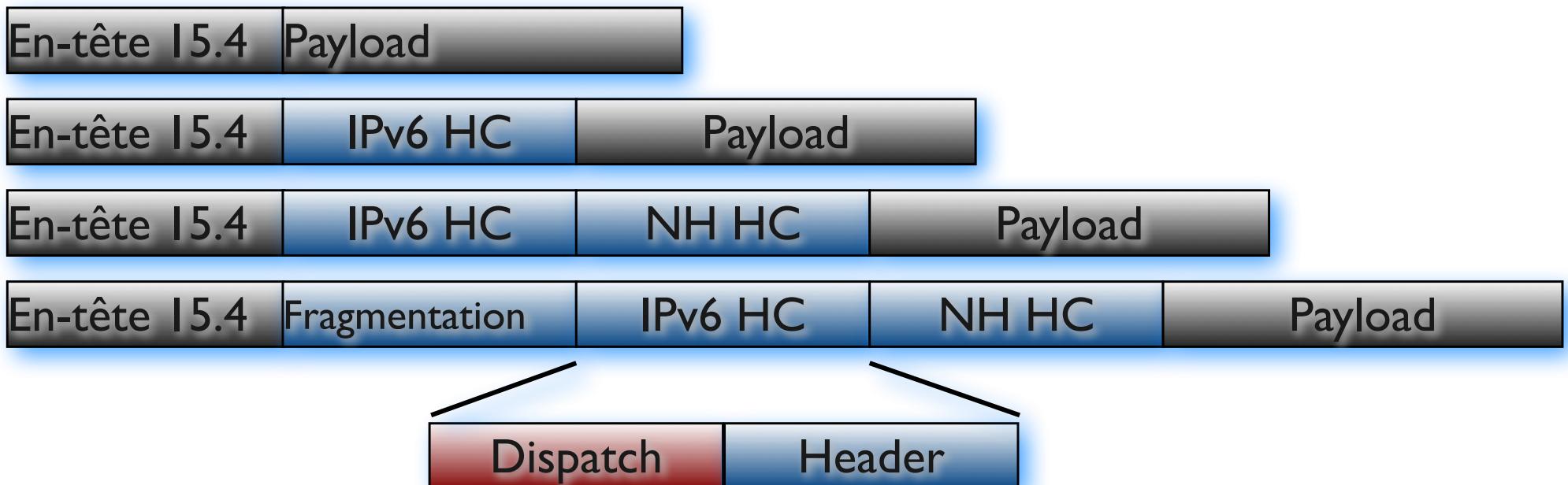


# Format adaptation summary

## IPv6 stackable header format



Adaptation header format 6LowPAN



# IPv6 Address Auto-Configuration

Préfixe de 64-bits

Suffixe de 64-bits ou  
Identificateur d'interface

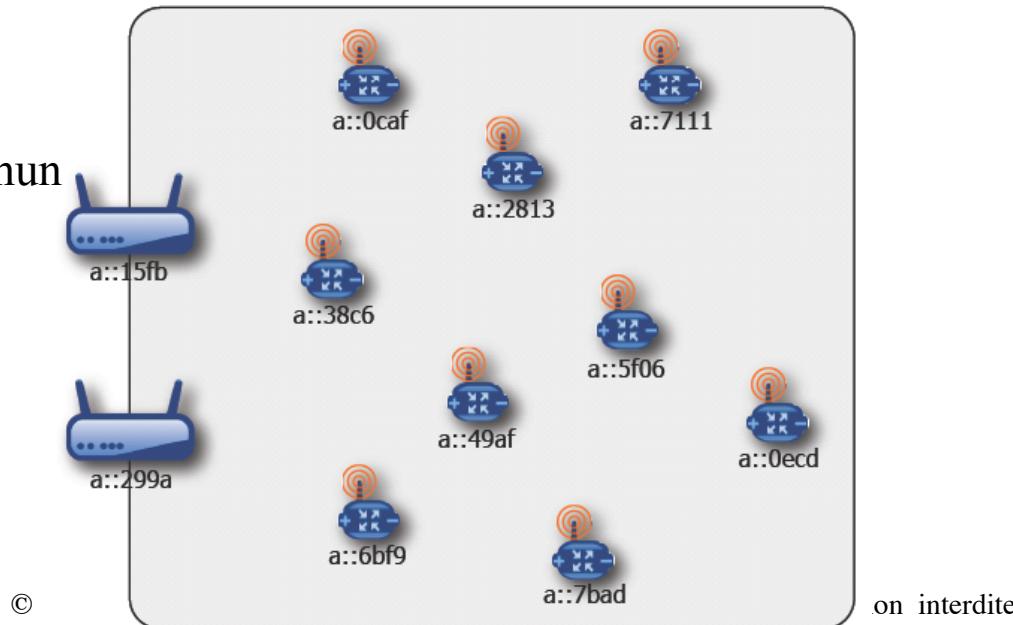


Liaison locale



Adresse  
802.15.4

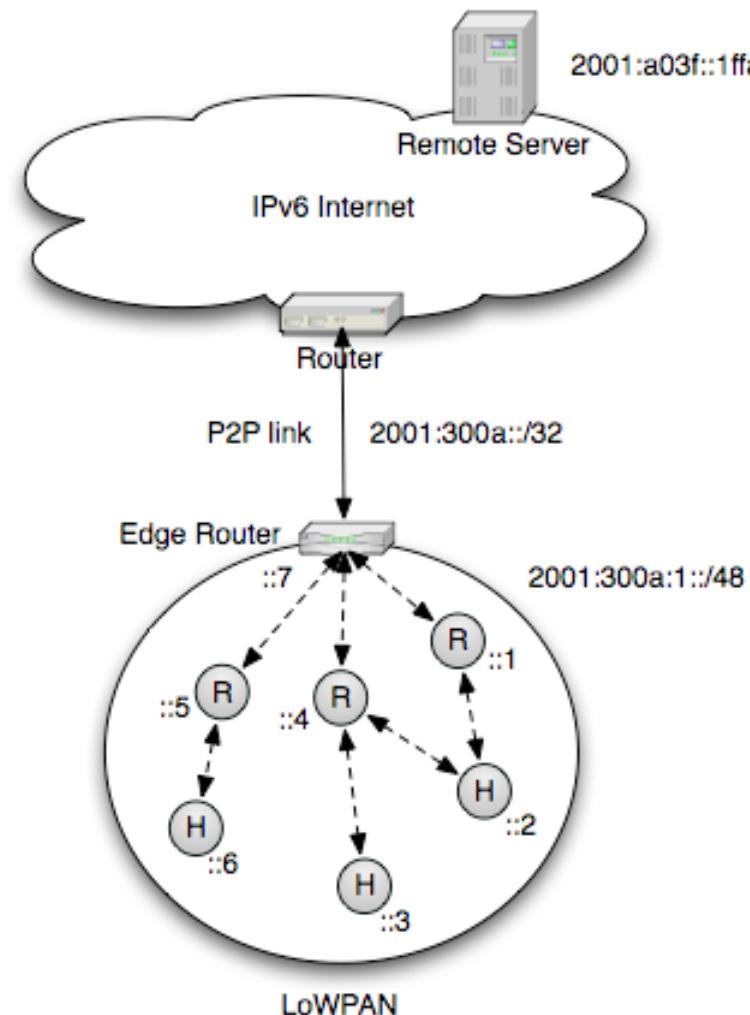
Exemple de préfixe commun  
routable



# Adressage avec 6LoWPAN

- Les adresses IPv6 sont compressées dans 6LoWPAN
- Un LoWPAN fonctionne selon le principe :
  - Des espaces d'adressage plats (le réseau sans fil est un sous-réseau IPv6)
  - Avec des adresses MAC uniques (i.e. 64-bits ou 16-bits)
- 6LoWPAN compresse les adresses IPv6 en :
  - Supprimant le préfixe IPv6
    - **Le préfixe global est connu de tous** les noeuds dans le réseau
    - Le préfixe du lien local est indiqué par le format de compression du header
  - Compression de l'IID
    - Elidée pour une communication du lien local
    - Compression pour des adressages multi-sauts dst/src
  - Compression avec un “contexte” fréquent
  - Les adresses multi-sauts sont compressées

# Exemple d'adressage



# Bootstrapping (Amorçage)

- L'Auto-configuration est importante dans les réseaux embarqués
- Pour que le réseau 6LoWPAN fonctionne :
  1. Connectivité au **niveau liaison** entre les nœuds (commissioning)
  2. Configuration **des adresses réseau**, découverte de voisinage, enregistrements (bootstrapping)
  3. Installation des routes via l'algorithme de routage (route initialization)
  4. Maintenance continue de 1-3

# Connectivity at the link level

- Afin que les nœuds puissent communiquer entre eux, ils doivent avoir une **technologie compatible** aux niveaux physique et MAC
- Exemple technologie MAC IEEE 802.15.4 :
  - Canal, modulation, débit-données
    - Souvent un canal par défaut est utilisé, et les canaux sont scannés pour trouver le routeur avec Neighbor Discovery (ND)
    - Mode d'adressage (64-bits ou 16-bits)
    - 64-bits est le mode par défaut, et 16-bits est utilisé si les adresses sont disponibles
  - Mode de la couche MAC (non-beaconé ou beaconé)
    - Le mode non-beaconé (non-beacon enabled) est simple et sans synchronisation
  - Sécurité (on ou off, clé de chiffrement)

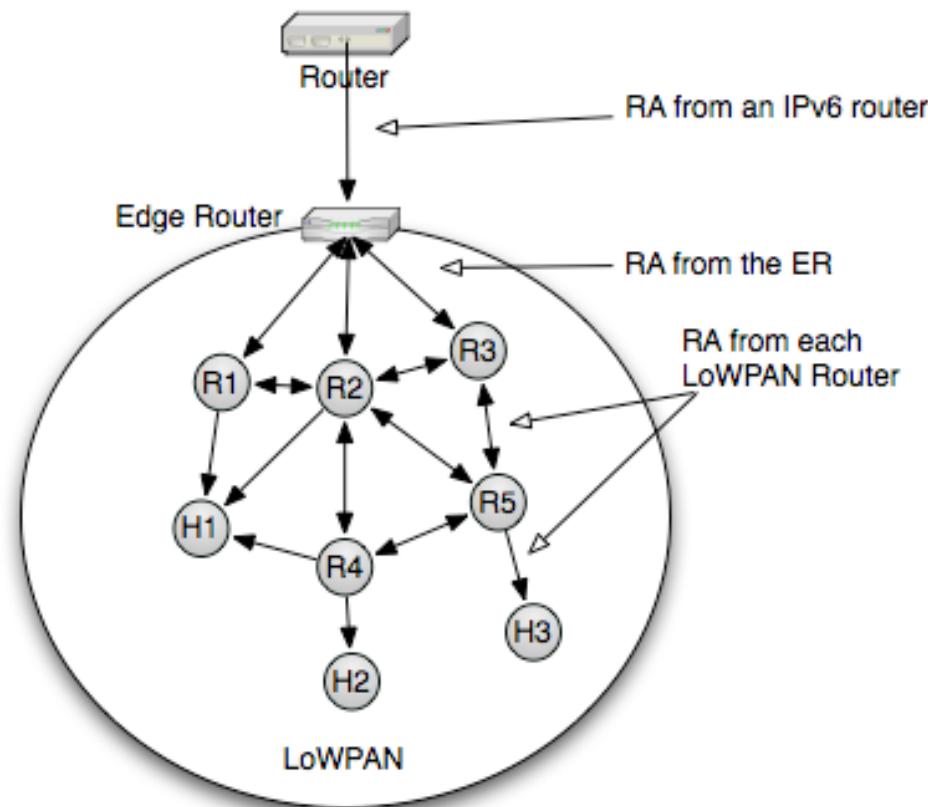
# 6LoWPAN Neighbor Discovery

- Le standard ND d'IPv6 **n'est pas compatible** pour 6LoWPAN :
  - Supposition d'un seul lien pour un préfixe de sous-réseau
  - Supposition que les noeuds sont **toujours activés**
  - Large utilisation du trafic multicast (broadcast/flux dans 6LoWPAN)
  - Pas de support multi-sauts efficace sur 802.15.4
- 6LoWPAN Neighbor Discovery génère :
  - Un lien approprié et un modèle de sous-réseau pour le sans fil à faible énergie
  - Un trafic de contrôle minimal initialisé par les noeuds
  - Enregistrement des Noeuds (NR) et Confirmation des Noeuds (NC)
  - Détection des Adresses de Duplicats (DAD) et restauration
  - Support pour les infrastructures Egde Router étendues
- ND pour 6LoWPAN a été spécifiée dans **draft-ietf-6lowpan-nd**

# Prefix Dissemination

- Dans les réseaux IPv6 traditionnels, les RA sont envoyés à un lien selon les informations ( préfixes, etc.) configurées pour cette interface routeur
- Dans ND pour 6LoWPAN, les RA sont aussi utilisés pour disséminer automatiquement les informations du routeur sur plusieurs sauts

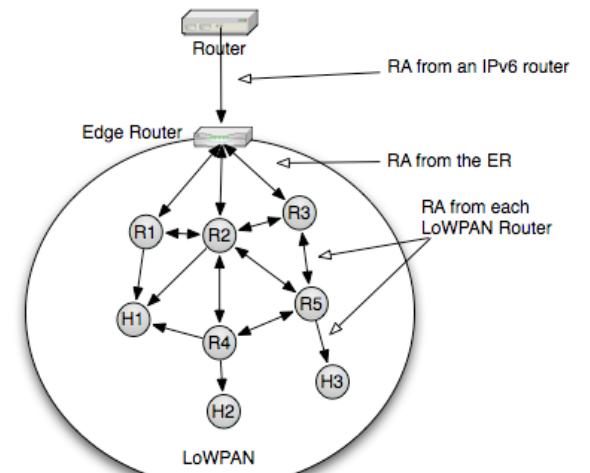
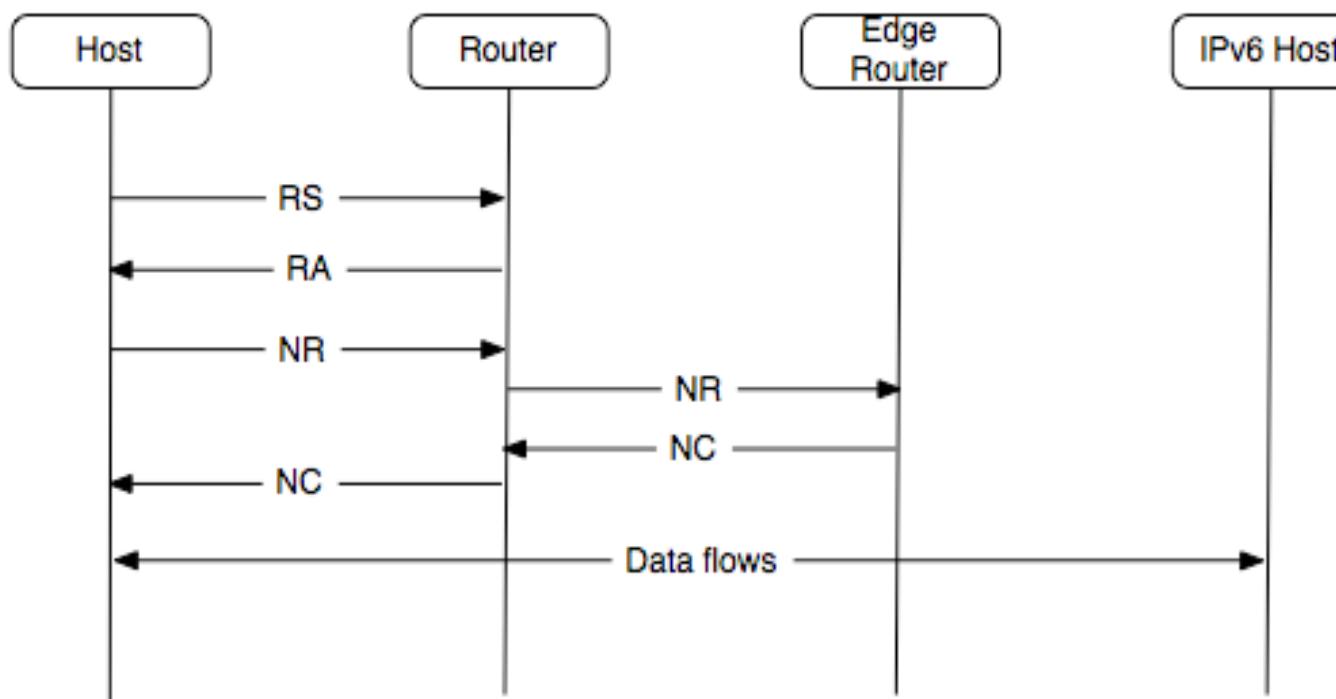
RS : Router Sollicitation  
RA: Router Advertisement



# Nodes registration

- 6LoWPAN-ND **optimise** uniquement l'interface du **routeur-hôte**
  - RFC4861 = signal entre tous les voisins (distribué)
- Les noeuds s'enregistrent avec leurs **routeurs voisins**
  - Echange de messages NR/NC
  - Table des liens entre les noeuds enregistrés conservée par le routeur
- L'échange d'enregistrement des noeuds permet
  - Détection si l'hôte/le routeur n'est pas atteignable
  - Résolution d'adresse (à priori)
  - Détection des adresses de dupliques
- Les enregistrements sont des liens temporaires (soft bindings)
  - Rafraîchissement périodique avec un nouveau message NR

