CASAN: A New Communication Architecture for Sensors Based on CoAP

Pierre David
pda@unistra.fr

Philippe Pittoli
p.pittoli@unistra.fr

Thomas Noël
noel@unistra.fr

Laboratoire ICube – Université de Strasbourg – France

16 march 2016

Outline

Introduction

The CoAP protocol

The CASAN architecture

Conclusion

Outline

Introduction

The CoAP protocol

The CASAN architecture

Conclusion

Introduction – Problem statement

- have multiple sensors
  manufacturing costs $\Rightarrow$ hardware, software
  operating costs $\Rightarrow$ ease of use, scalability
  maintenance costs $\Rightarrow$ reliability
- adapt IoT to industrial specific constrains
  - avoid electromagnetic interferences
- new features
  - geolocation of industrial components and products

Introduction – Context

Lack of standard for IoT?

- industry uses most often proprietary protocols and low-end technologies

Software world:

- trend: service-oriented architectures
- REST (Representational State Transfer) architecture
  - Generalization of REST using HTTP protocol

IETF answer: CoRE working group $\Rightarrow$ CoAP protocol

Outline

Introduction

The CoAP protocol

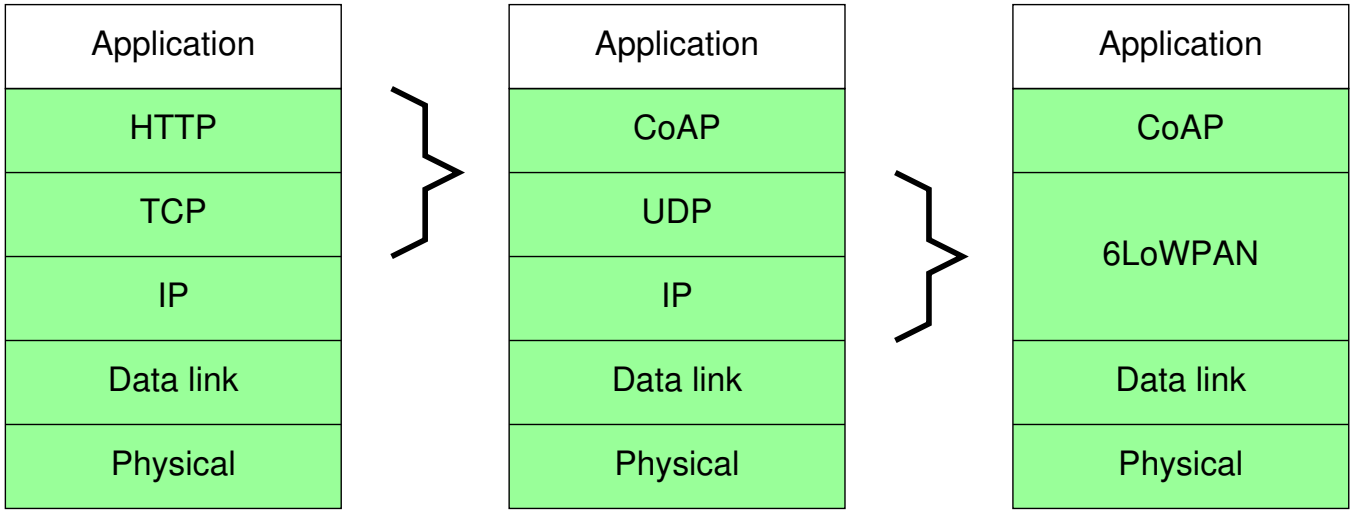
The CASAN architecture

Conclusion

CoAP – Overview

IETF CoRE working group:

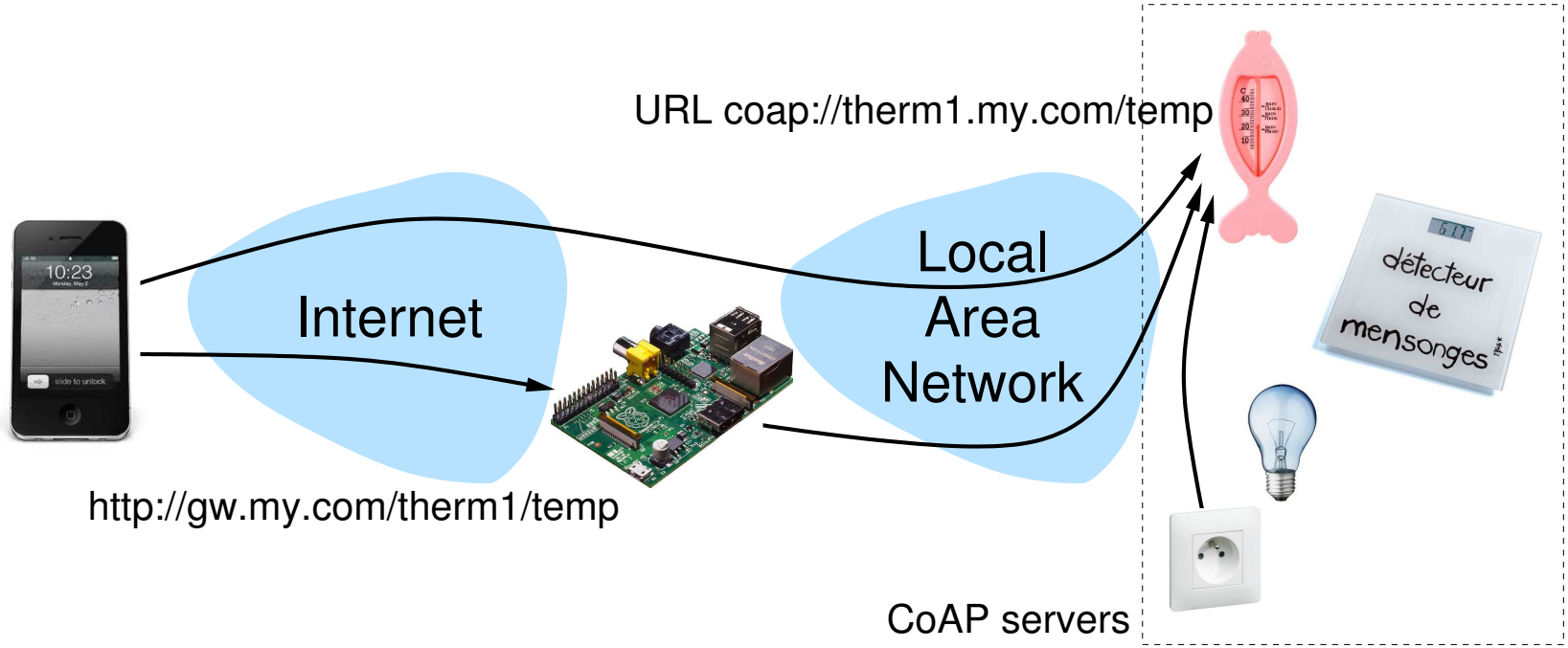
- CoAP: Constrained Application Protocol
- RFC 7252: June 2014
- In short: CoAP is HTTP, binary encoded, over UDP



