Your Connection to ICT Research

Internet des objets et Contiki

Jeudis du Libre, UMons

FEDER

UNION EUROPEENNE

Wallonie

LE FONDS EUROPEEN DE DEVELOPPEMENT REGIONAL
ET LA WALLONIE INVESTISSENT DANS VOTRE AVENIR.

Sébastien DAWANS
Laurent DERU
17/10/2013

Embedded &
Communication Systems

Software & System Engineering

Software & Services Technologies

Menu du jour

Définition de l'Internet des Objets

Standards de l'IoT

Contiki

Activités du CETIC

Internet Of Things

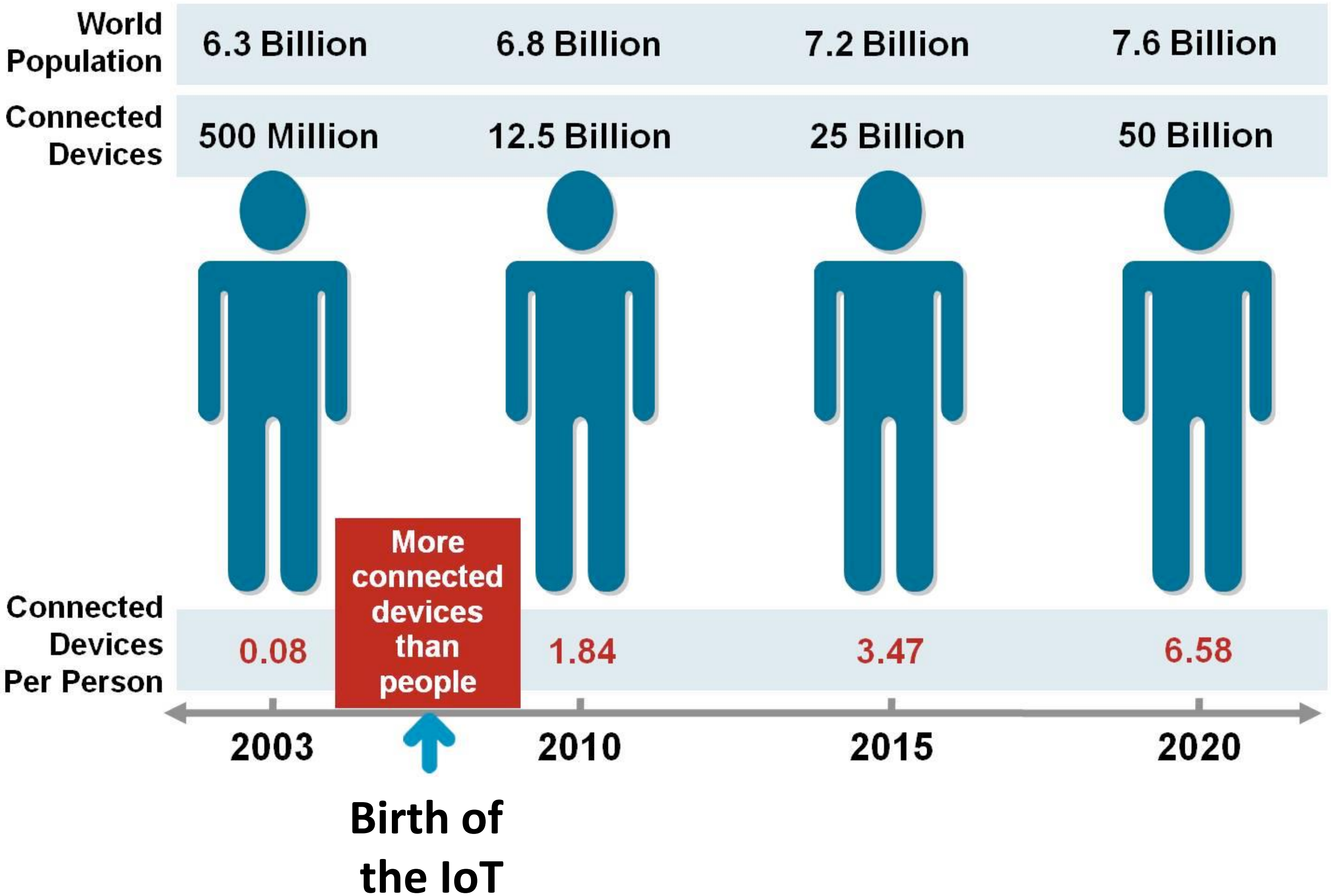
1999
RFID

200x
M2M

201x
Smartphones, RFID, Cloud…

Junction of the physical and digital worlds

Cisco's Vision