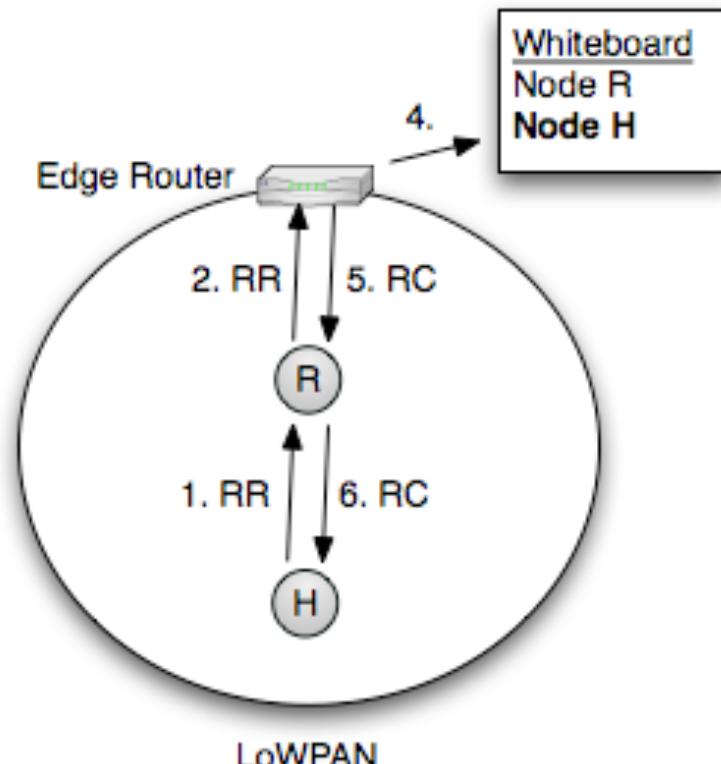
# Example of a typical 6LoWPAN-ND exchange



RS : Router Sollicitation  
 RA : Router Advertisement  
 NR : Node Registration  
 NC : Node Confirmation

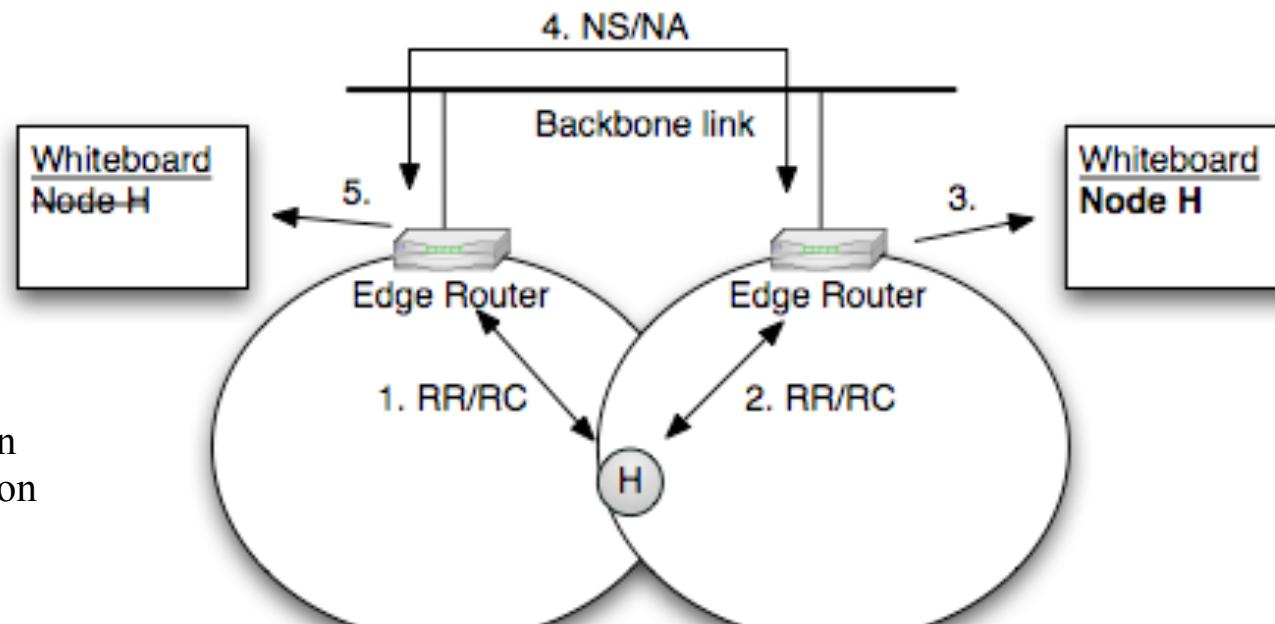
# Le Whiteboard

- The whiteboard is used in LoWPAN for :
  - Detect **duplicate addresses** for LoWPAN (= prefix)
  - Manage **mobility** (extended LoWPANs)
  - Generate **short addresses**
  - Locate** the nodes



# Extended LoWPANs

- Extended LoWPANs consist of two or more LoWPANs:
  - Sharing the same IPv6 prefix
  - Interconnected by a the same backbone link
- The whiteboards are synchronized on the backbone link.



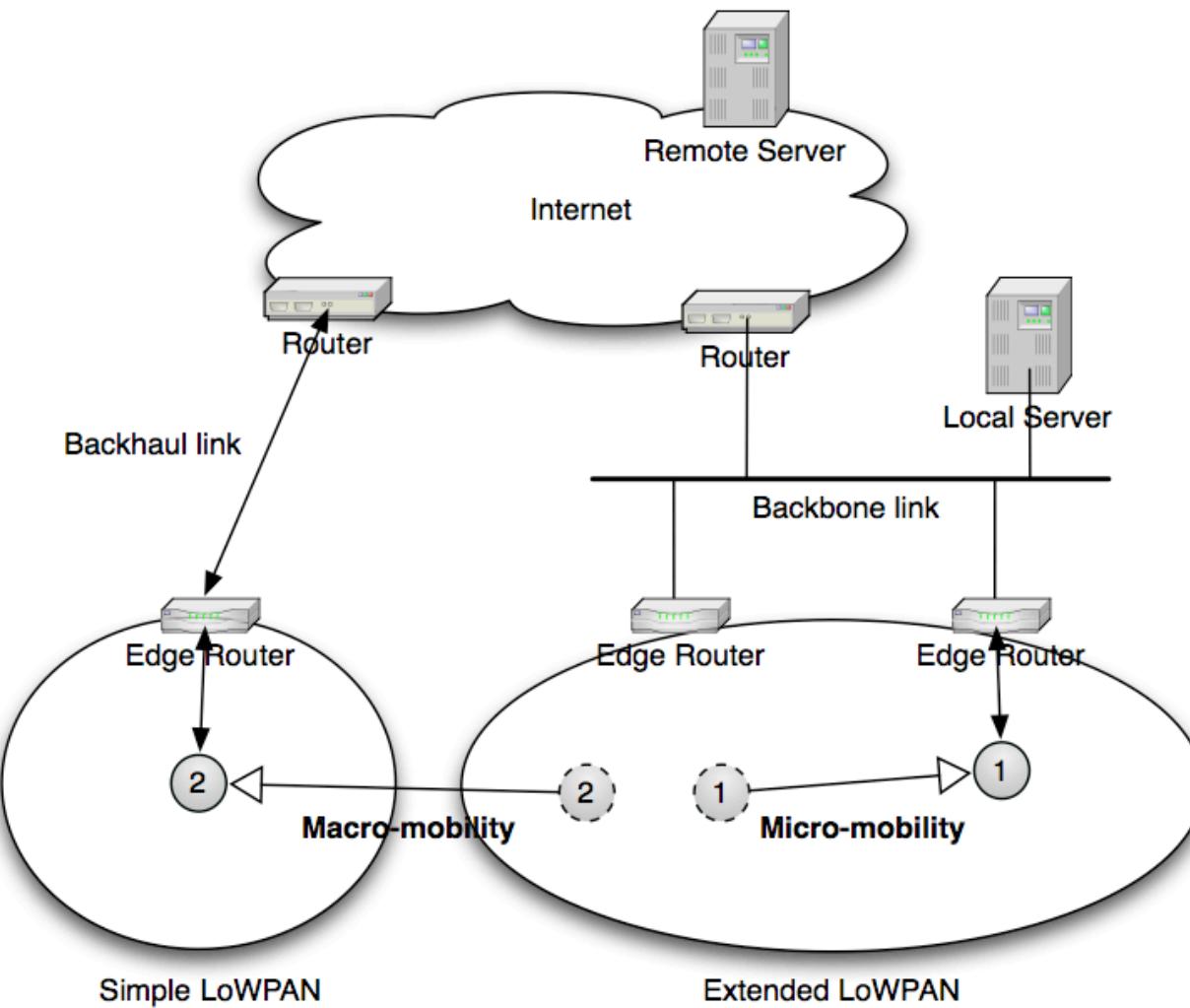
RR : Router Registration  
RC : Router Confirmation

# Mobility

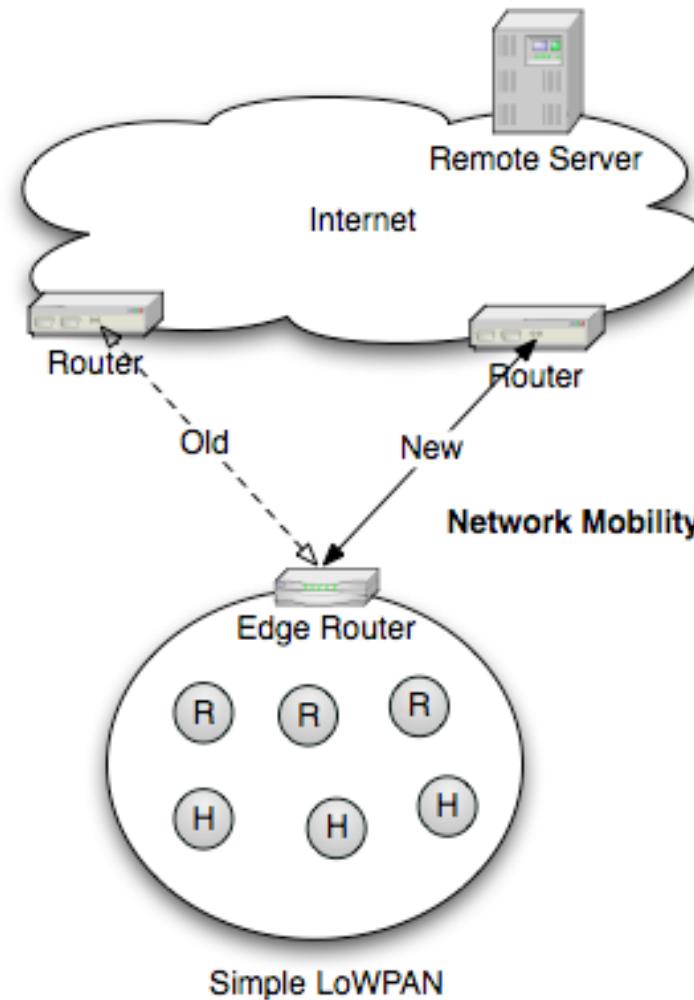
# Different types of mobility

- Mobility includes two processes:
  - **Roaming** – *moving from one network to another*
  - **Handover** – *change the attachment point (and data streams)*
- There are different classes of mobility:
  - **La Micro-mobility** – *within a network domain*
  - **La Macro-mobility** – *between different network domains (change of IP address)*
- There is a mobility of nodes and mobility of network
- What causes mobility?
  - A physical movement
  - A radio channel
  - Network performance
  - Duty-cycle process
  - Node failure

# Nodes mobility



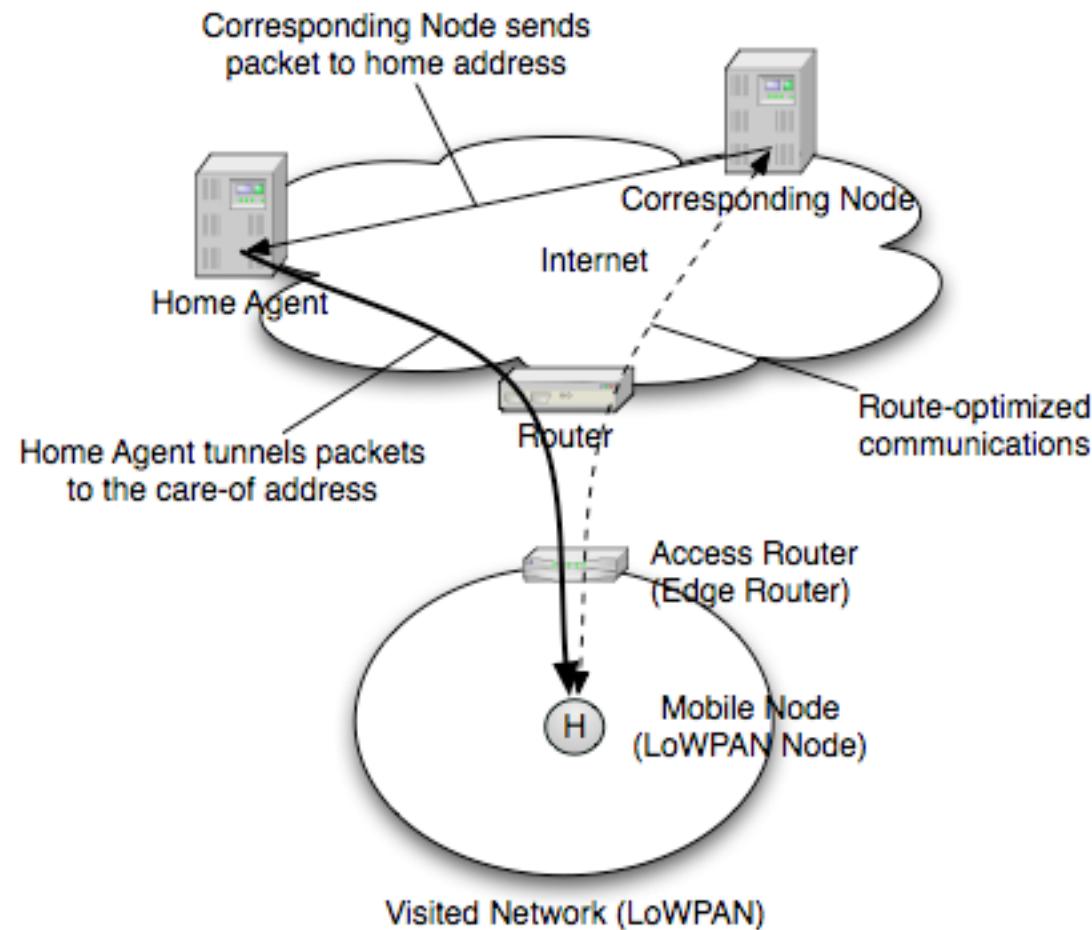
# Network mobility



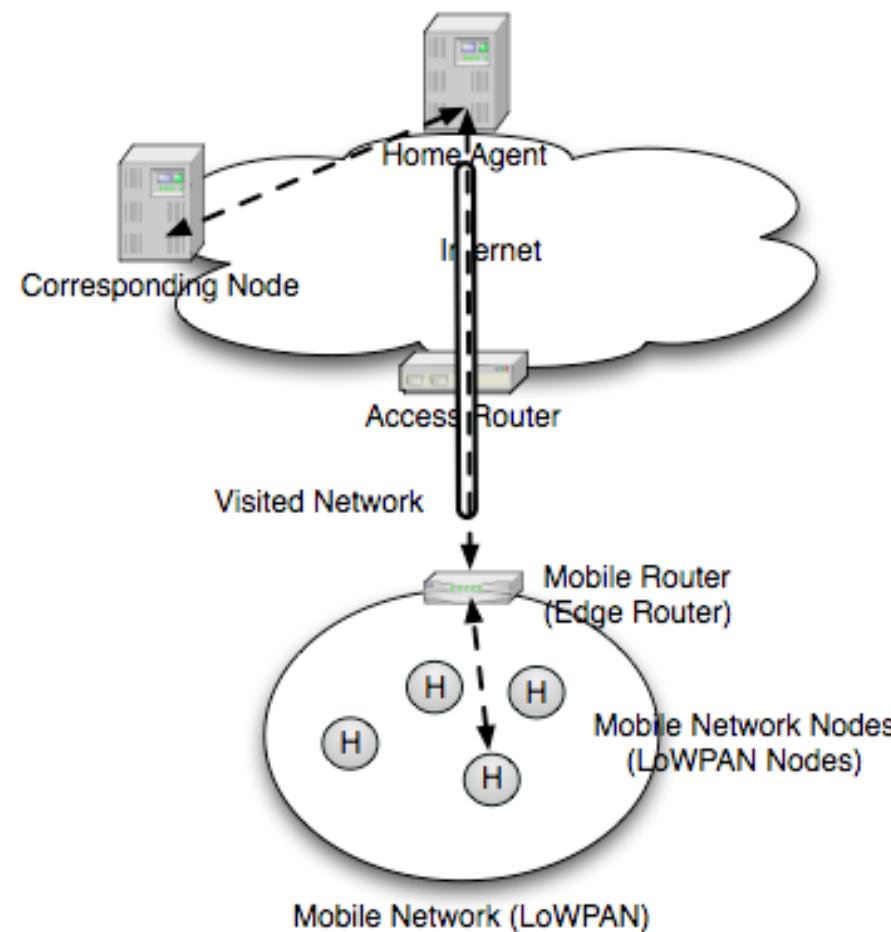
# How to deal with mobility?