Ex. Data Link Target: IEEE 802.15.4 (127 bytes payload)

CoAP – Overview

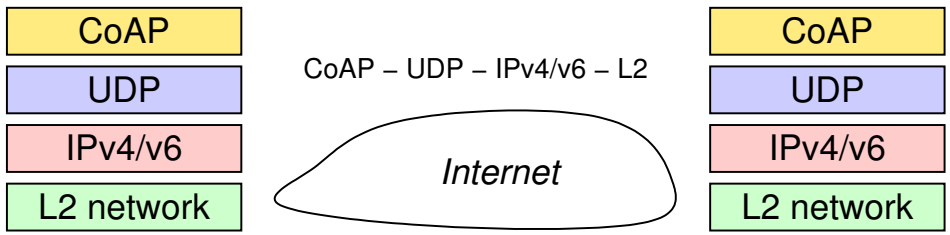
Architectural view:



"IP everywhere" dogma: each sensor must be IP addressable

CoAP – Example use cases

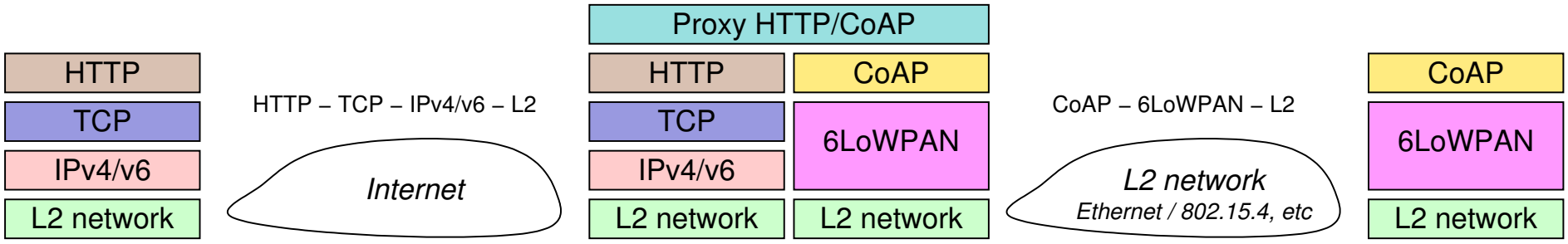
End-to-end CoAP



- end-to-end scenario limited by middle-boxes
- application must handle MTU of sensor network
- limited to specific perimeters:
  - local networks (car networks, smart buildings, etc.)

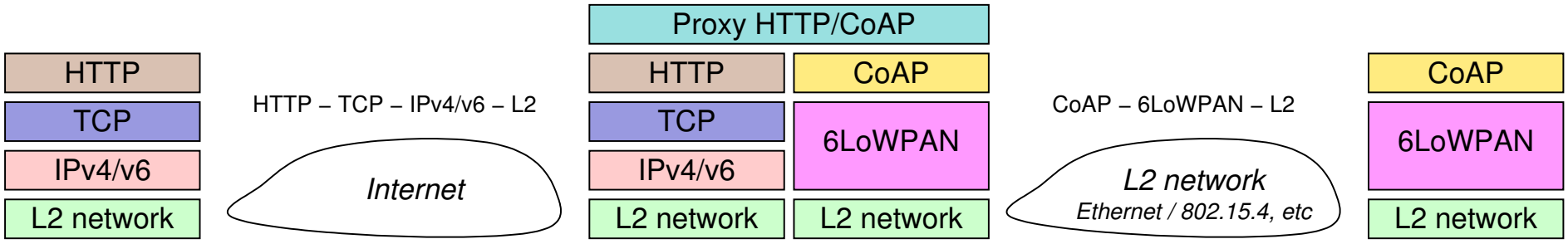
CoAP – Example use cases

HTTP-CoAP proxy with 6LoWPAN



CoAP – Example use cases

HTTP-CoAP proxy with 6LoWPAN



- proxy may handle the MTU difference
- rely on IPv6 deployment on the sensor network
- added complexity on sensors
- CoAP is not used as an end-to-end protocol
  - it is not the expected universal IoT protocol

Is CoAP a good solution?

- CoAP is a carefully designed protocol
- CoAP implementations are compact
  - libcoap : 10,000 lines of C, 2,800 semi-columns
  - contiki : 4,000 lines of C, 1,100 semi-columns
- In practice, CoAP is restricted to a local perimeter
  - no end-to-end paradigm
- CoAP itself is not sufficient. Sensors must embed:
  - IPv4, IPv6, UDP, 6LoWPAN, multicast
  - address protocols (DHCP, SLAC, ARP, IPv6-ND, etc.)
  - discovery protocols (DNS-SD, mDNS, etc.)