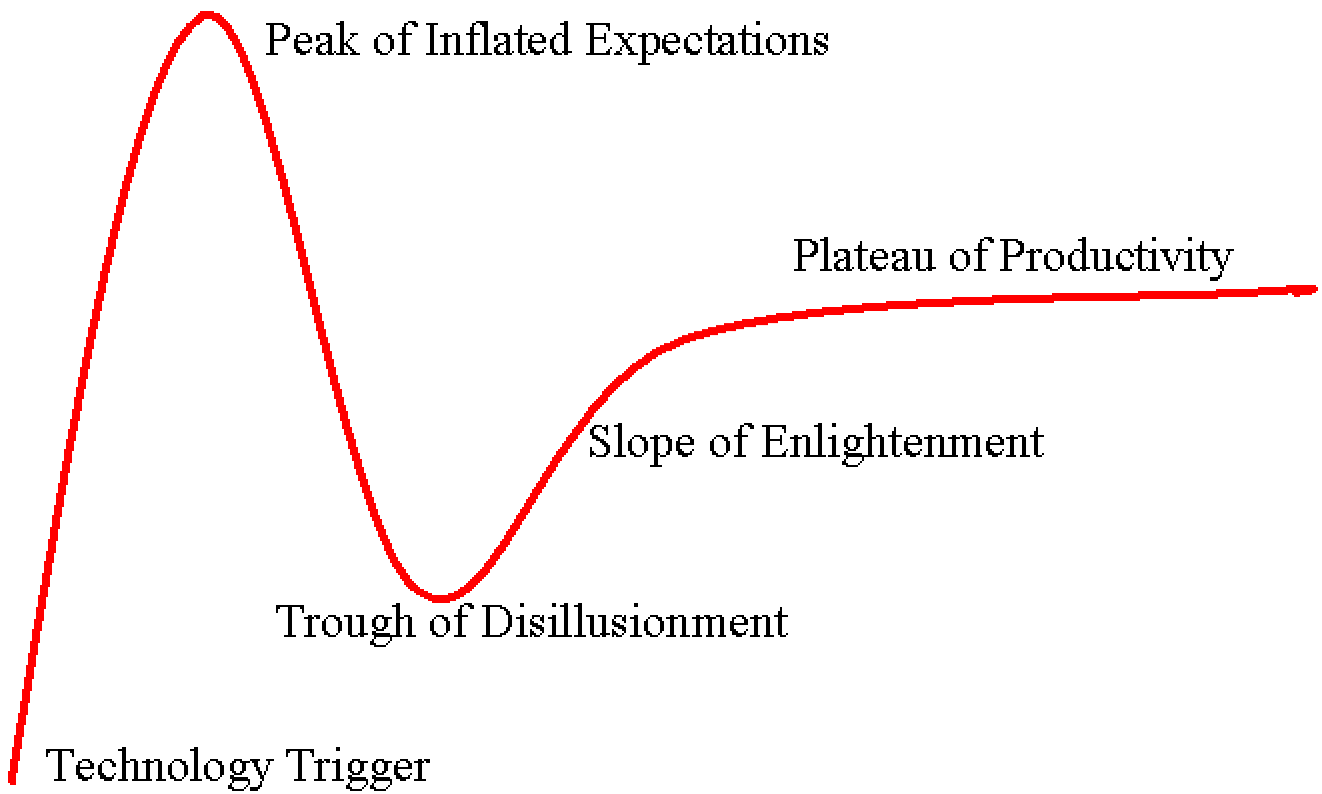


Source: Cisco IBSG, April 2011

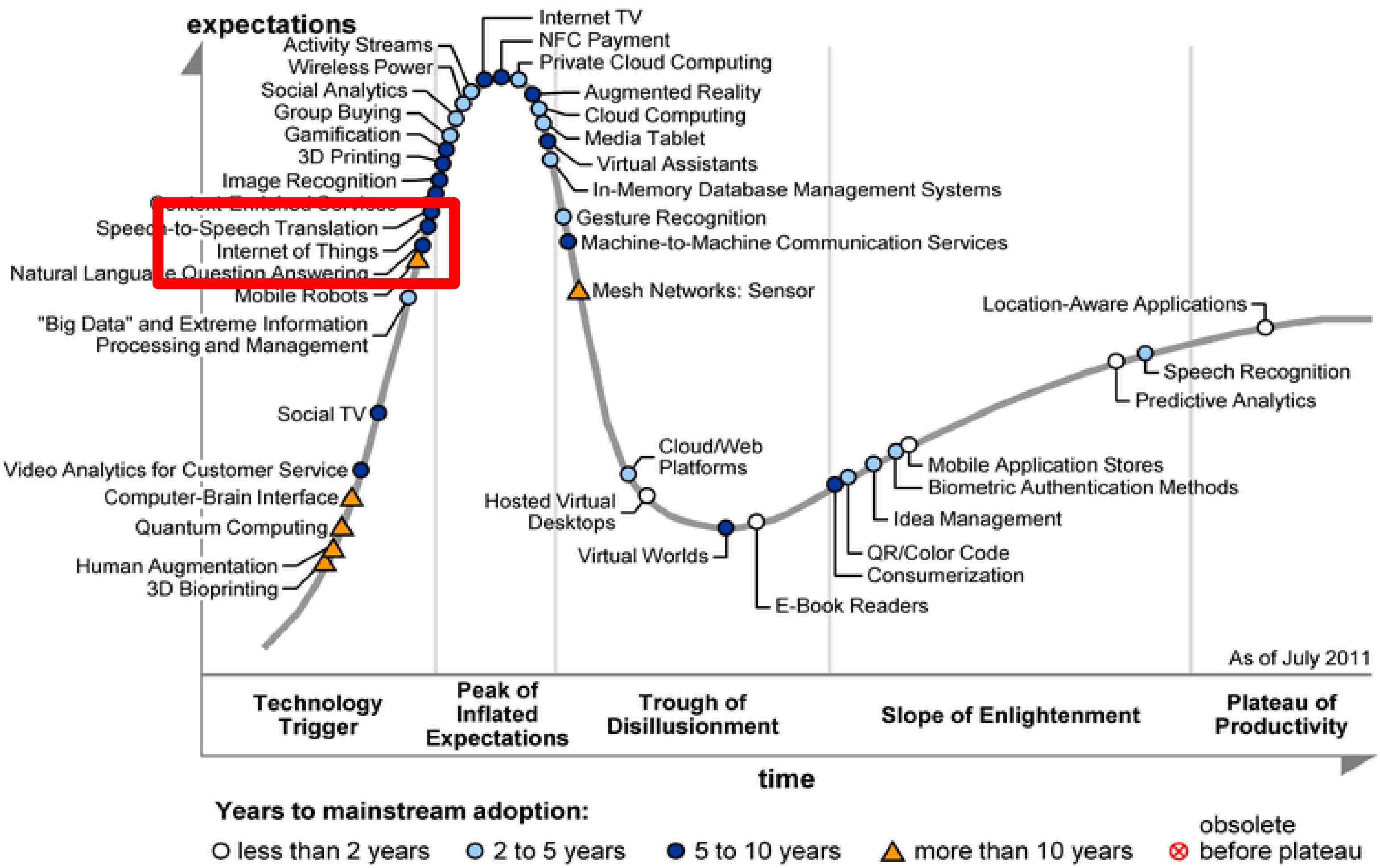
« the * syndrome »

Any*	Smart*	*aware
AnyOne	SmartHome	Context-Aware
AnyThing	SmartBuilding	Self-Aware
AnyWhere	SmartEnergy	Process-Aware
AnyBusiness	SmartGrid	… ?
AnyNetwork	SmartCity	
AnyContext	SmartFactory	
	SmartAgriculture	