- Micro-mobility
  - Link-layer techniques (i.e. GPRS, WiFi)
  - Extended LoWPANs to 6LoWPAN-ND
  - Routing protocols
- Macro-mobility
  - The application layer (SIP, UUID, DNS)
  - Mobile IPv6 [RFC3775]
  - Proxy Home Agent
- Network Mobility
  - NEMO (NEtwork MObility) [RFC3963]

# MIPv6



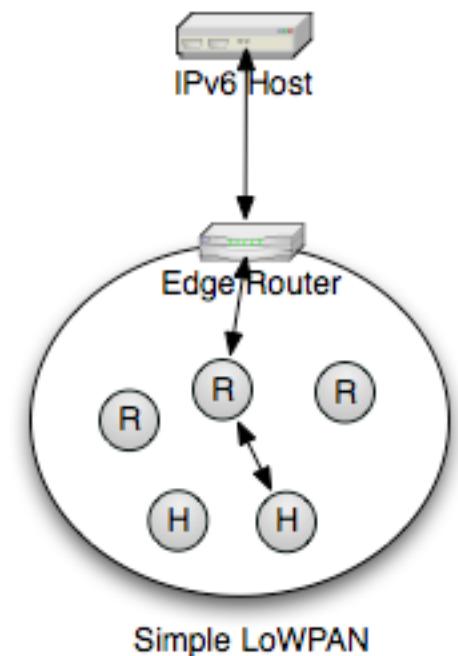
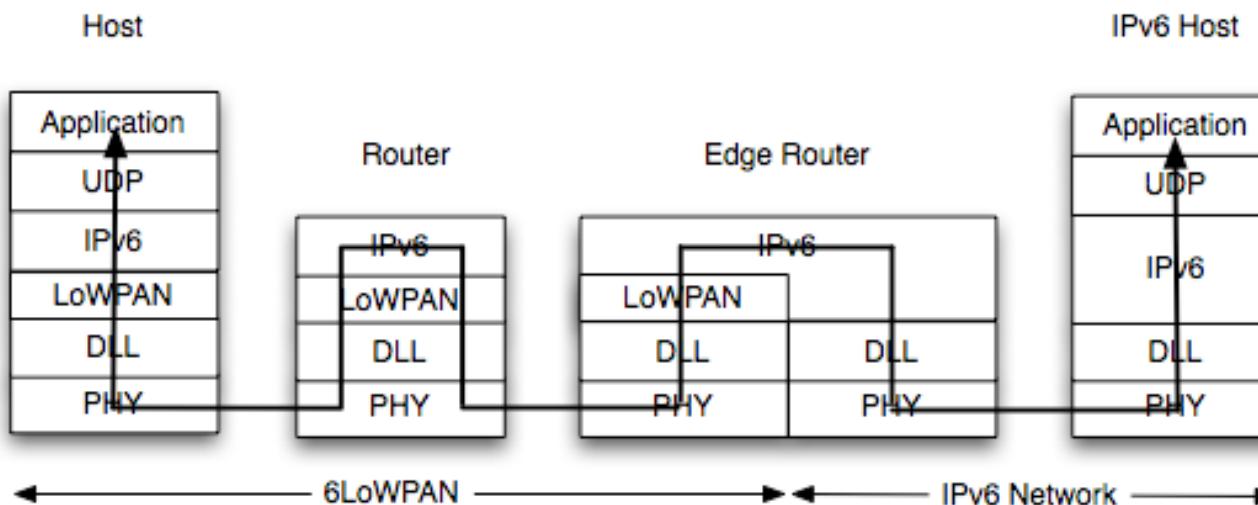
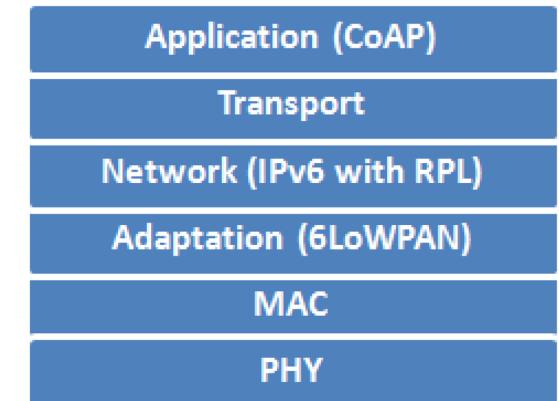
# NEMO



# Routing in 6LowPAN

# Routing with 6LoWPAN

- Routing in LoWPAN
  - Single interface routing
  - Flat address space (exact match)
  - End-to-end network (no transit routing)



# The different types of routing protocols

- Algorithm classes

- *Distance Vector*

*Links are associated with cost, and used to find the shortest path. Each router on the path records the local information of the next hop concerning its routing table.*

- *Link status*

*Each node obtains **complete information** about the network. Each node calculates a path tree to find the best path to each destination.*

- The different types of signals

- Proactive

*Routing information is obtained before it is needed.*

- Reactive

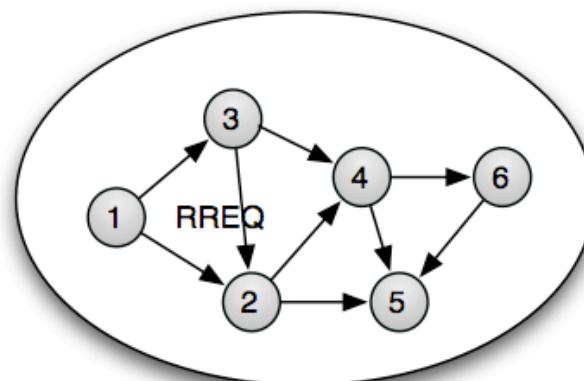
*Routing information is dynamically discovered when needed.*

# Protocoles pour 6LoWPAN

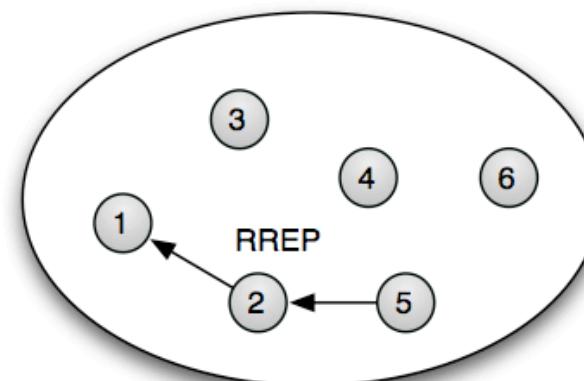
- L'IP supporte n'importe quel protocole de routage
  - Il transmet selon les entrées de la table de routage
- 6LoWPAN supporte n'importe quel protocole de routage
- Quelques spécificités du routage sur les LoWPAN :
  - Routage à interface unique, **topologie plate**
  - Technologies sans fil à faible puissance et avec pertes (LoWPAN)
  - Flux de données spécifiques pour les applications embarquées
- Protocoles MANET utiles dans certains cas ad-hoc :
  - i.e. AODV, DYMO
- Routage dans les réseaux à faible puissance et à pertes
  - Protocole RPL (Routing Protocol for Low power and Lossy networks) – RFC 6550
  - Développé spécifiquement pour les applications embarquées

# Protocoles MANET réactifs

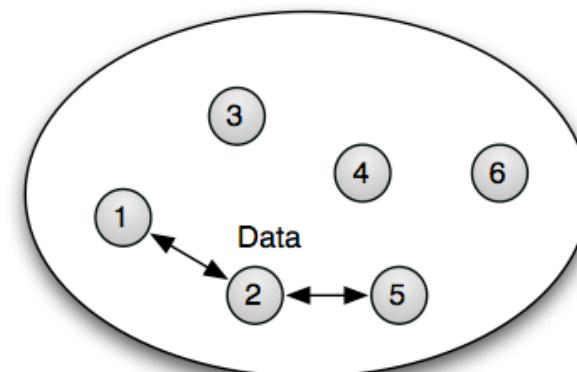
- Le cas d'AODV



1. RREQ for node 5 broadcast over multiple hops.



2. RREP unicast back to node 1, creates route entries.



3. Route entries in 1, 2 and 5 enable forwarding.

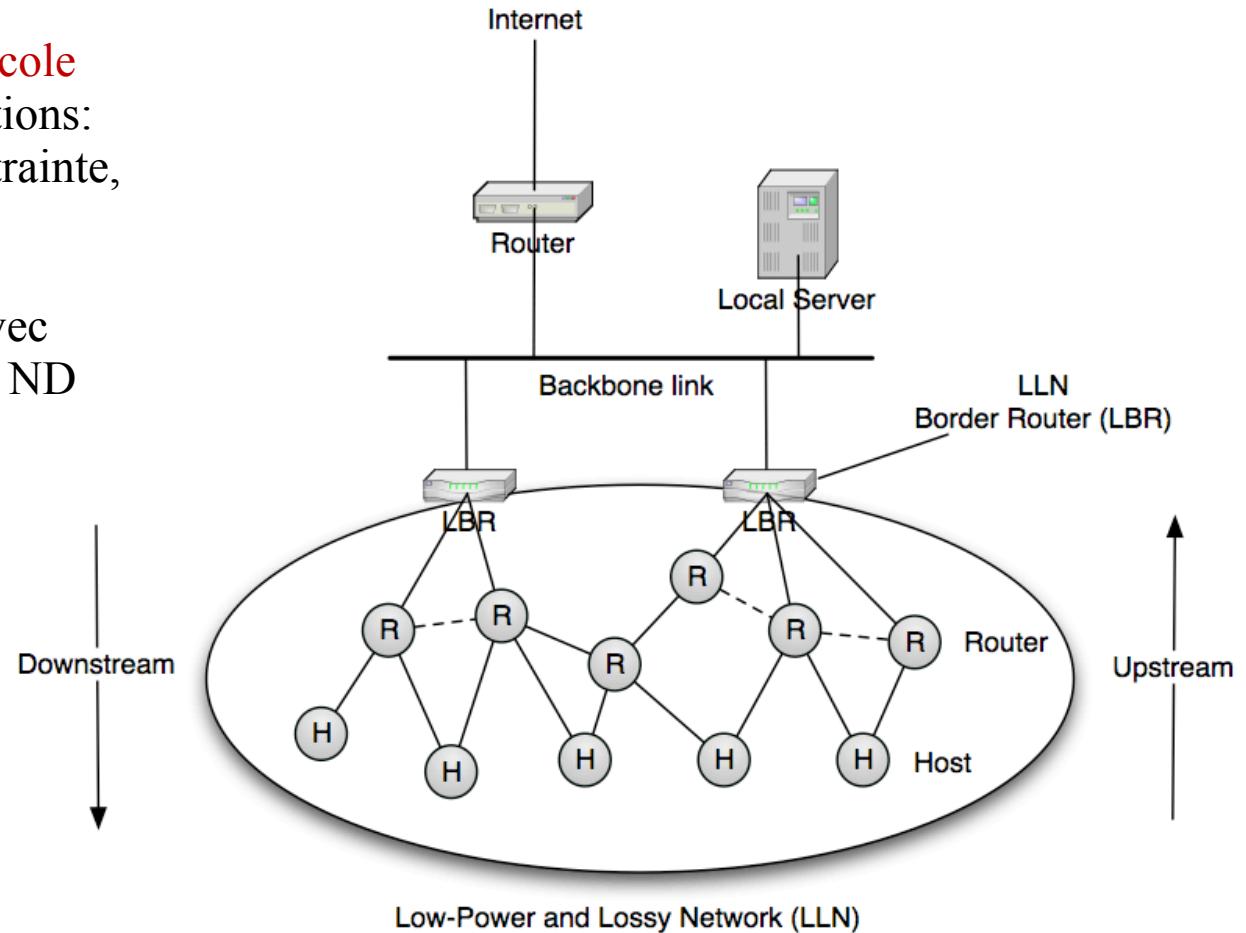
# IETF ROLL - RPL

- Groupe de travail à l'IETF standardise un algorithme de routage pour les applications embarquées
- Publié en mars 2012
- Exigences spécifiques relatives à l'application
  - Domotique
  - Immotique (bâtiments commerciaux)
  - Automatisation industrielle
  - Environnements urbains
- La solution doit fonctionner sur IPv6 et 6LoWPAN
- Protocole est nommé **RPL** (Routing Protocol for Low power and Lossy networks) – RFC 6550
  - Approche vecteur distance proactive

# ROLL RPL

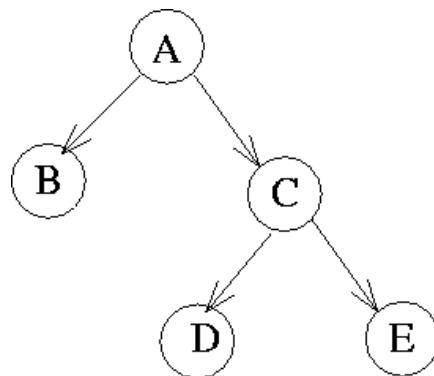
Le protocole RPL est classé comme un **protocole proactive de vecteur de distance** avec des options: routage multi-topologique, routage avec contrainte, etc.

Le protocole ROLL maintient la topologie avec l'utilisation des message de notification (e.x. ND Router Advertisements)

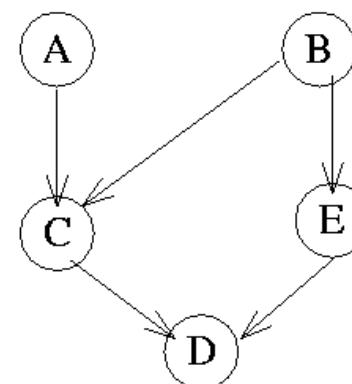


# Protocole RPL - Définitions

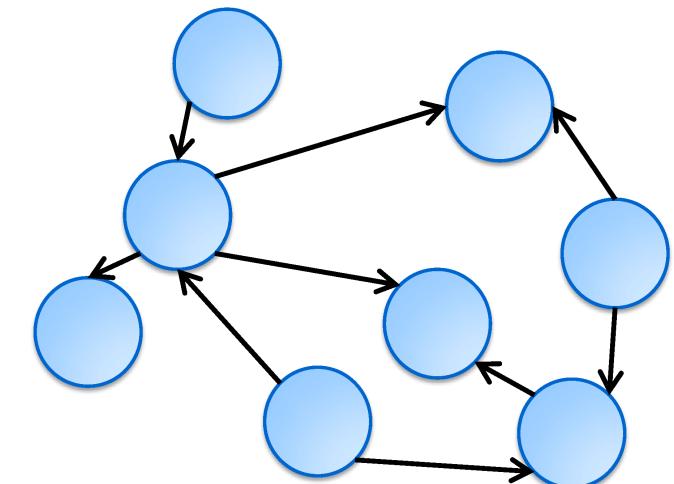
- RPL est basé sur la construction d'un arbre de routage sans boucle ou tous les noeuds peuvent communiquer
- La notion de graphe acyclique orienté nommé DAG (Directed Acyclic Graph) est utilisé par RPL
- Un DAG décrit les liens orientés entre les nœuds du réseau.



Tree



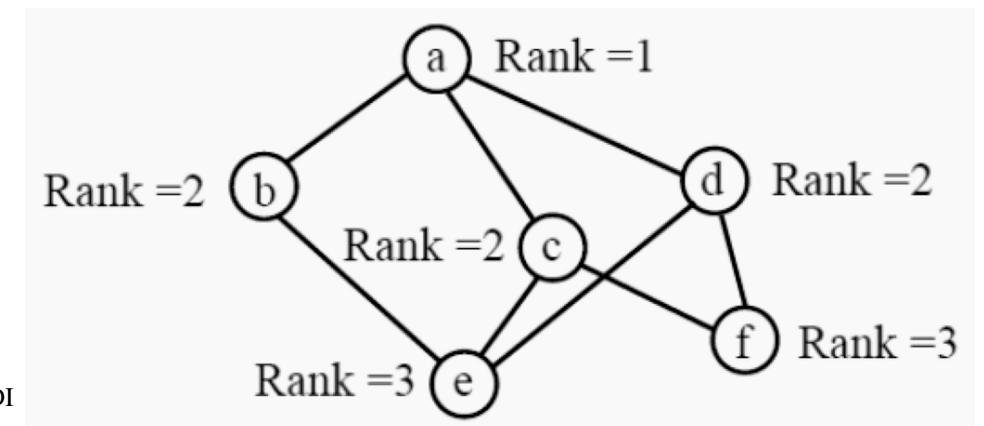
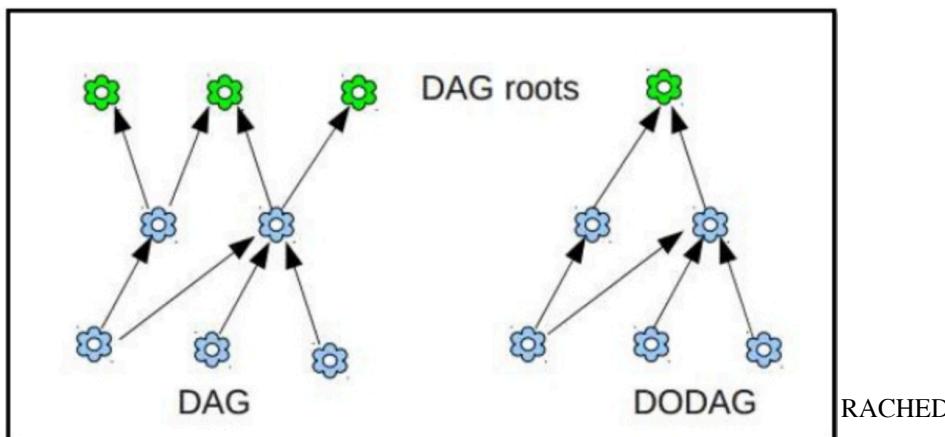
DAG but not a tree.