Outline

Introduction

The CoAP protocol

The CASAN architecture

Conclusion

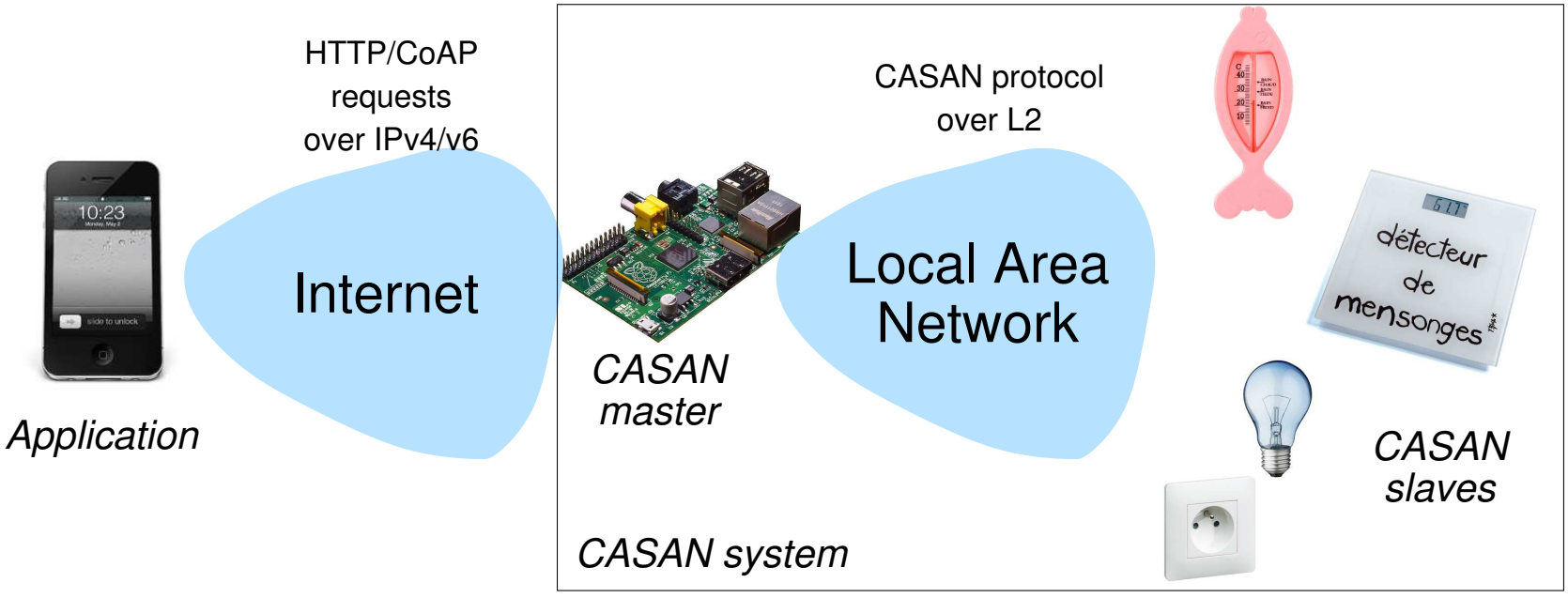
CASAN architecture

Common **A**rchitecture for **S**ensor and **A**ctuator **N**etworks

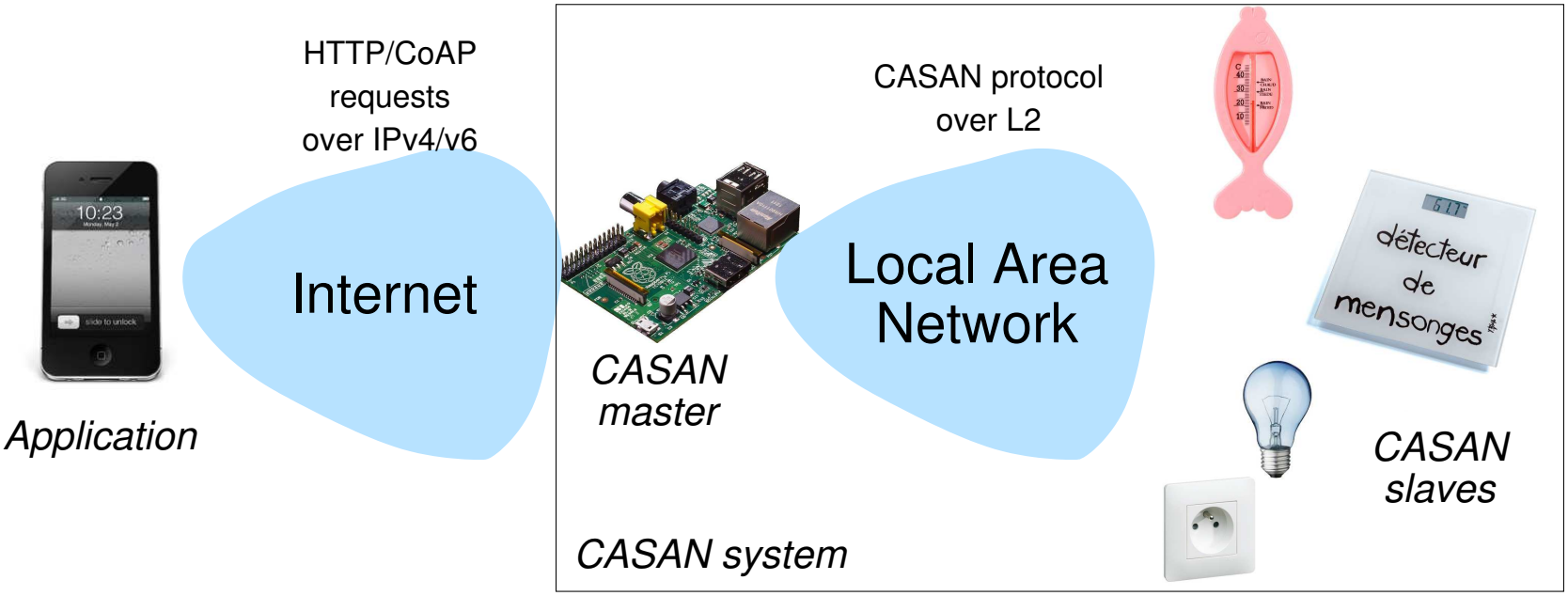
Goal: reduce sensor complexity as much as possible

- Example: ATmega328 (8-bits, 16 MHz, 2 KB SRAM)
- Remove all unnecessary protocols
  - Remove IP, UDP, 6LoWPAN
  - Use CoAP directly on any L2 layer
    $\Rightarrow$ Ethernet, IEEE 802.15.4 with 127 bytes of MTU, etc.
- Shift complexity to a more capable device
  - CoAP most realistic use-case: needs a gateway
  - Example: Raspberry PI, with a general purpose OS
  - Master of the CASAN domain (sensors = slaves)

CASAN architecture

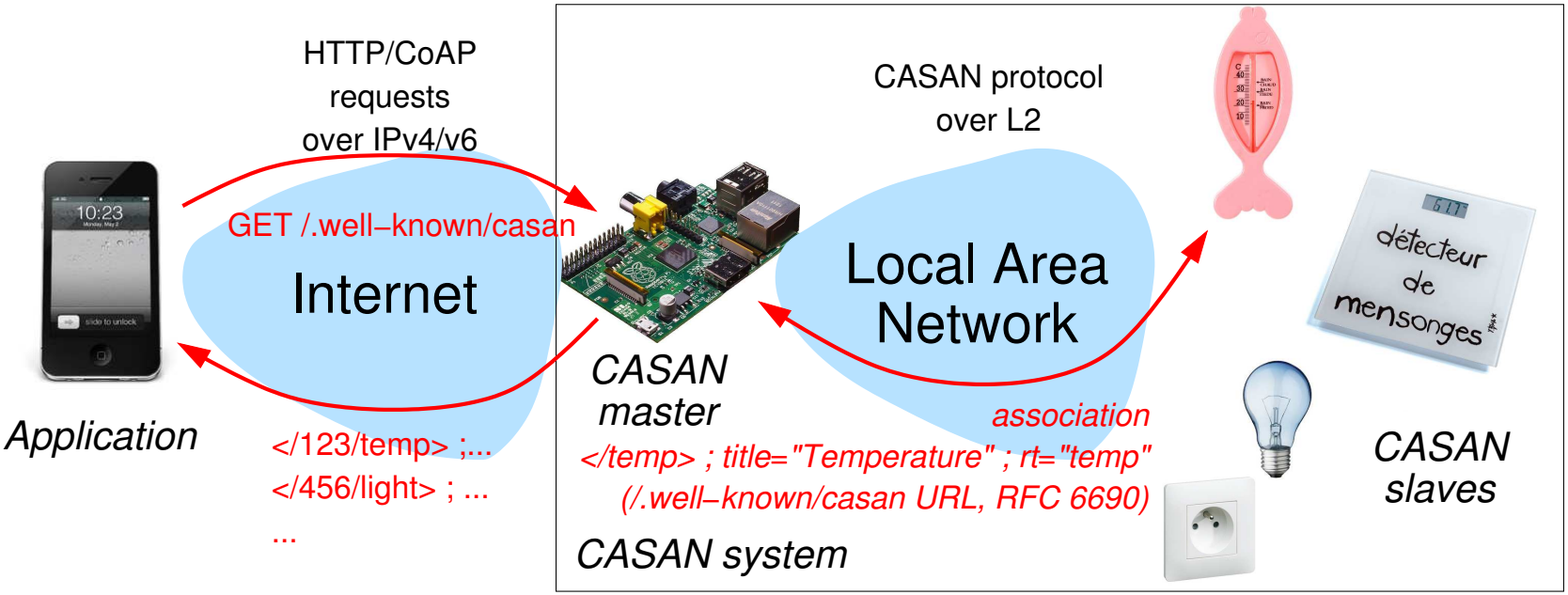


CASAN architecture



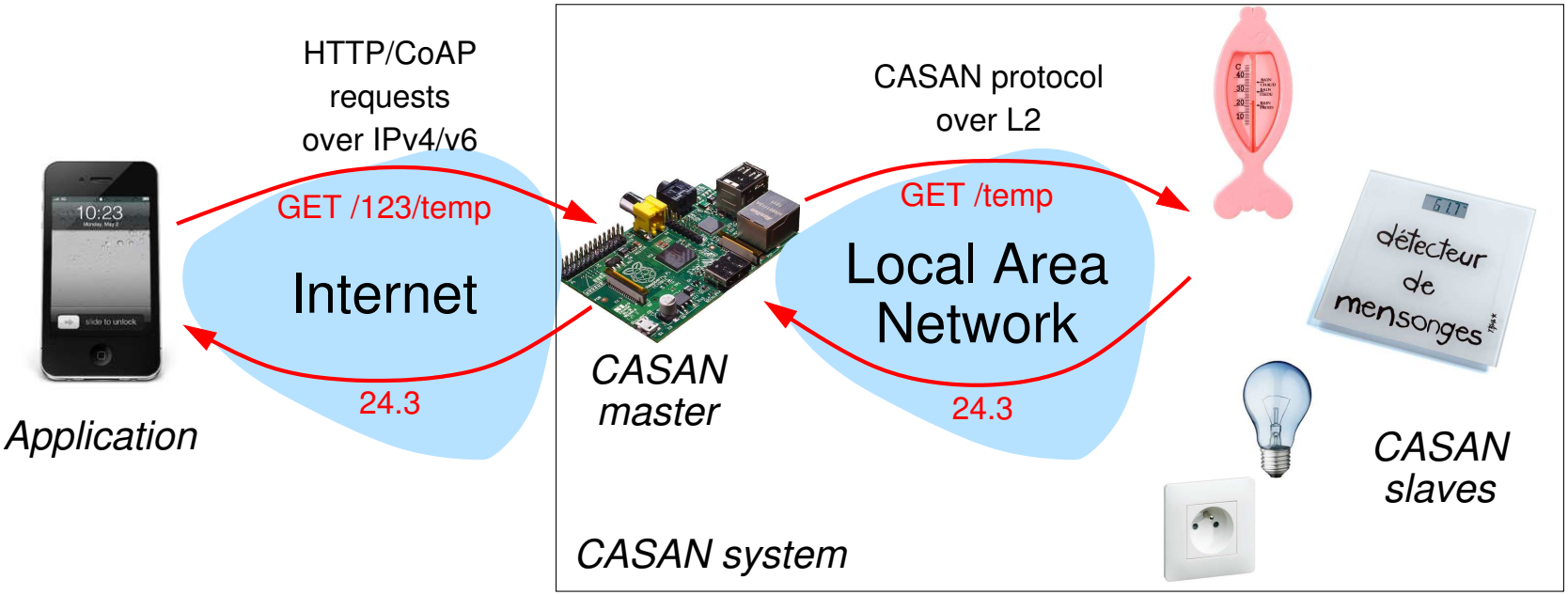
1. Initial pairing: done once

CASAN architecture



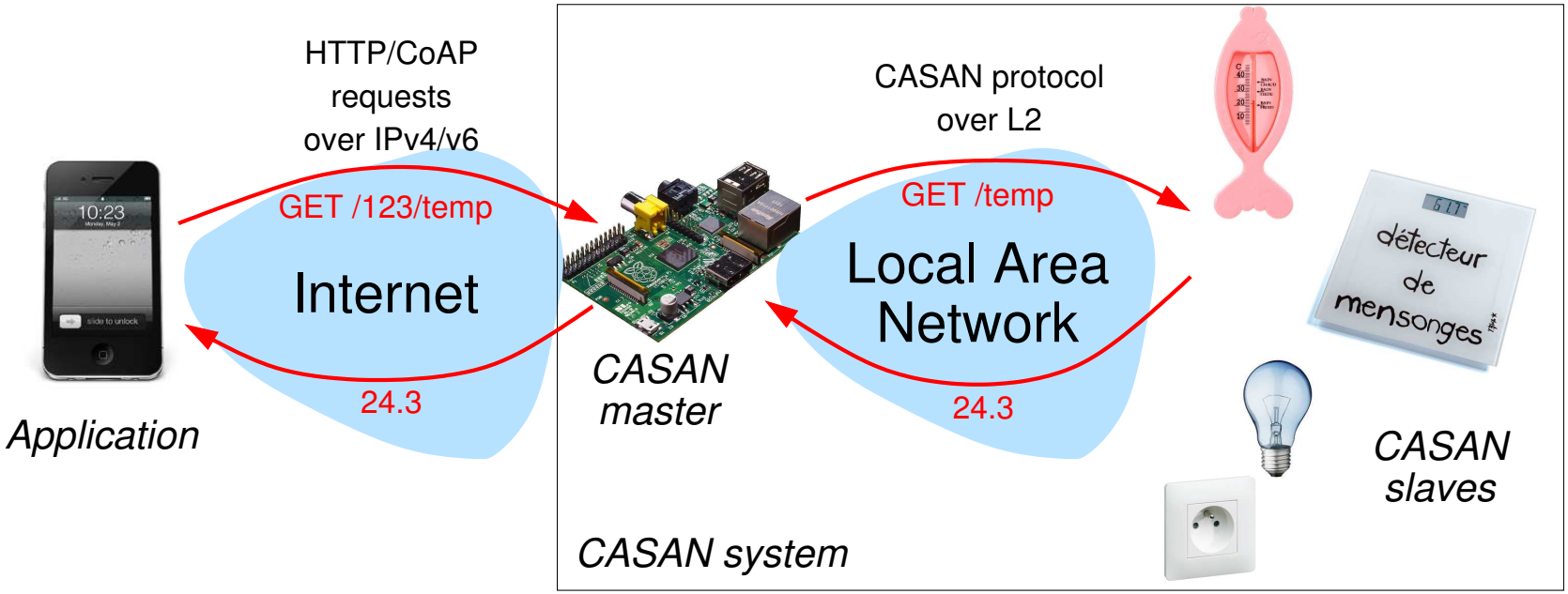
1. Initial pairing: done once
2. Tight coupling (association)