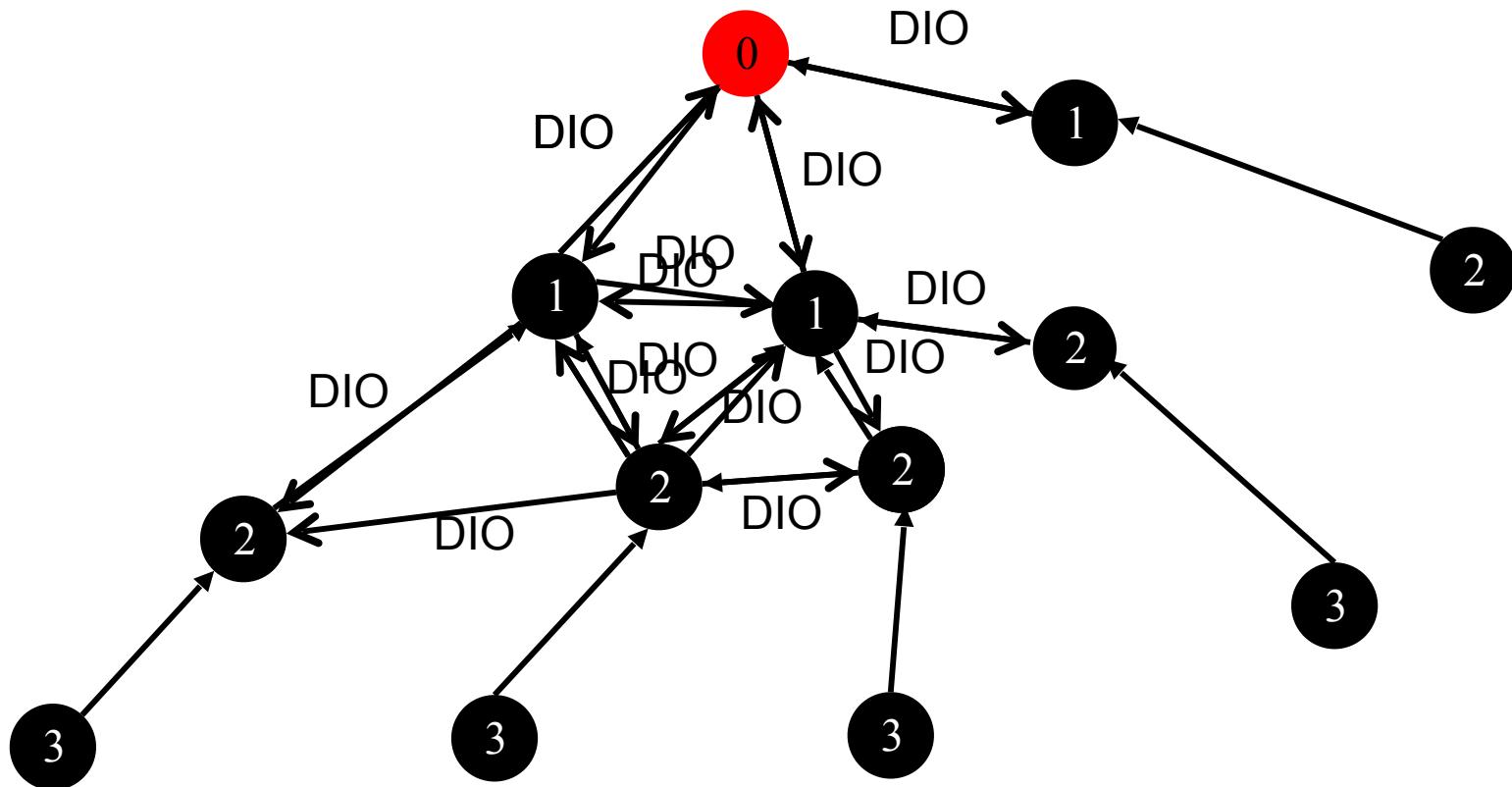
Exemple d'un DAG

# Protocole RPL

- Un DODAG (*Destination Oriented Directed Acyclic Graph*) est un DAG avec une seule destination à la racine.
- Dans 6LowPAN, la racine est le LBR (*Low power and Lossy network Border Router*)
- Le rang du nœud (*RPL Node Rank*) définit la position du nœud dans le DODAG à partir de la racine
- Le LBR envoie périodiquement en multicast des DIO (*DODAG Information Object*) pour construire et mettre à jour le graphe
- Lorsqu'un équipement reçoit une nouvelle version de DIO, il mis-à-jour son rang, et il fait suivre son DIO



# Protocole RPL – Exemple avec DIO



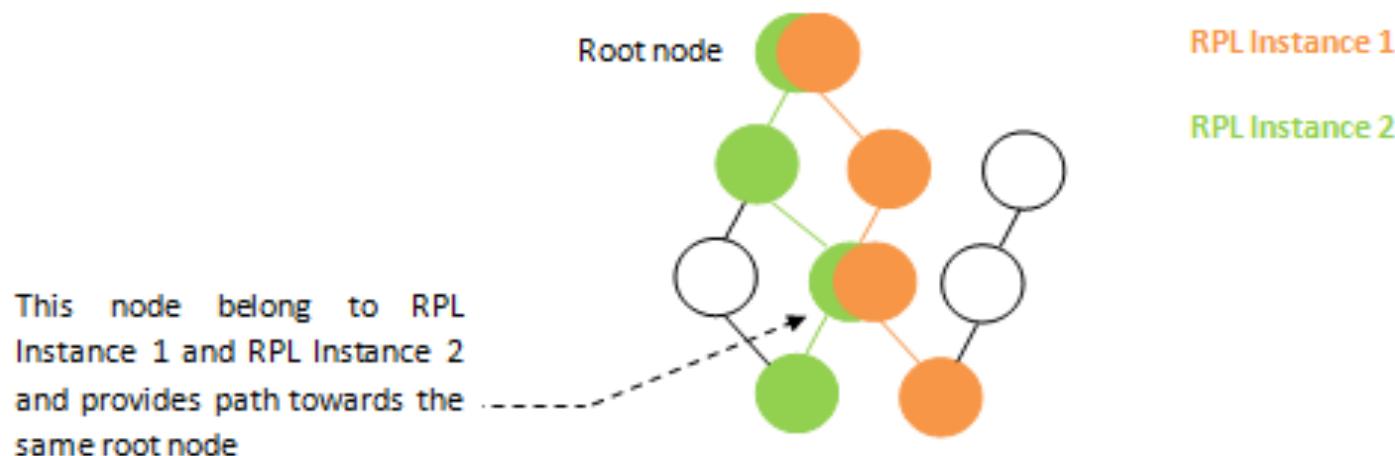
- DIO est diffusé afin de construire le graphe et il contient le rang du noeud, ETX, etc

# RPL – la notion d'Instance

- Une instance permet de définir un objectif d'optimisation (*Objective Function*) d'une route vers la racine suivant un ou plusieurs métriques.
- L'objectif est donné par la racine (ROOT)
- Une fonction objective permet aux nœuds de sélectionner les parents
- Les métriques permettent de prendre en compte :
  - des propriétés sur les liens (débit, latency, fiabilité, distance...) ;
  - des propriétés sur les nœuds (énergie résiduelle, la porté radio,...)
- Plusieurs instances peuvent être utilisées sur un même réseau physique avec différents critères d'optimisation

# RPL – Topology

- RPL combine deux type de topologies : mesh et hiérarchique
- Multi-Topology Routing (MTR)
  - Permet une grande fiabilité et de fléxibilité avec différentes possibilités de routes



**Figure . Multi-topology routing (MTR)**

# RPL – Types de messages

- RPL introduit trois types de messages de contrôle ICMPv6 :
  - **DIO (DAG Information Object)** : contient l'information permettant à un nœud de : - découvrir une instance RPL (paramètres de configuration)  
- de calculer son rang, - de choisir des parents dans le DODAG.
  - **DIS (DAG Information Solicitation)** : pour solliciter un DIO à partir d'un nœud RPL
  - **DAO (DAG Advertisement Object)** : pour propager les informations de destination en remontant le DODAG.

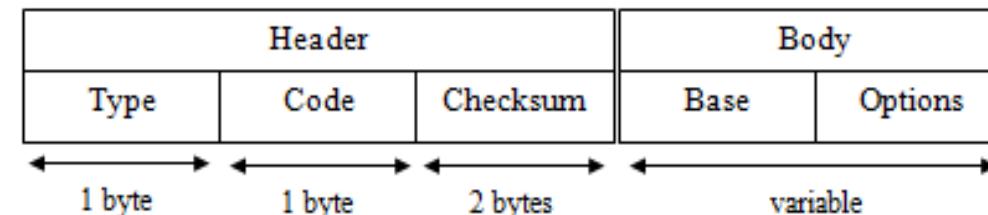
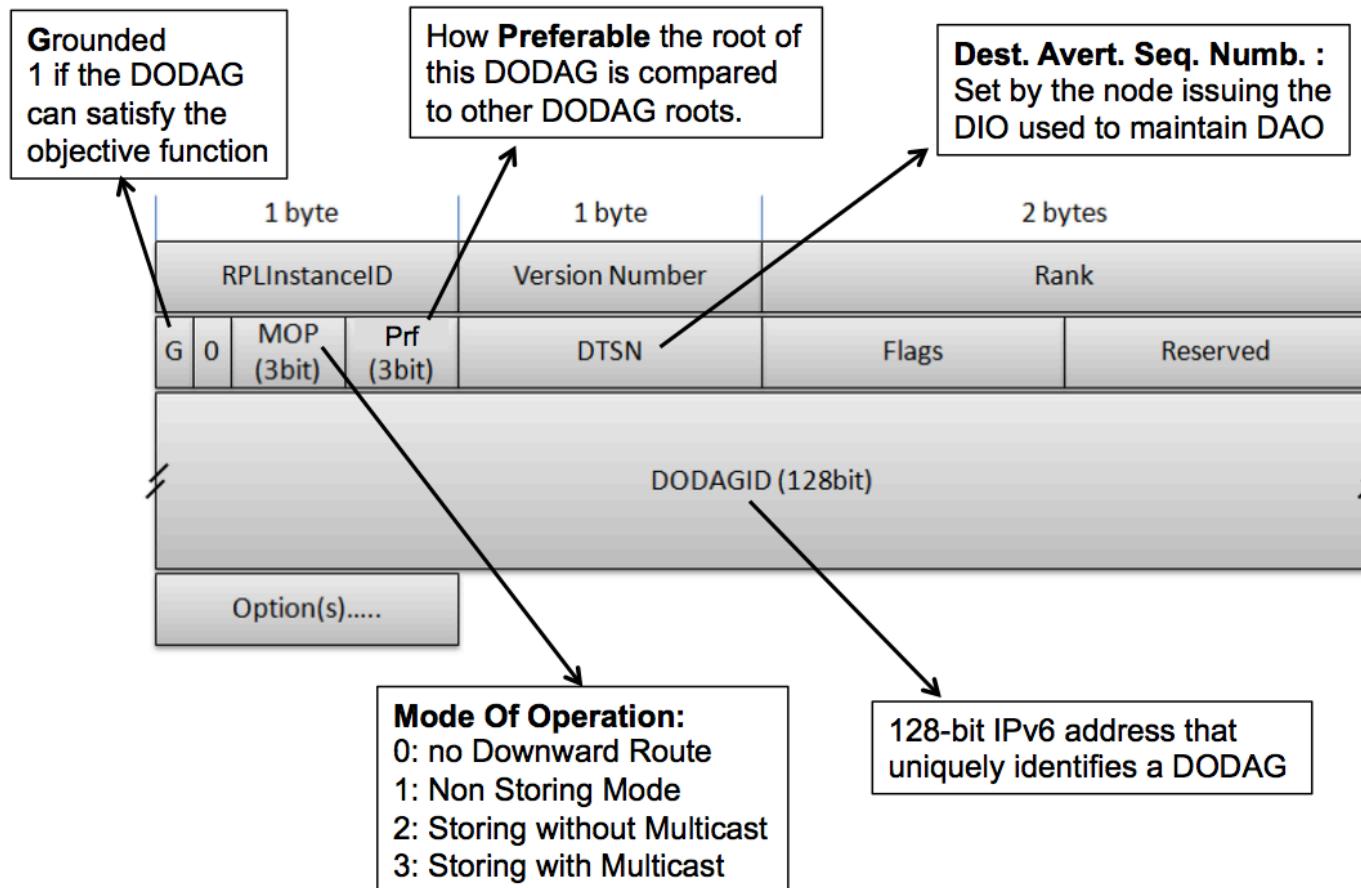


Figure 2. RPL control message

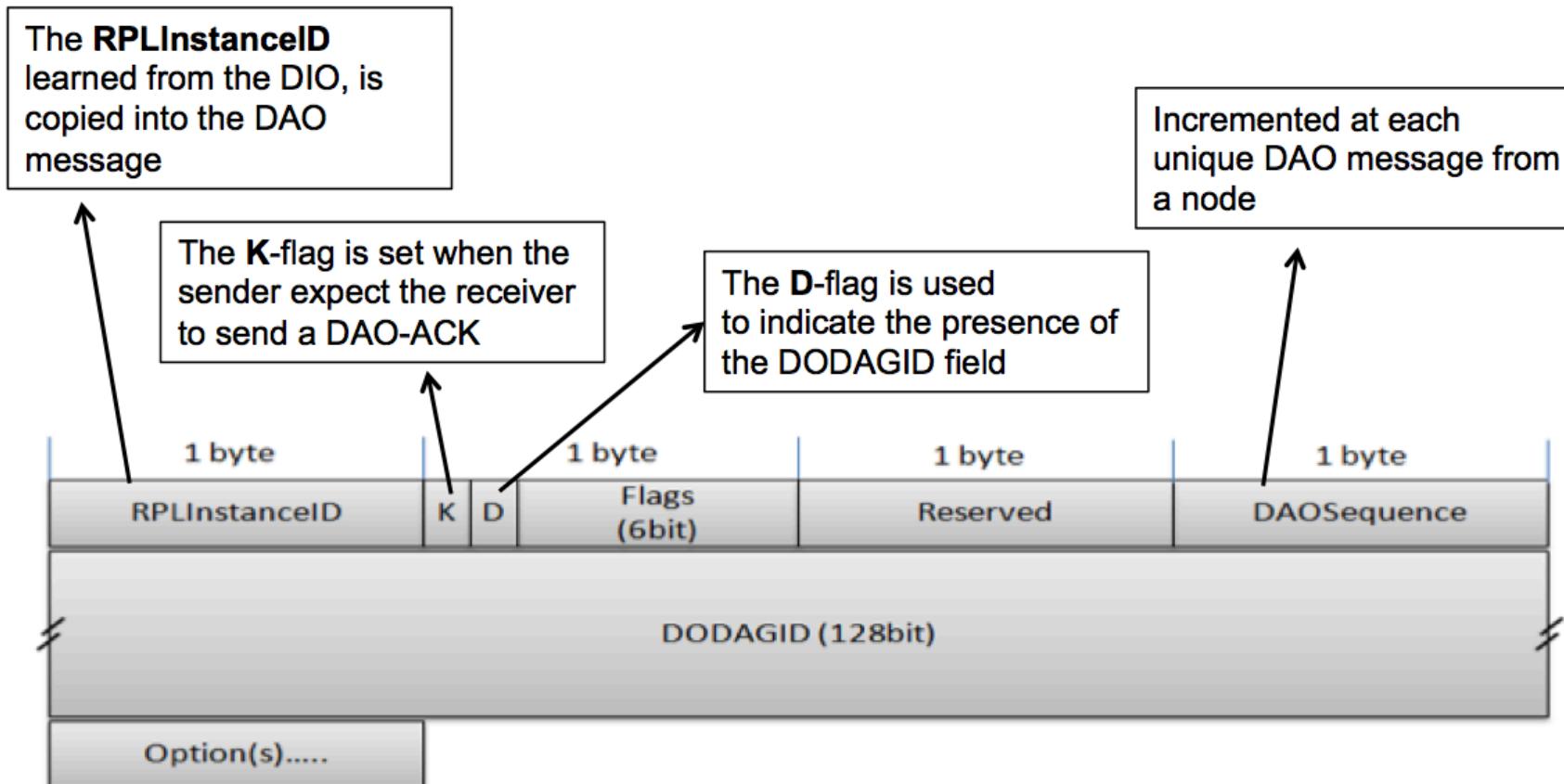
# RPL- Format des messages

## Message DIO : DODAG Information Object



# RPL- Format des messages

## Message DAO : Destination Advertisement Object



# RPL – Type de transport de données

- **Multipoint-to-point** ([data gathering](#)): Upward forward
  - Les paquets sont égualisés vers les meilleurs parents
- **Point-to-multipoint** ([data dissemination](#)): Downward forward
  - Storing mode
    - Les données sont acheminées sur la base des routes des tables de routage de chaque nœud dans le DODAG
    - Ce mode peut activer ou désactiver le multicast
  - Non storing mode (Source Routing Header)
    - Les données sont acheminées dans la route indiquée par le nœud ROOT dans l'en-tête du paquet (RH4)
- **Point-to-point** ([data reading](#))
  - Les paquets sont transmis directement avec un seul saut
  - RPL ne définit pas des routes optimales pour ce type de communication

# RPL- la fonction objective (OF)

- La fonction objective permet de comparer les metriques de deux objets et de choisir :
  - Le “Preferred Parent” entre deux candidats A et B
  - Le rang du noeud en prenant en compte le “Preferred Parent”
- La fonction objective 0 (Of0)
  - OF très simple, le noeud va sélectionné toujours le noeud avec le plus petit rang
- Minimum Rank Hysteresis Objective Function (Mrhof)
  - Le “Preferred Parent” is changé uniquement si la différence du rang est supérieure à une certaine valeur. Cette fonction permet une optimisation moins agressive et des DAGs plus stable
- ETXOF : C'est la fonction MRHOF avec ETX comme métrique
  - La métrique ETX (Estimated Transmission Count) est une estimation du nombre total de transmissions nécessaires pour atteindre la destination.
  - ETX est calculée en additionnant l'inverse du taux de réception de paquets (PRR) de chaque saut sur le chemin.