CASAN architecture



1. Initial pairing: done once
2. Tight coupling (association)
3. REST-routing

CASAN architecture

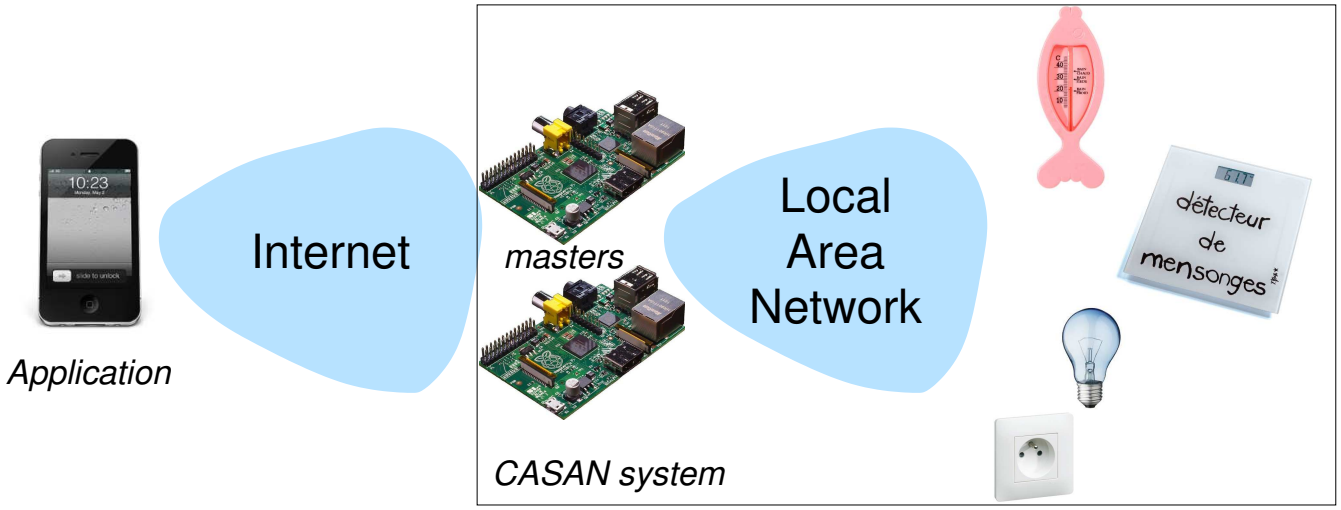


1. Initial pairing: done once
2. Tight coupling (association)
3. REST-routing

All CASAN messages are CoAP ones directly over L2

CASAN architecture – reliability

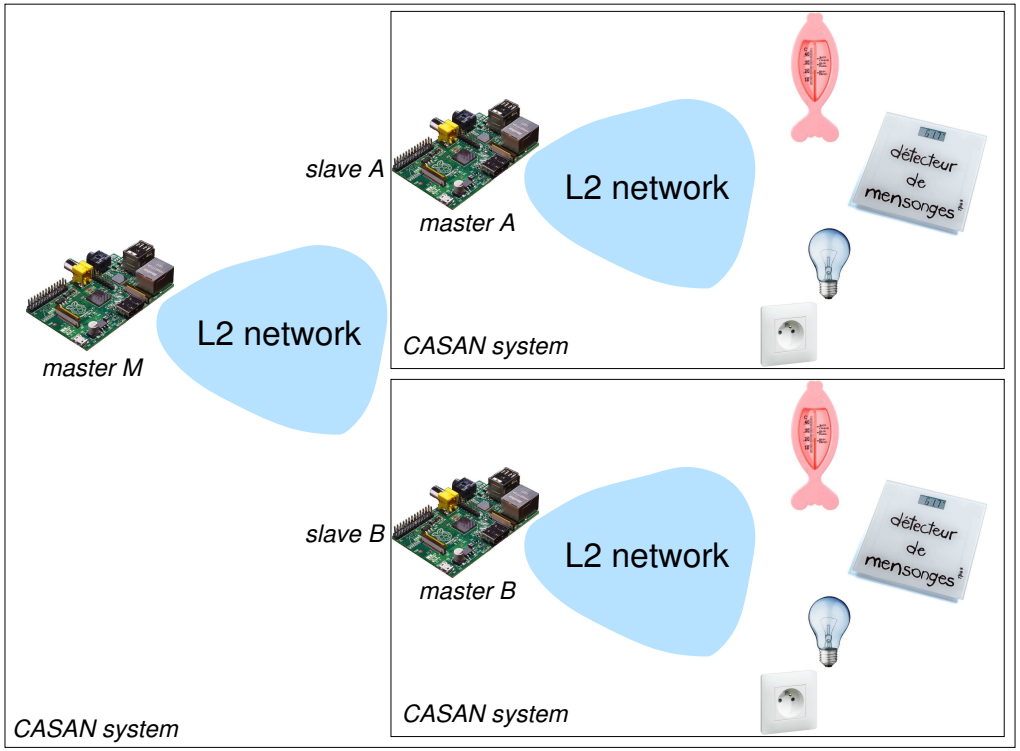
Single point of failure ⇒ master redundancy



- Masters are linked with a redundancy protocol (VRRP, CARP, HSRP, etc.)
- Each sensor associates with all masters

CASAN architecture – scalability

CASAN is not limited to a L2 network perimeter:



- A master may be a slave in another CASAN system
- Slaves advertise their resources (`/.well-known/casan`)
- REST-level routing

CASAN – Implementation

Prototype:

- Master: C++ (2011) on Linux
- Slave:
  - Arduino ATmega328 (8-bits, 16 MHz, 2 KB SRAM) + Wiznet W5100 Ethernet shield
  - Zigduino ATmega128RFA1 with on-chip 802.15.4

Comparison:

- not on performances (CASAN = CoAP over L2)
- on code complexity (code size) and functionalities
  - Contiki CoAP (general purpose implementation)
  - Pico-IPv6 (highly specialized stack / Arduino-XBee)

CASAN – Evaluation

	Contiki	Pico-IPv6	CASAN
Sensor code size			
Software maintainability			
Sensor hardware cost			
Interoperability			
Internet support			
L2 independence			
Service discovery			
Scalability			
Access control			

Outline

Introduction

The CoAP protocol

The CASAN architecture

Conclusion

Conclusion

- CoAP is a good protocol, but:
  - in the real world, its use is restricted to local networks
  - CoAP-enabled sensors are complex (IPv*, 6LoWPAN, ARP/ND, DHCP/SLAC, DNS-SD, etc.)
- The CASAN system is reducing sensor complexity without losing functionalities
  - code size similar to a highly specialized one
  - no loss in functionalities compared to a general system such as Contiki
  - added functionalities such as master replication
  - not restricted to local networks (scalability)

Conclusion

CASAN system is a great base for 4P factories

- masters can share data, logic, be participative
- anomalies can be detected locally
  we have global awareness
- and a lot more thanks to the high proximity of masters and slaves