# Security with 6LoWPAN

# Security for 6LoWPAN

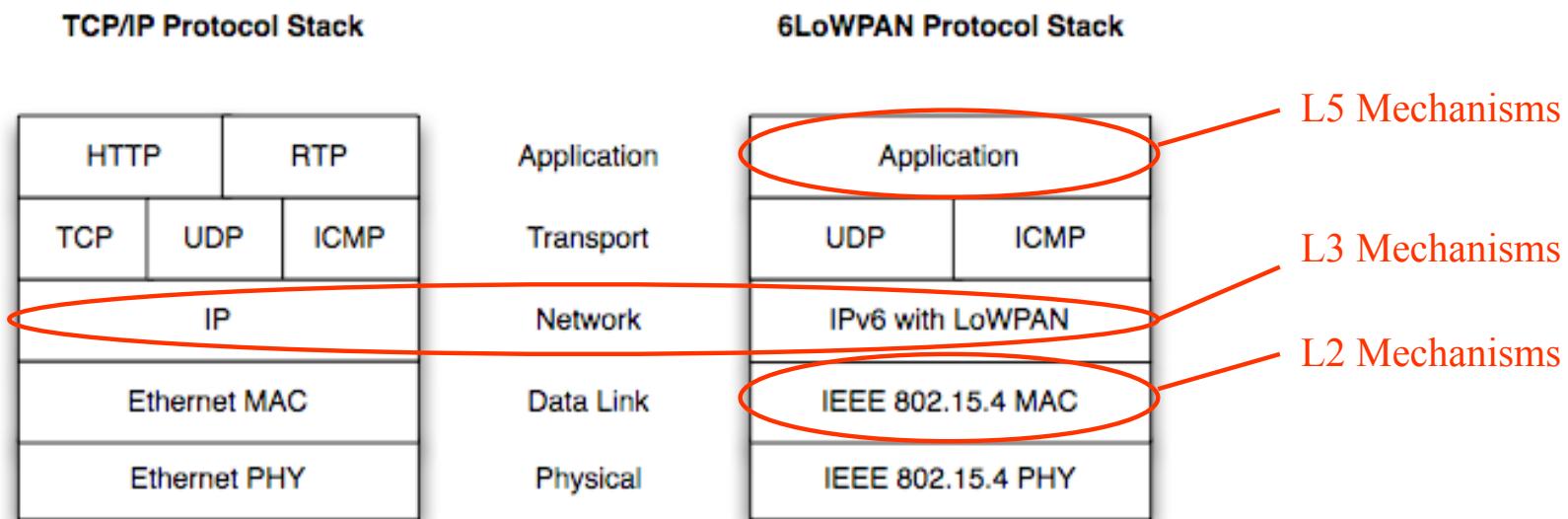
- Characteristics of networks of communicating objects:
  - Communication channel is wireless
  - Open Network
  - Resource constraints: energy, computing and memory
  - Deployment in a hostile environment (physical security)
  - Lack of infrastructure



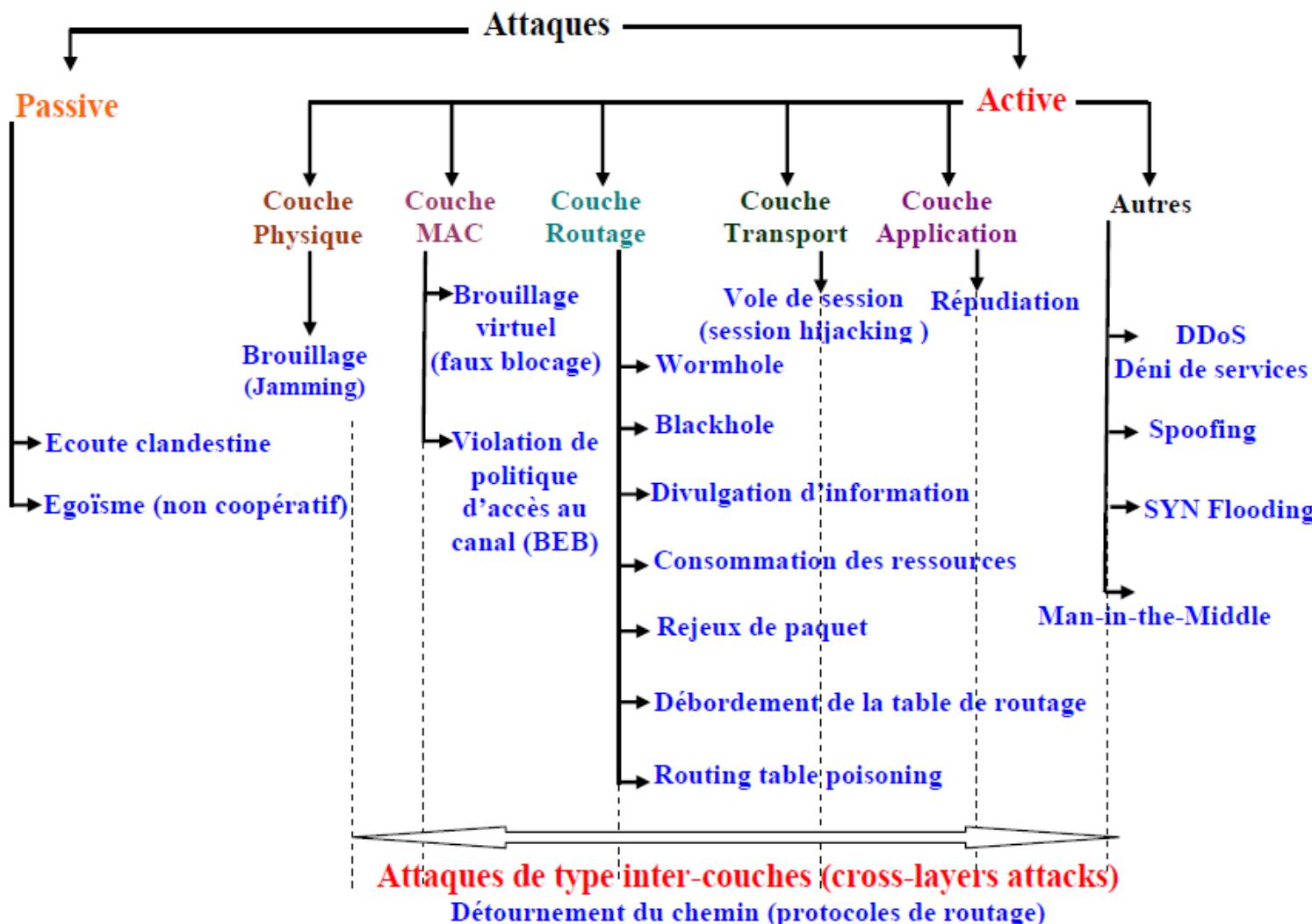
- New Vulnerabilities
  - Ecoute clandestine du canal de communication
  - Association des nœuds malicieux
  - Faire consommer l'énergie des nœuds inutilement
  - Capture des nœuds et violation de confidentialité des données
  - Difficulté de gérer la sécurité dans l'environnement dynamique

# Sécurité pour 6LoWPAN

- Un système a en général trois principaux objectifs de sécurité
  - Confidentialité
  - Intégrité
  - Disponibilité
- Consulter le modèle de menace pour la sécurité Internet dans RFC3552



# Classification des attaques



# Mécanismes de la 2ème couche

- La sécurité Internet est en général considérée de bout-en-bout
- Dans les réseaux sans fil, le canal lui-même est très vulnérable
  - Le canal peut facilement être écouté
  - Les noeuds et les paquets sont faciles à copier
- La sécurité sur la couche liaison de données vise donc à :
  - Protéger le réseau sans fil contre les attaquants
  - Augmenter la robustesse contre les attaques
- IEEE 802.15.4 fournit un chiffrement inclus
  - Basé sur le Standard de Chiffrement Avancé (AES) de 128 bits
  - En concurrence avec le mode CBC-MAC (CCM)
    - Il propose également la vérification du chiffrement et de l'intégrité
  - La plupart des puces contiennent du matériel du modèle AES-128

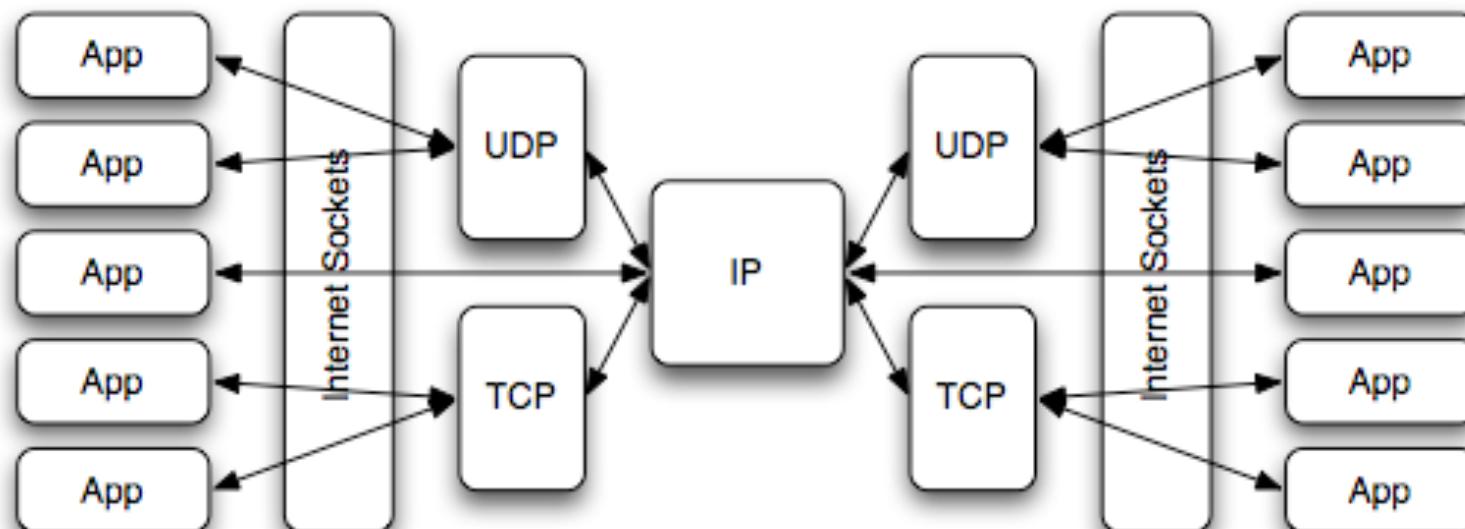
# Mécanismes de la 3ème couche

- La sécurité de bout-en-bout peut être assurée par l'IP
  - Il protège la route entre deux points
- La sécurité IP est définie par le standard IPsec
- Deux formats de paquet sont définis :
  - Authentication Header (AH) dans [RFC4302]
    - Authentification et protection de l'intégrité uniquement
  - Encapsulating Security Payload (ESP) [RFC4303]
    - Chiffre également la confidentialité
- ESP est le format le plus utilisé
- Un mode d'ESP définit l'utilisation d'AES/CCM [RFC4309]
  - Peut être utilisé avec les noeuds 6LoWPAN
  - La même matériel L2 IEEE 802.15.4 peut être appliqué !

# Formats et Protocoles Applicatifs

# Introduction

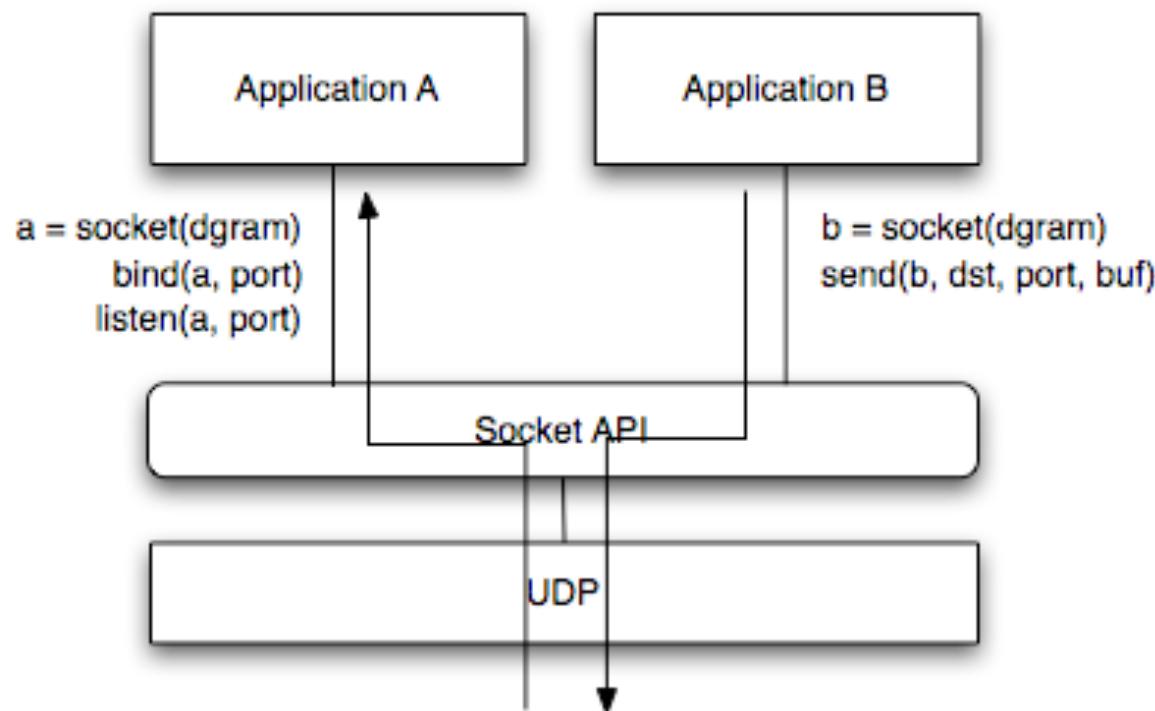
- Les processus applicatifs communiquent sur IP en utilisant une approche **Internet Socket**
- 6LoWPAN utilise aussi le paradigme Internet Socket
- Les protocoles applicatifs utilisés avec 6LoWPAN requièrent, eux, une **conception et des performances** spécifiques



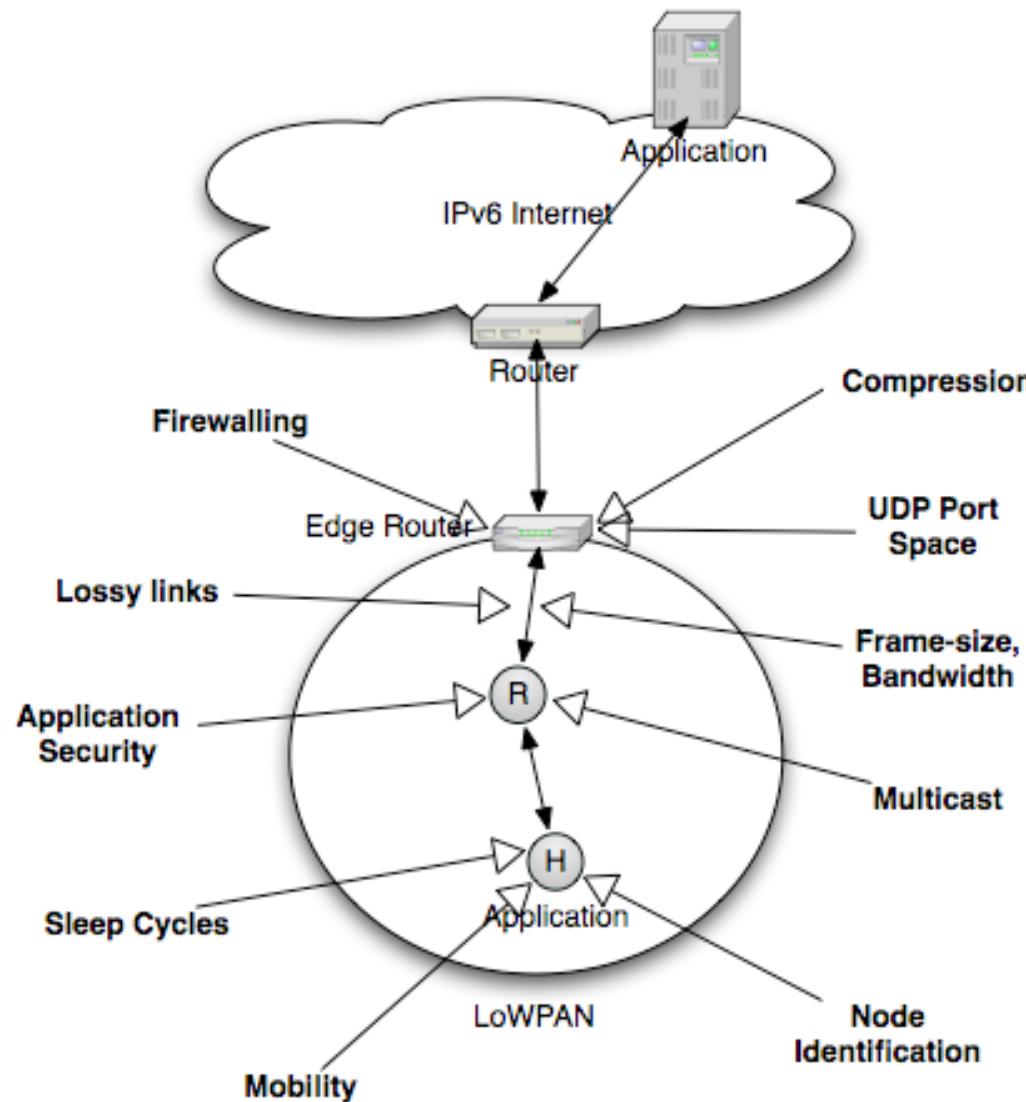
# Socket API

- La Socket API fournit **un accès aux communications** de données pour les applications
- **Interface connue** pour traiter le flux de données et la gestion des buffers via une Socket
- Supporte aussi les messages de contrôle aux protocoles
- Les commandes incluent :
  - socket, bind, send, read, close etc.
- Exemples de Socket API
  - Sockets Berkeley dans les systèmes \*nix
  - Mac OSX (Darwin)
  - Contiki **uIP** (approche pseudo Socket)

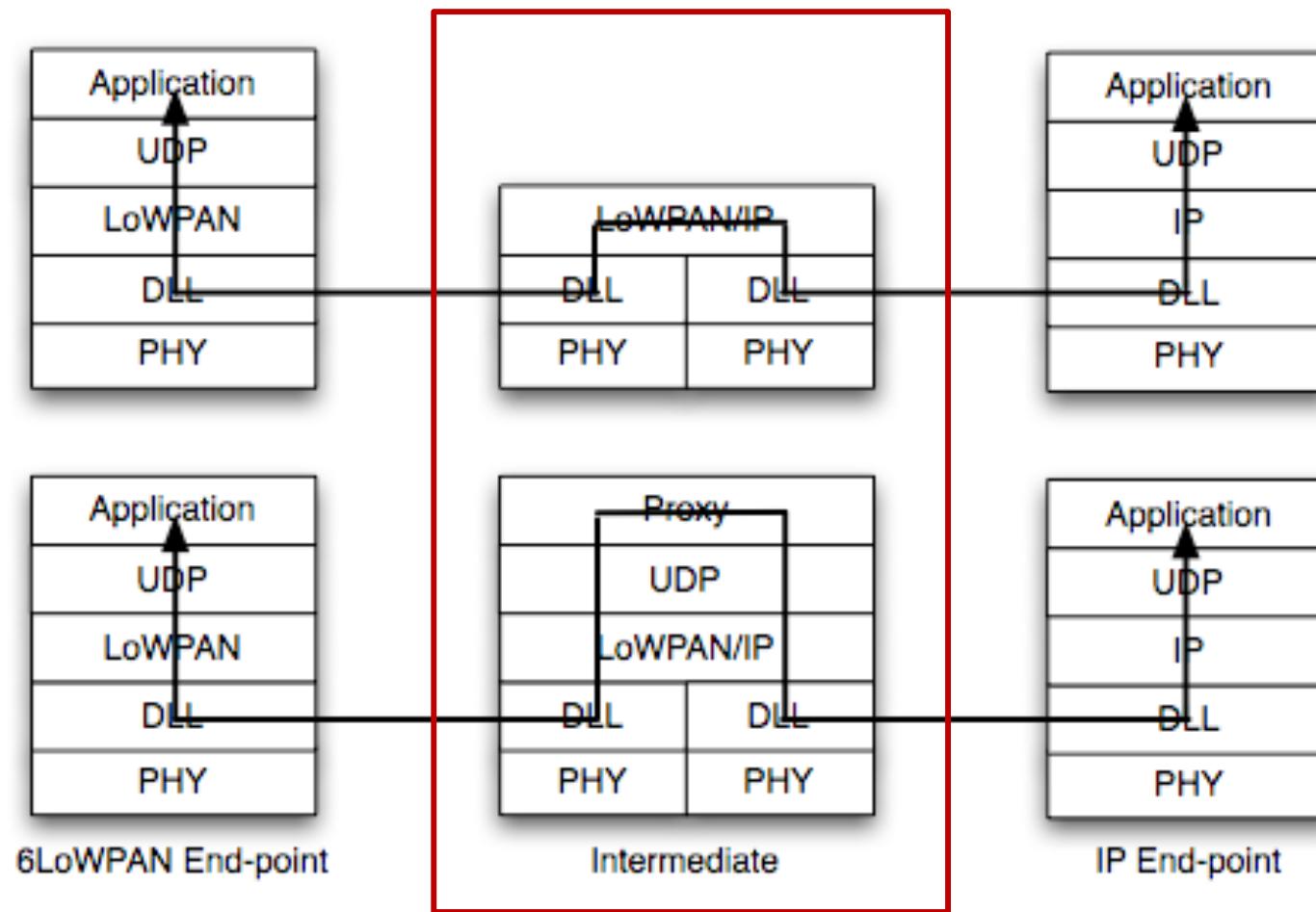
# Socket API



# Difficultés de conception



# Paradigme de bout-en-bout



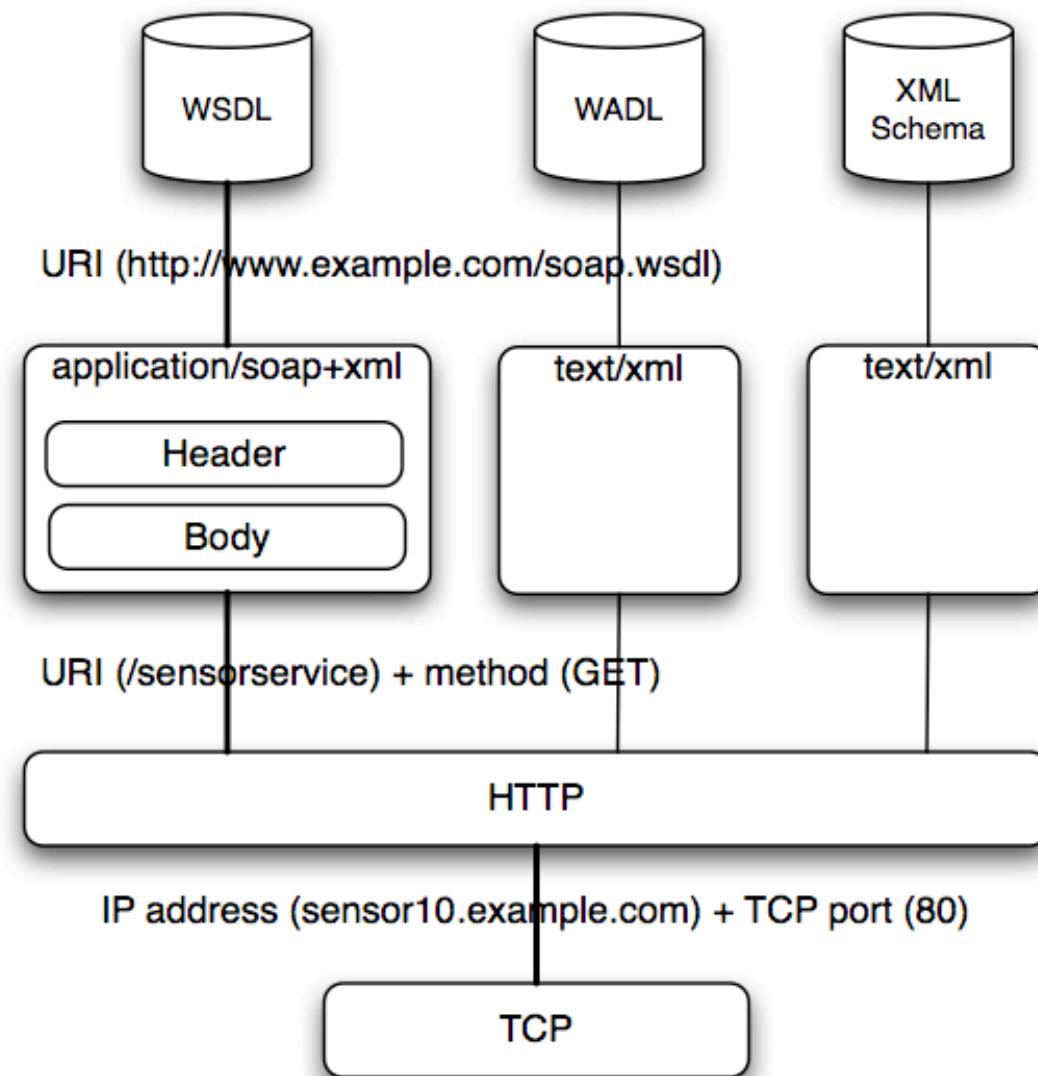
# Paradigme du service web

**WSDL** : Web Services Description Language

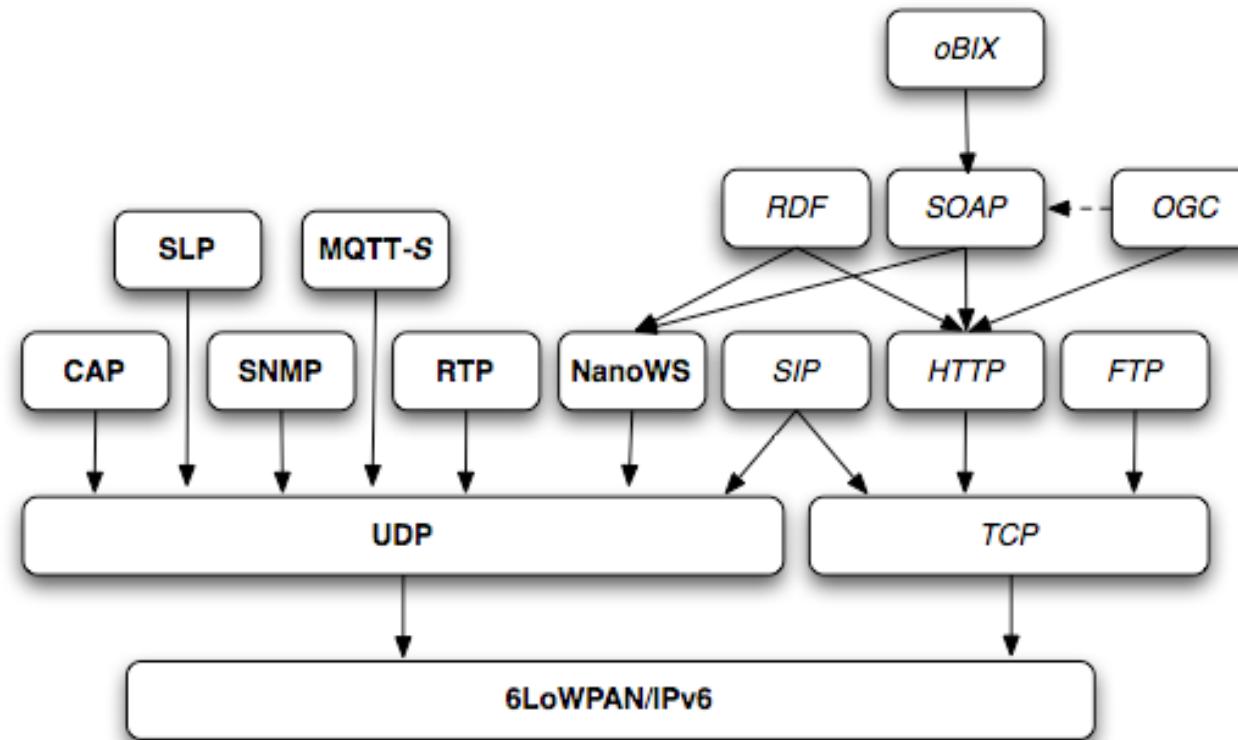
**WADL** : Web Application Description Language

**XML** : Extensible Markup Language, « langage de balisage extensible »

**Soap** : Simple Object Access Protocol



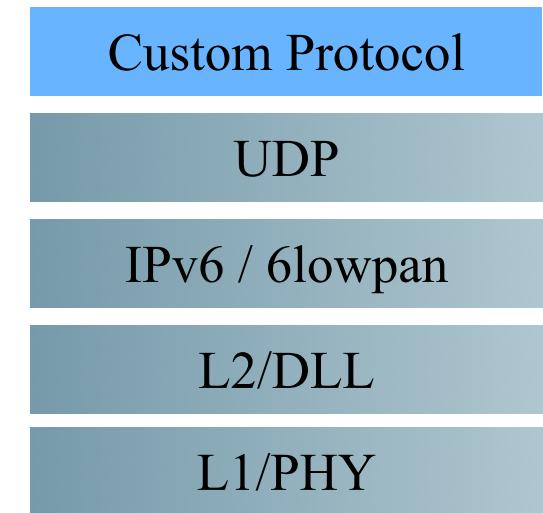
# Formats et Protocoles Applicatifs



- **SLP:** Service Location Protocol est un protocole de **découverte de service**
- **SNMP :** Simple Network Management Protocol est un protocole de **gestion réseau**
- **MQTT-S :** Message Queue Telemetry Transport for Sensor est un protocole de *transport de message pour le service publish/subscribe*
- **NanoWS :** Nano Web Services
- **RDF :** Resource Description Framework est un modèle de graphe destiné à décrire de façon formelle les ressources Web
- **OGC :** Open Geospatial Consortium est développé pour assurer une compression large des données géospatiales
- **oBIX :** Open Building Information Exchange

# Personnalisation des protocoles

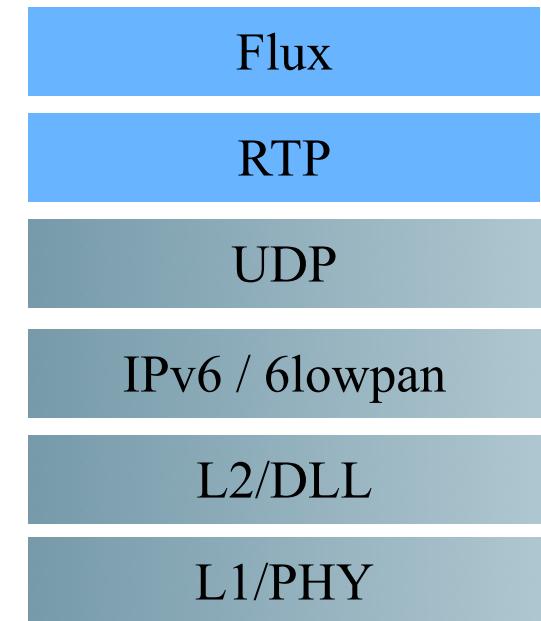
- Solution la plus commune aujourd’hui
- Données applicatives encodées de façon binaire, selon l’application
- Le protocole applicatif utilise un port UDP spécifique, selon l’application
- Comme 6LoWPAN désigne des communications IPv6 de bout-en-bout



- Avantages :
  - Compact, efficace, la sécurité peut être intégrée, de bout-en-bout
- Inconvénient :
  - Application serveur personnalisée nécessaire, peu de réutilisation, courbe d’apprentissage, interopérabilité

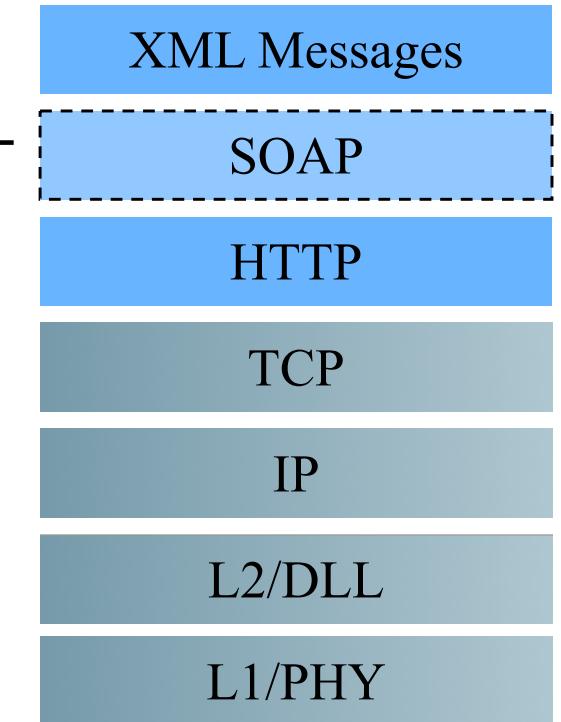
# Flux et RTP (Real-time Transfert Protocole)

- Solution de flux correcte
- Pour un flux audio ou de capteurs en continu
  - L'audio sur 802.15.4 nécessite un bon codec
- Avantages :
  - RTP peut être utilisé sur 6LoWPAN
  - Propose une solution de bout-en-bout
  - Aucune modification de serveur n'est nécessaire
  - Contrôle de la jigue (variation du délai)
- Inconvénients :
  - Les headers pourraient être plus efficaces pour un flux de données **de capteurs simple**



# XML/HTTP

- De-facto pour **les communications inter-serveurs**
- Tous les serveurs Internet utilisent le langage HTTP/XML
- Utilisable pour RPC, pub/sub et les événements
- Paradigme SOAP ou REST
- Avantages :
  - Schéma XML populaire
  - Séquences de message formelles
  - Support Internet large
- Inconvénients :
  - Inefficace, complexe
- Solution : services web embarqués
  - Consulter le document IETF sur l'application 6lowpan <http://6lowapp.net>



# Autres Protocoles Applicatifs

- Service Discovery
  - Service Location Protocol (SLP)
  - Device Profile Web Services (DPWS)
- Gestion
  - Simple Network Management Protocol (SNMP)
- Télémétrie M2M
  - MQ Telemetry Transport for Sensors (MQTT-S)
- Immotique
  - BACnet/IP
  - oBIX
- Industrie de l'énergie
  - ANSI C12
  - Device Language Message Specification (DLMS)

# Systèmes d'exploitation

# List of IoT-Related Projects

Name of Project/Product	Area of Focus
Tiny OS	Operating System
Contiki	Operating System
Mantis	Operating System
Nano-RK	Operating System
LiteOS	Operating System
FreeRTOS	Operating System
RIOT	Operating System
Wit.AI	Natural Language
Node-RED	Visual Programming Toolkit
NetLab	Visual Programming Toolkit
SensorML	Modeling and Encoding
Extended Environments Markup Language (EEML)	Modeling and Encoding
ProSyst	Middleware
MundoCore	Middleware
Gaia	Middleware
Ubiware	Middleware
SensorWare	Middleware
SensorBus	Middleware
OpenIoT	Middleware and development platform
Koneki	M2M Development Toolkit
MIHINI	M2M Development Toolkit