Conclusion

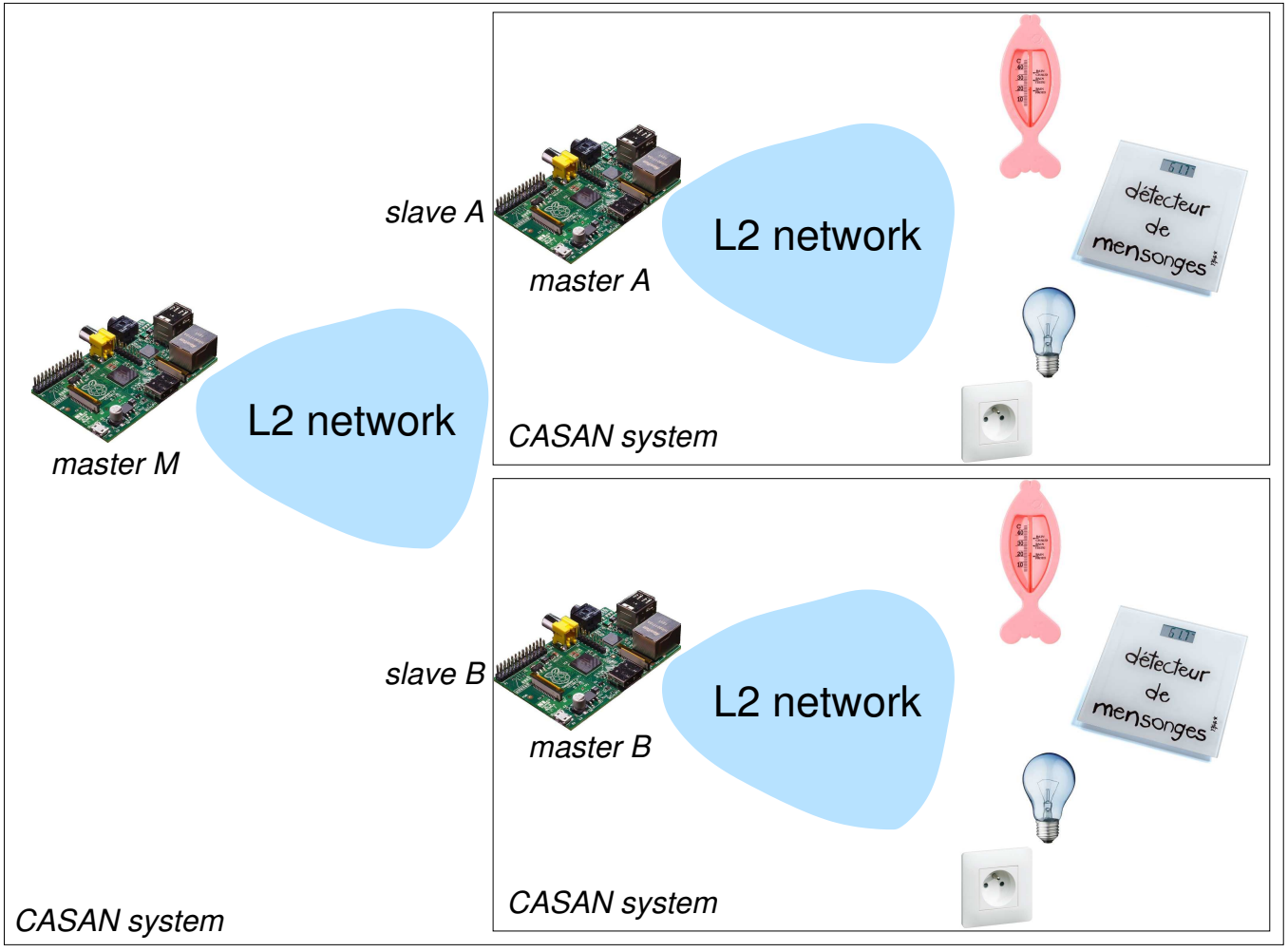
CASAN system is a great base for 4P factories

- masters can share data, logic, be participative
- anomalies can be detected locally
  we have global awareness
- and a lot more thanks to the high proximity of masters and slaves

Future works:

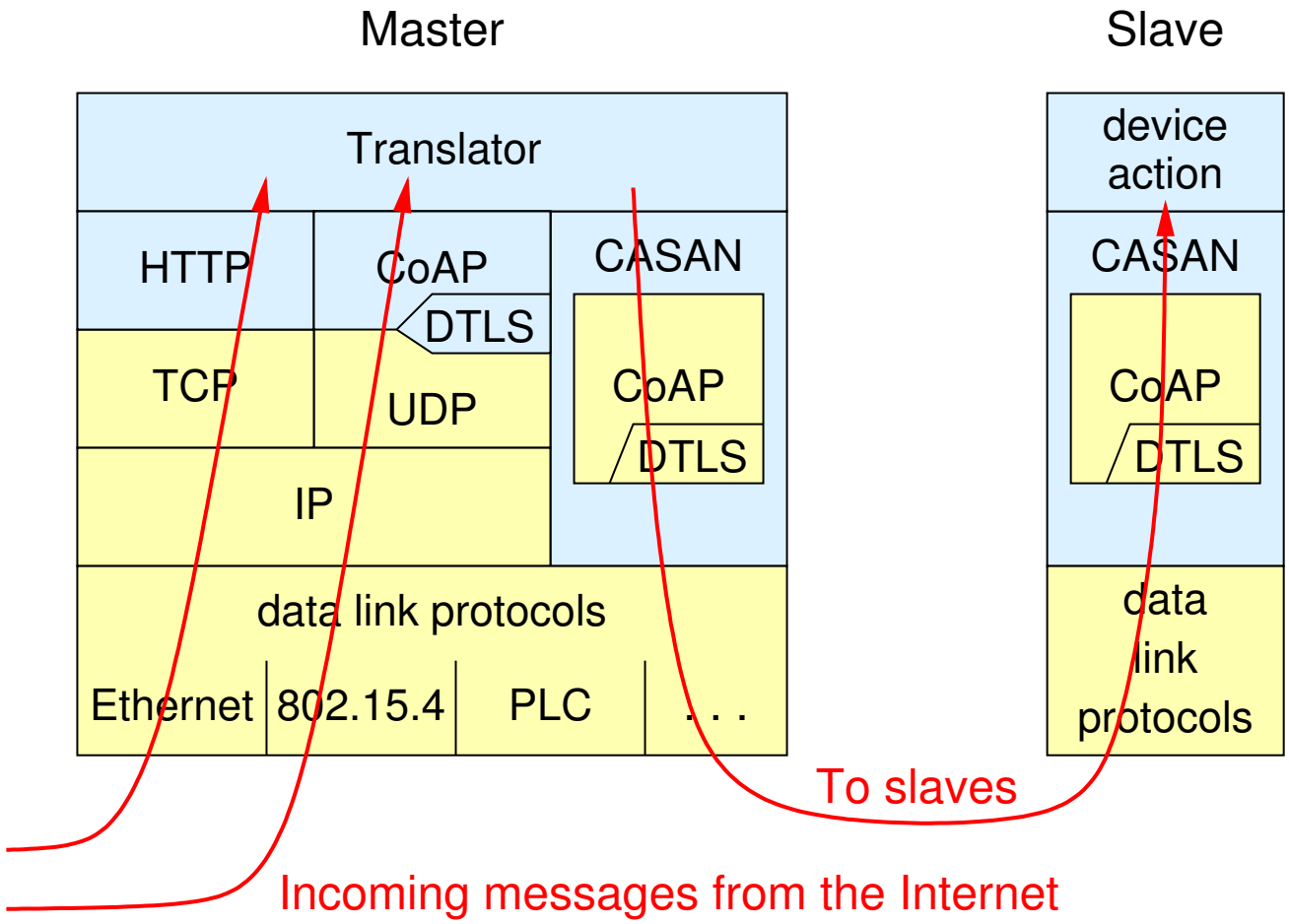
- security layer
- smart-building deployment