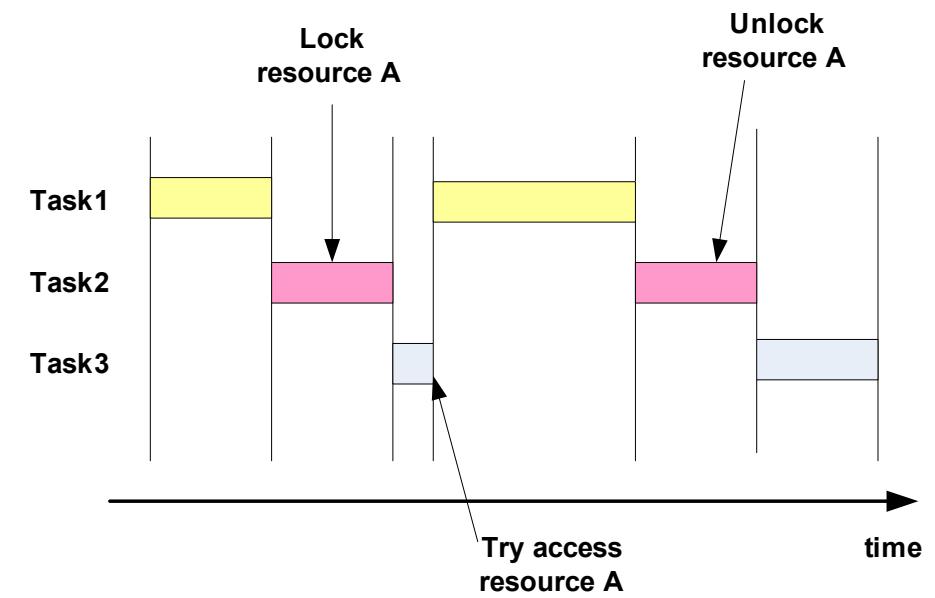
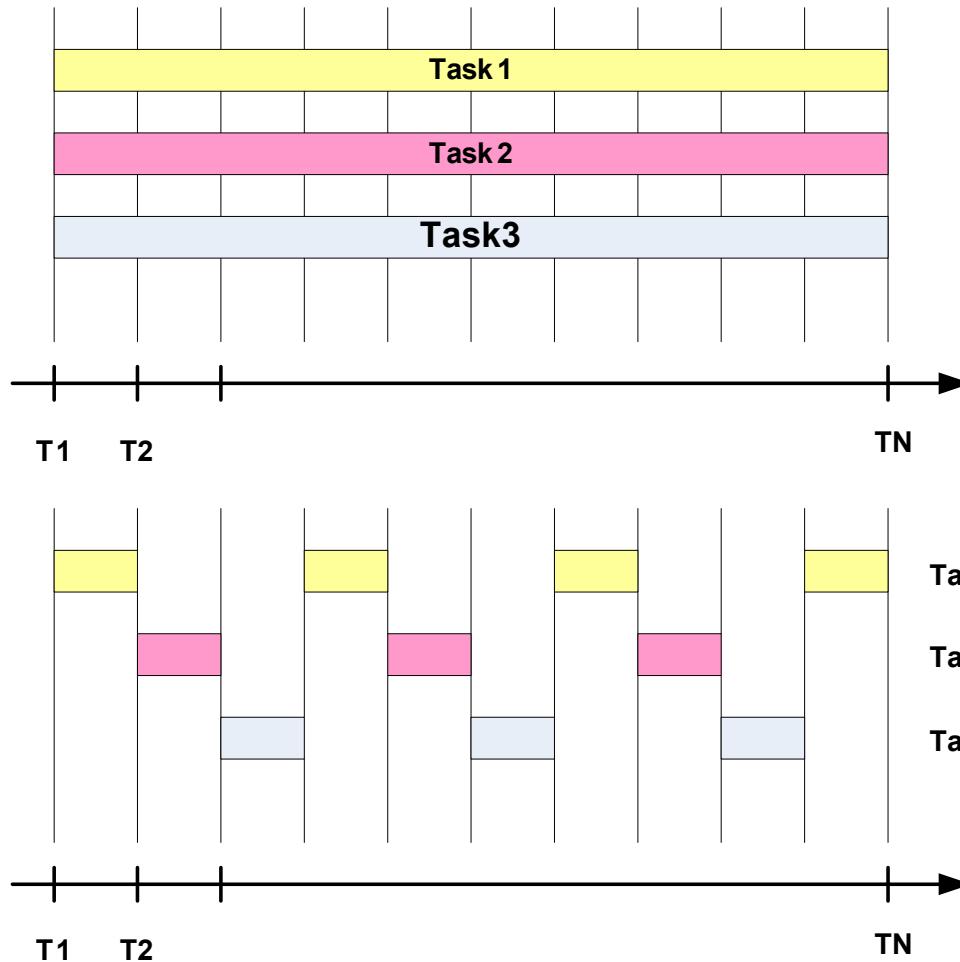
# Systèmes d'exploitation en temps réel

- Bibliothèque vs système d'exploitation
  - Le système d'exploitation gère toutes les ressources
- Les systèmes embarqués ont des tâches pré-définies
  - Conçus pour optimiser la taille, le coût, l'efficacité, etc.
- Temps réel
  - Le système d'exploitation en temps réel propose des outils pour **respecter les délais**
  - Prévention, queues/buffers, sémaphores
- Concurrence
  - Flux exécutifs (tâches) capables de fonctionner en **parallèle**
  - Fils d'exécution et processus
- Processeurs et API

# Difficultés relatives au temps réel

- Les systèmes embarqués sans fil fonctionnent en général en temps réel
  - Visualisation, robot, capteur de bâtiment, noeud de contrôle
- Les RTOS ne sont pas toujours très performants
  - Délais du système général (soft real-time)
  - Ou déterministe (hard real-time)
- Les délais peuvent être respectés en utilisant :
  - Des algorithmes de programmation **préventive spécifiques**
  - Une conception et une communication inter-tâches appropriées
  - Sémaphores et queues pour éviter une ruée

# Difficultés relatives au temps réel

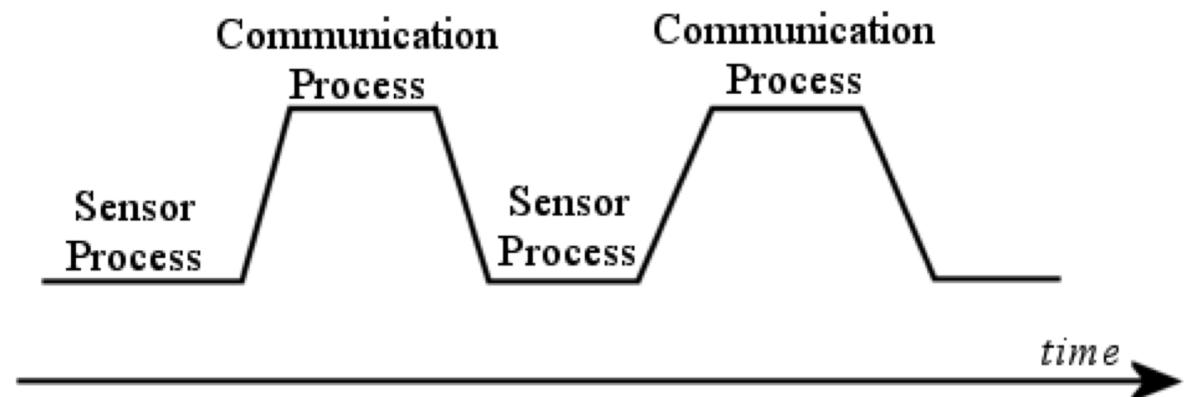


# Concurrence

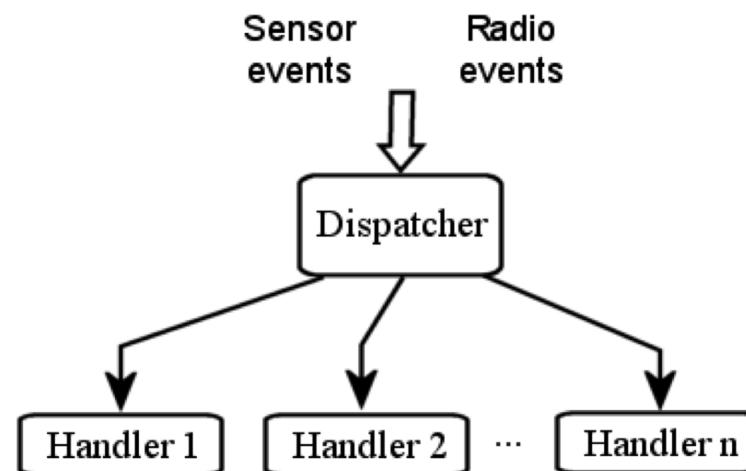
- On parle de concurrence lorsque deux ou plus de flux exécutifs fonctionnent en même temps
- Cette situation entraîne plusieurs problèmes, tels que :
  - Les conditions de concurrence à partir des ressources partagées
  - Impasse et famine
- Le système d'exploitation doit coordonner les tâches
  - Échange de données, mémoire, exécution, ressources
- Deux techniques principales
  - Processus
    - La durée CPU est divisée entre les différentes tâches exécutives
    - Les systèmes embarqués utilisent des fils plus clairs
  - Evénement

# Concurrence

- Processus



- Événement

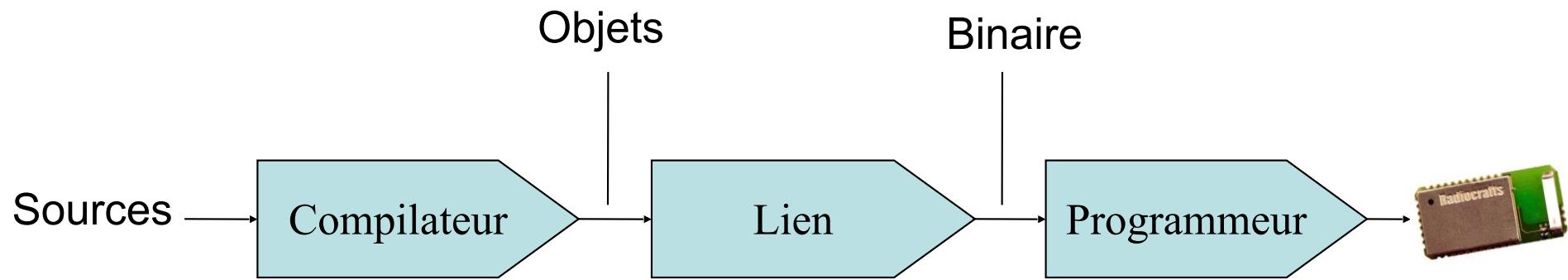


# Exemples de systèmes d'exploitation

- Exemples de systèmes d'exploitation embarqués
  - ContikiOS ([www.sics.se/~adam/contiki](http://www.sics.se/~adam/contiki))
  - FreeRTOS ([www.freertos.org](http://www.freertos.org))
  - TinyOS ([www.tinyos.org](http://www.tinyos.org))
  - Et des milliers d'autres...
- Pour des MCU plus performantes (i.e. ARM)
  - VX Works
  - Microcontroller Linux (Android, Maemo, etc.)
  - Windows CE
  - Symbian

# Développement embarqué

# Développement embarqué



# Développement embarqué

- Le logiciel réside dans la mémoire fixe du système
  - Mémoires flash/EEprom/prom externes ou internes
  - Les microcontrôleurs modernes sont capables d'écrire le temps d'exécution de la mémoire flash interne et n'ont en général aucun bus de mémoire
- Le développement est effectué en-dehors du système
- Les compilateurs croisés sont utilisés pour créer des fichiers binaires
  - Le compilateur croisé crée des fichiers binaires pour une architecture différente de celle où il fonctionne
  - Différents compilateurs gratuits ou payants sont disponibles
- Le système est programmé en chargeant les outils de programmation ou les flashers externes embarqués du système

# Environnements de compilateurs croisés

- Environnements de développement intégrés (IDE)
  - Les compilateurs payants sont en général de ce type
  - Ils dépendent d'un système d'exploitation spécifique (ex. Windows)
  - Ils intègrent un éditeur de textes, des outils de compilation et de gestion de projet, ainsi qu'une bibliothèque C
  - Un outil de programmation du système est en général intégré
  - Des IDE open-source sont aussi disponibles
    - Les IDE open-source utilisent en général une architecture “**plugin**”
    - Environnements extensibles pour les généralités
    - Incluent des outils d'écriture pour lancer n'importe quel outil de ligne de commande : compilateurs, liens, éditeurs et programmeurs externes
    - Exemple: Eclipse (développé sous Java)

# Environnements de compilateurs croisés

- Fonctionnalités de ligne de commande
  - Séparation compilateur/lien, séparation des outils de gestion des éditeurs et des projets, bibliothèque C dépendante de l'architecture
- Gestion de projet : **make**
  - make est un outil de création logicielle automatisée
  - Basé sur les blocs **cible-dépendance-opération**
  - Permet d'utiliser des modèles de projets et des règles de création de plateforme séparée en utilisant la fonction “inclure les fichiers”
  - Moyen le plus connu de gérer les projets logiciels open-source
  - Les outils **automake** et **autoconf** étendent les fonctionnalités au développement de logiciels indépendants de la plateforme

# Environnements de compilateurs croisés

- Compilateurs de lignes de commande
  - Le plus répandu est gcc : disponible pour une multitude d'architectures de micro-contrôleurs et de processeurs
  - Sdcc : Small Device C Compiler : PICs, 8051's etc.
  - Compilateurs à architecture unique
- Outils de programmation du système
  - En général spécifiques à une famille unique **de micro-contrôleurs**
  - Grande différence dans la simplicité d'utilisation et le type d'interface
  - La plupart requièrent un câble de programmation ou un matériel de programmation pour charger le logiciel
  - Dépendent de l'algorithme de programmation du micro-contrôleur
    - Bus standards (SPI, UART, JTAG) vs. bus déposés

# Difficultés des compilateurs croisés

- Portabilité
  - Les fichiers en-têtes ne portent pas forcément le nom standard
  - Les fichiers en-têtes spécifiques au matériel ne sont pas automatiquement sélectionnés
    - La plupart des IDE payants utilisent des noms différents pour chaque modèle de matériel différent -> difficultés de portabilité
    - gcc i.e. utilise des macros internes pour la sélection des modèles -> portabilité plus simple via les variables de l'environnement, pas de changement nécessaire des en-têtes
- Accès au registre du matériel et traitement des interruptions
  - La déclaration de traitement des interruptions dépend du compilateur
    - Le format de déclaration n'est pas standardisé
    - Peut être traitée via les macros ( dans la plupart des cas)
  - Certains compilateurs (et bibliothèques C) requièrent des macros E/S
    - Les ports gcc développent des modes d'accès direct au registre

# Outils open-source

- Différents éditeurs de textes disponibles : nedit, emacs, vi ...
- Système de création de projet : make
- Compilateurs/liens : binutils & gcc, sdcc
  - Binutils : as, ld, objcopy, size etc.
  - Gcc : c compiler; utilise binutils pour créer des fichiers binaires
- Bibliothèques C standards
  - Fournissent les en-têtes de développement nécessaires et les fichiers d'objets pour les liens et la cartographie de la mémoire
  - msp430-libc for MSP430, avr-libc for AVR
- Programmeurs
  - AVR : uisp, avrdude
  - MSP430 : msp430-bsl, msp430-jtag
  - CC2430 : nano\_programmer
- IDE : Eclipse

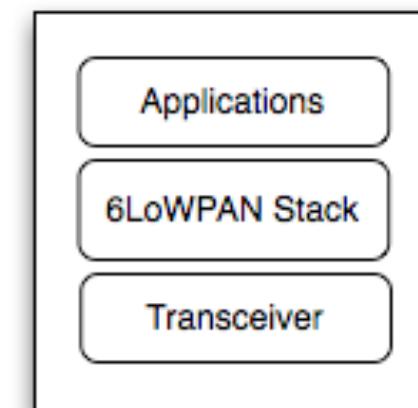
# Compilateur SDCC

- Compilateur C simple de matériel
- <http://www.sdcc.org>
- Spécialisé dans les micro-contrôleurs 8051, PIC, HC08 etc.
  - Supporte CC2430 et CC2510
- L'application sdcc traite à la fois la compilation et les liens
- Utilise un environnement make build
- Compatible avec Eclipse
- Support des opérations bancaires (nécessaire sur 8051 avec une mémoire de 64k+)
  - Merci à Peter Kuhar pour le support des opérations bancaires

# Difficultés d'implémentation 6LoWPAN

# Modèles de puces

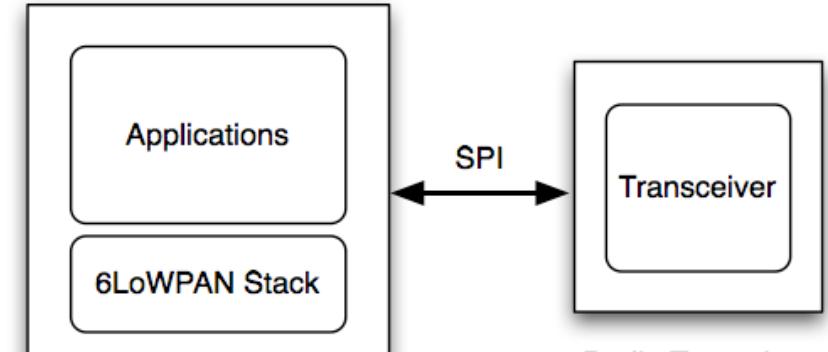
- Comment intégrer 6LoWPAN dans du matériel embarqué ?
- Défis :
  - Manque d'interfaces standards (pas d'USB ou de PCMCIA)
  - Pas de systèmes d'exploitation standards
  - Limitations dans la consommation d'énergie
  - Limitations en termes de prix
- Les modèles d'intégration 6LoWPAN incluent le processeur SoC, à deux puces ou réseau
- Modèle du système sur une puce
  - Tout sur la même puce
    - + intégration maximale
    - + taille et prix minimaux
    - Développement plus long et plus difficile
    - Mauvaise portabilité, voire absence de portabilité



System-on-a-Chip Radio

# Modèles de puces

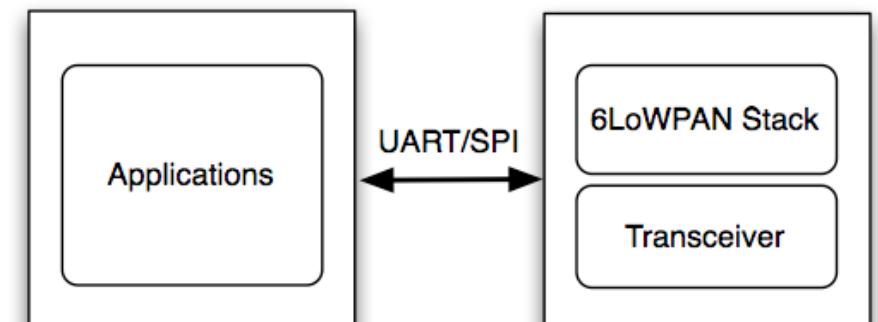
- Solution à deux puces
  - Emetteur-récepteur séparé
    - + libre choix d'uC
    - + meilleure portabilité
    - Plus chère
    - Intégration des applications avec le stack



Application Microcontroller

Radio Tranceiver

- Solution du processeur réseau
  - Stack réseau sur la radio
    - + Libre choix d'uC
    - + Applications indépendantes de la radio
    - Plus chère

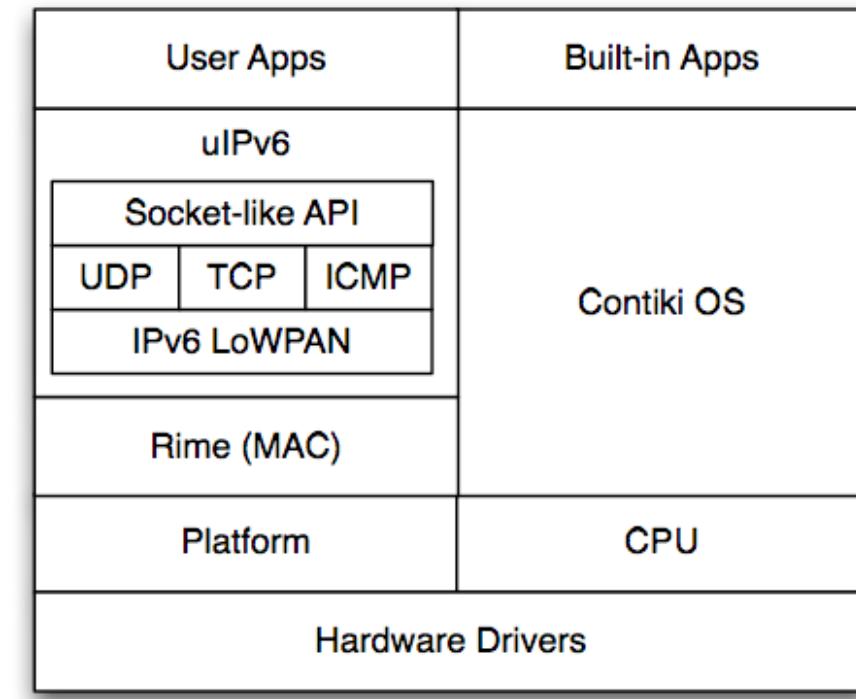


Application Microcontroller

Network Processor

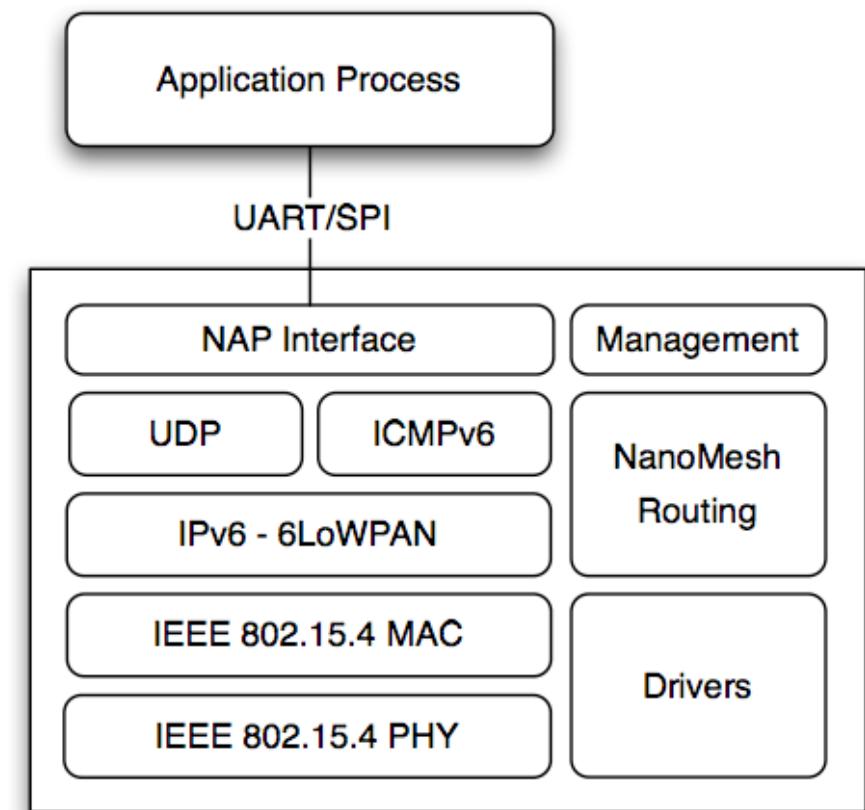
# Contiki uIPv6

- Système d'exploitation embarqué populaire pour les petits micro-contrôleurs
  - MSP430, AVR, PIC, 8051 etc.
- <http://www.sics.se/contiki>
- Standard basé sur C
- Applications portables
- Protothreads légers
- Stack uIPv6
  - Support IPv6 complet
  - RFC4944 + 6lowpan-hc
  - UDP, TCP, ICMPv6
- Idéal pour la recherche



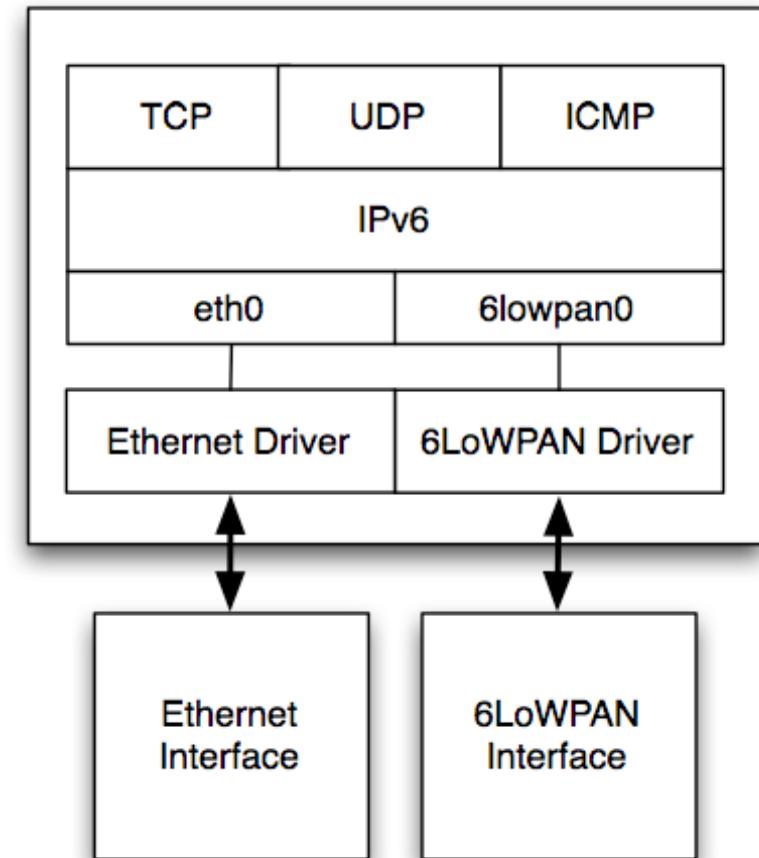
# NanoStack 2.0

- Stack 6LoWPAN de Sensinode
- Modèle de processeur réseau
  - Interface NAP sur UART
- Optimisé pour les radios SoC
  - TI CC2430, CC2530
  - TI CC1110
  - Portable
- Stack IPv6/6LoWPAN
  - UDP, ICMPv6
  - RFC4944, 6lowpan-hc
  - 6lowpan-nd
  - Routage IP NanoMesh



# Intégration des Routeurs

- Les routeurs de protection interconnectent le monde IPv6 et 6LoWPAN
- Un routeur ER doit développer :
  - Des interfaces 6LoWPAN
  - Une adaptation à 6LoWPAN
  - Une ND 6LoWPAN simple
  - Un protocole IPv6 complet
- On compte aussi d'autres fonctionnalités types :
  - Support et tunneling IPv4
  - Techniques de proxy pour les applications
  - Support LoWPAN étendu
  - A firewall
  - Management

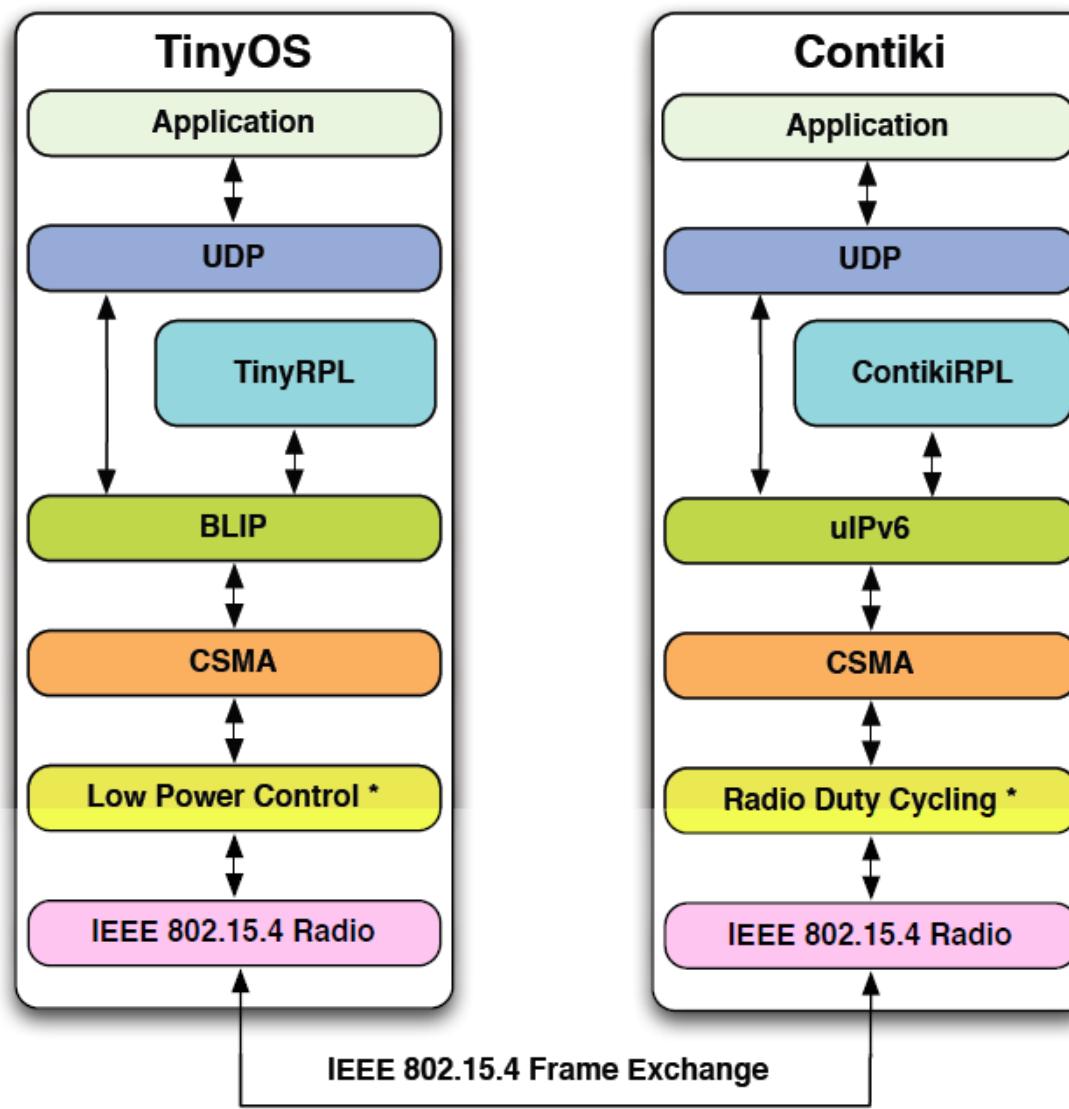


# Contiki

# Qu'est-ce que Contiki ?

- Contiki est un système d'exploitation open-source pour les systèmes embarqués
- Très bonne portabilité et raisonnablement compact
- Configuration du protocole personnalisable
- Crée par Adam Dunkels, développeur du stack uIP
- <http://www.sics.se/contiki>

# Contiki vs TinyOS



# Processus Contiki

- Contiki dépend des événements
  - Les interruptions et les pilotes HW génèrent des événements
  - Les événements sont diffusés aux agents d'événements par Contiki
  - Les agents d'événements doivent retourner le contrôle à Contiki le plus rapidement possible
  - Polyvalence coopérative
- Les processus basiques sont développés en utilisant les protothreads
  - La création d'opérations séquentielles est plus simple
  - Abstraction pour éviter la programmation complexe des états-machines
    - Dans les applications plus complexes, la quantité d'états peut être importante

# Modèles d'exécution Contiki

- Contiki propose plusieurs modèles d'exécution
- Protothreads : agents d'événements semblables au thread
  - Ils permettent de créer des structures semblables au thread sans stacks supplémentaires
  - Ils limitent la structure du processus : pas de structure changement/cas permise
  - Parfois ils n'utilisent pas les variables locales
- Modèle multi-threading disponible
  - Pour des systèmes plus performants
  - Permet la conception structurée des applications

# Processus Contiki : exemple

```
/* Declare the process */
PROCESS(hello_world_process, "Hello world");

/* Make the process start when the module is loaded */

AUTOSTART_PROCESSES(&hello_world_process);

/* Define the process code */

PROCESS_THREAD(hello_world_process, ev, data) {

    PROCESS_BEGIN();                  /* Must always come first */

    printf("Hello, world!\n");        /* Initialization code goes here */

    while(1) {                      /* Loop for ever */

        PROCESS_WAIT_EVENT();         /* Wait for something to happen */

    }

    PROCESS_END();                   /* Must always come last */
}
```

# Processus Contiki : Notes

- Un processus n'utilise pas toujours des constructions changement-cas (switch-case)
  - Limite du modèle protothread
  - Les structures et changements d'état complexes devraient être des sous-routines
- Un processus peut ne pas utiliser les variables locales
  - Les variables perdront leurs valeurs lors d'un appel en attente pour un événement
  - Toutes les variables requises par le processus principal doivent être statiques
- Effets sur la conception de l'application
  - Le principal thread du processus ne devrait pas seulement contenir des séquences entre les attentes d'événements
  - Toutes les opérations devraient être traitées en tant que sous-routines

# Evénements Contiki

- `process_post(&process, eventno, evdata);`
  - Le processus sera invoqué plus tard
- `process_post_sync(&process, evno, evdata);`
  - Le processus sera invoqué immédiatement
  - Ne doit pas être appelé par un pilote matériel
- `process_poll(&process);`
  - Envoie un événement PROCESS\_EVENT\_POLL au processus
  - Peut être appelé à partir d'un pilote
- Utilisation des événements

```
PROCESS_THREAD(rf_test_process, ev, data) {  
    while(1) {  
        PROCESS_WAIT_EVENT();  
        if (ev == EVENT_PRINT) printf("%s", data);  
    }  
}
```

# Programmateurs Contiki

- Contiki a deux types de programmateurs principaux : etimer et rtimer
- Etimer : génère les événements programmés

Declarations:

```
static struct etimer et;
```

In main process:

```
while(1) {  
    etimer_set(&et, CLOCK_SECOND);  
    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));  
    etimer_reset(&et);  
}
```

- Rtimer : utilise la fonction callback
  - Fonction callback exécutée après une durée spécifiée

```
rtimer_set(&rt, time, 0, &callback_function, void *argument);
```

# Stacks du Protocole Contiki

- Contiki a deux stacks de protocole différents : uIP et Rime
- uIP fournit un stack TCP/IP complet
  - Pour les interfaces qui permettent l'overhead du protocole
  - Matériel Ethernet
  - Ligne de série IP
  - Inclue le support IPv4 et IPv6/6LoWPAN
- Rime fournit un support header compressé
  - L'application peut n'utiliser que la couche MAC
- Les stacks de protocole peuvent être interconnectés
  - Les données uIP peuvent être transmises sur Rime et vice versa

# Le stack du protocole Rime

- Des modules séparés pour l'analyse du protocole et les machines états
  - Rime contient les modules d'opération du protocole
  - Chameleon contient les modules d'analyse du protocole
- Mise en route de Rime : exemple
  - Configurer Rime pour utiliser sicslowmac sur cc2430 rf
  - La mise en route est effectuée par la fonction principale de la plateforme : platform/sensinode/contiki-sensinode-main.c

```
rime_init(sicslowmac_init(&cc2430_rf_driver));  
  
set_rime_addr(); //this function reads MAC from flash and places  
//it to Rime address
```

# Rime : Réception

- Configurer la réception Rime : diffusion
  - Configurer une fonction callback

Declarations:

```
static struct broadcast_conn bc;
static const struct broadcast_callbacks broadcast_callbacks = {recv_bc};
```

The callback definition:

```
static void
recv_bc(struct broadcast_conn *c, rimeaddr_t *from);
```

In main process:

```
broadcast_open(&bc, 128, &broadcast_callbacks);
```

- La réception Unicast est similaire

# Rime : Envoi

- Envoi de données de diffusion en utilisant Rime

Declarations:

```
static struct broadcast_conn bc;
```

In main process:

```
packetbuf_copyfrom("Hello everyone", 14);  
broadcast_send(&bc);
```

- Envoi de données unicast en utilisant Rime

Declarations:

```
static struct unicast_conn uc;
```

In your function:

```
rimeaddr_t *addr;  
addr.u8[0] = first_address_byte;  
addr.u8[1] = second_address_byte;  
packetbuf_copyfrom("Hello you", 9);  
unicast_send(&uc, &addr);
```

# Création de Ports Contiki

- Première étape : vérifiez si votre unité centrale a déjà un code
  - Si oui, configurez votre plateforme pour l'utiliser
  - Si non, consultez d'autres répertoires d'unités centrales pour voir les modèles d'implémentation
- Deuxième étape : vérifiez si votre matériel est compatible avec les autres plateformes
  - Si oui, copiez le code de la plateforme qui sert d'exemple puis modifiez-le
  - Si non, consultez les autres plateformes pour un modèle minimal
- Création d'une application test
  - Commencez avec les LED dans platform/myplatform/contiki-myplatform-main.c
  - Utilisez-les pour les boucles pour être sûr que votre compilateur fonctionne
  - Poursuivez en ajoutant printf's pour voir si votre UART fonctionne
- Première application réelle
  - Créez un timer pour votre processus de test : LED flash, print info
  - Effectuez différentes pauses pour voir si vos horloges sont correctes
- Ajoutez plus de pilotes et testez-les

# Simulateur COOJA pour Contiki

# Simulateur COOJA pour Contiki

- COOJA est l'acronyme de **Contiki OS Java Simulator**
- Simulateur pour l'OS Contiki
- Cooja est développé avec Java pour simuler des applications sur Contiki
- Il supporte les simulations inter-couches (*cross-level*)