Questions?


slave A
master A
L2 network
détecteur de mensonges
master M
L2 network
CASAN system
slave B
master B
L2 network
détecteur de mensonges
CASAN system
CASAN system

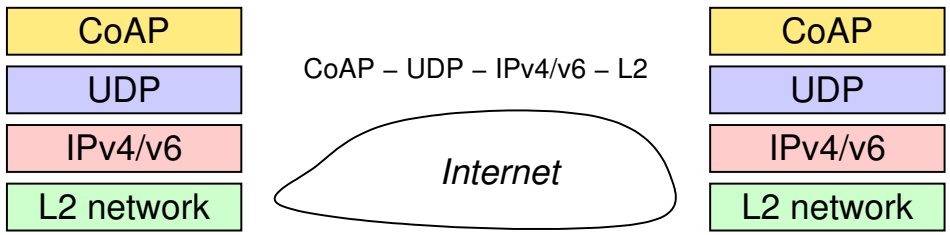
Additional slides

CASAN – Protocol stack



CoAP – Typical use cases (1/4)

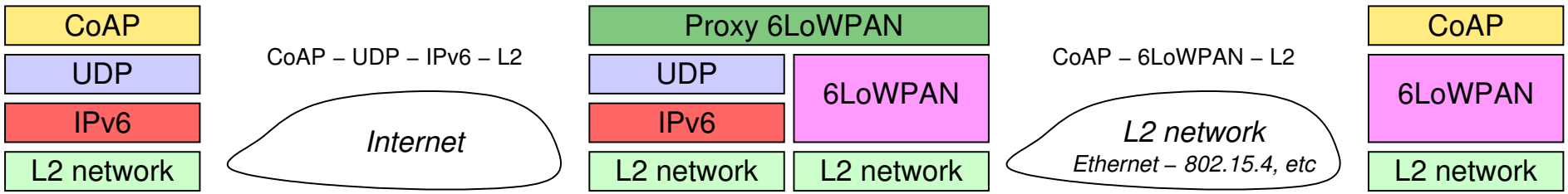
End-to-end CoAP



- end-to-end scenario limited by middle-boxes
- application must handle MTU of sensor network
- limited to specific perimeters:
  - local networks (car networks, smart buildings, etc.)

CoAP – Typical use cases (2/4)

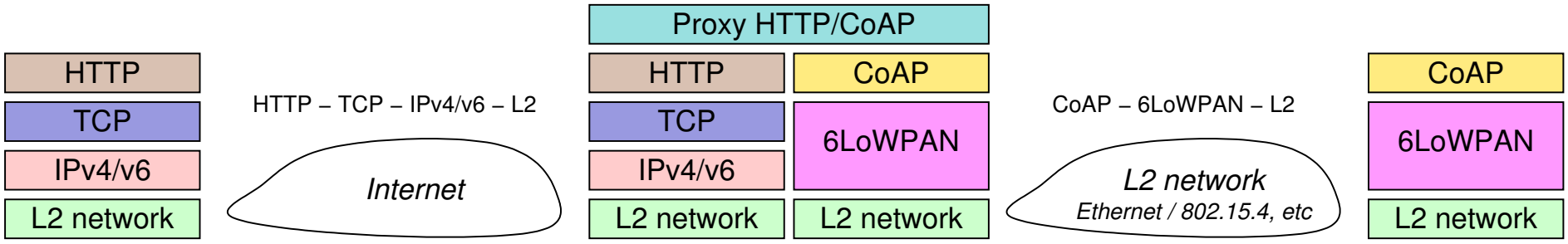
6LoWPAN compression



- 6LoWPAN: message size + IPv6 overhead
- rely on IPv6 deployment
- added complexity on sensors:
  - handle compression + process IPv6 packets

CoAP – Typical use cases (3/4)

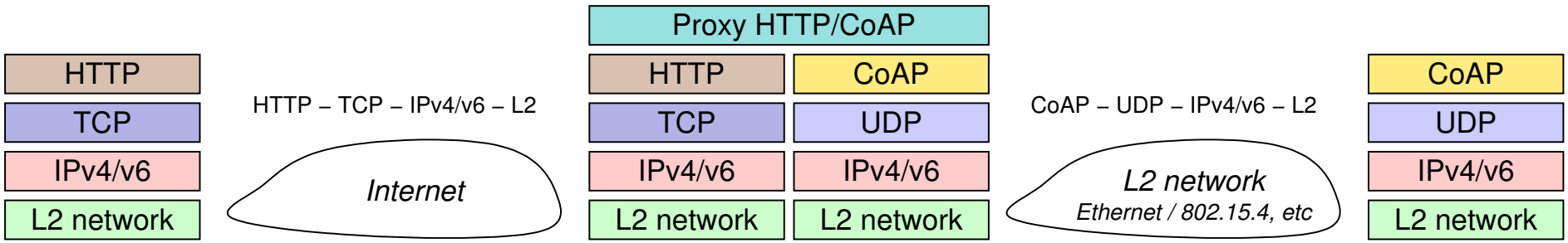
HTTP-CoAP proxy with 6LoWPAN



- proxy may handle the MTU difference
- rely on IPv6 deployment on the sensor network
- added complexity on sensors
- CoAP is not used as an end-to-end protocol
  - it is not the expected universal IoT protocol

CoAP – Typical use cases (4/4)

HTTP-CoAP proxy without 6LoWPAN



- more realistic use-case (with IPv4)
- proxy may handle the MTU difference
- CoAP is not the expected universal IoT protocol
  - end-to-end paradigm is broken

CASAN – Protocol stack