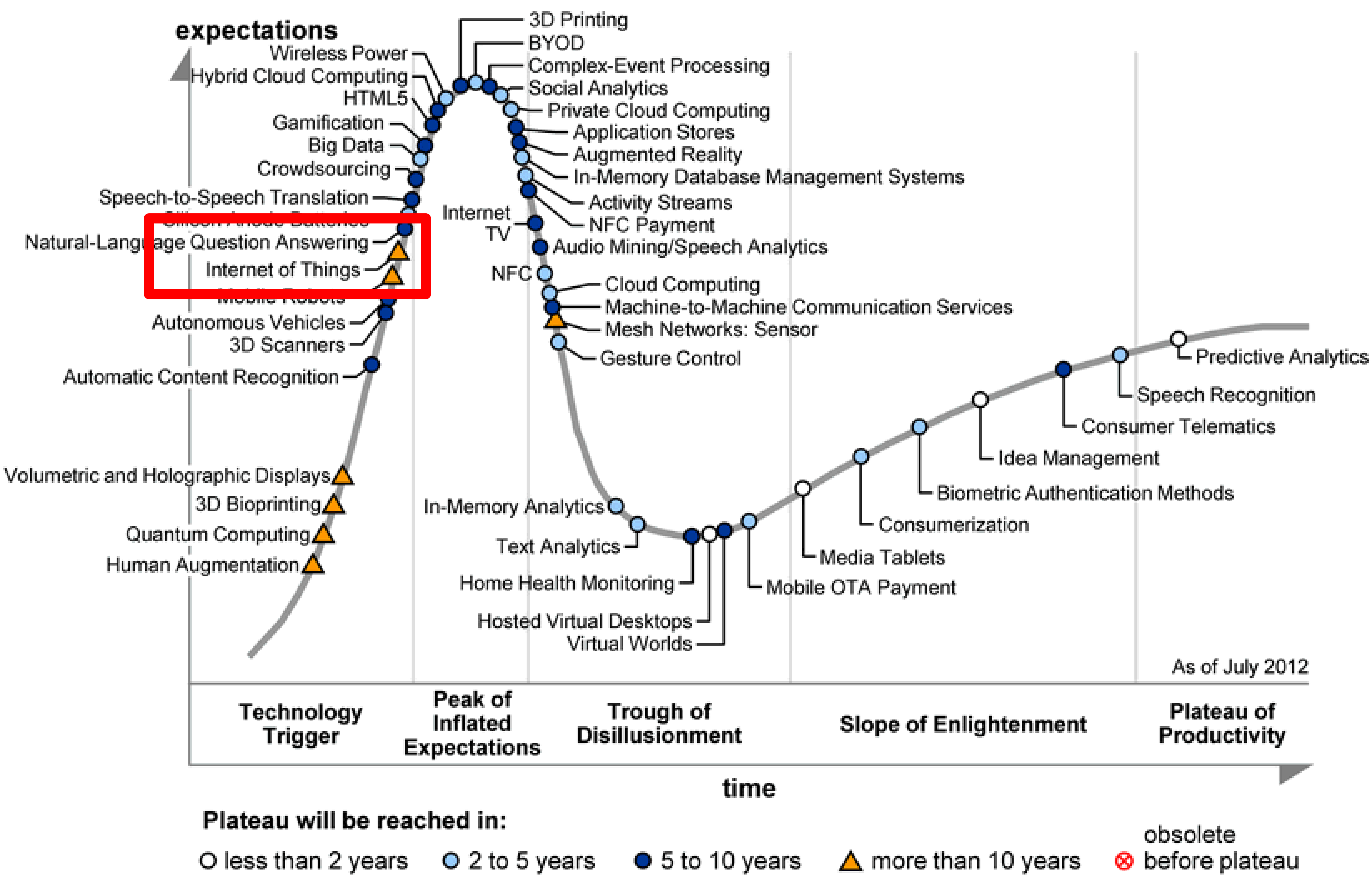
Gartner Hype Cycle



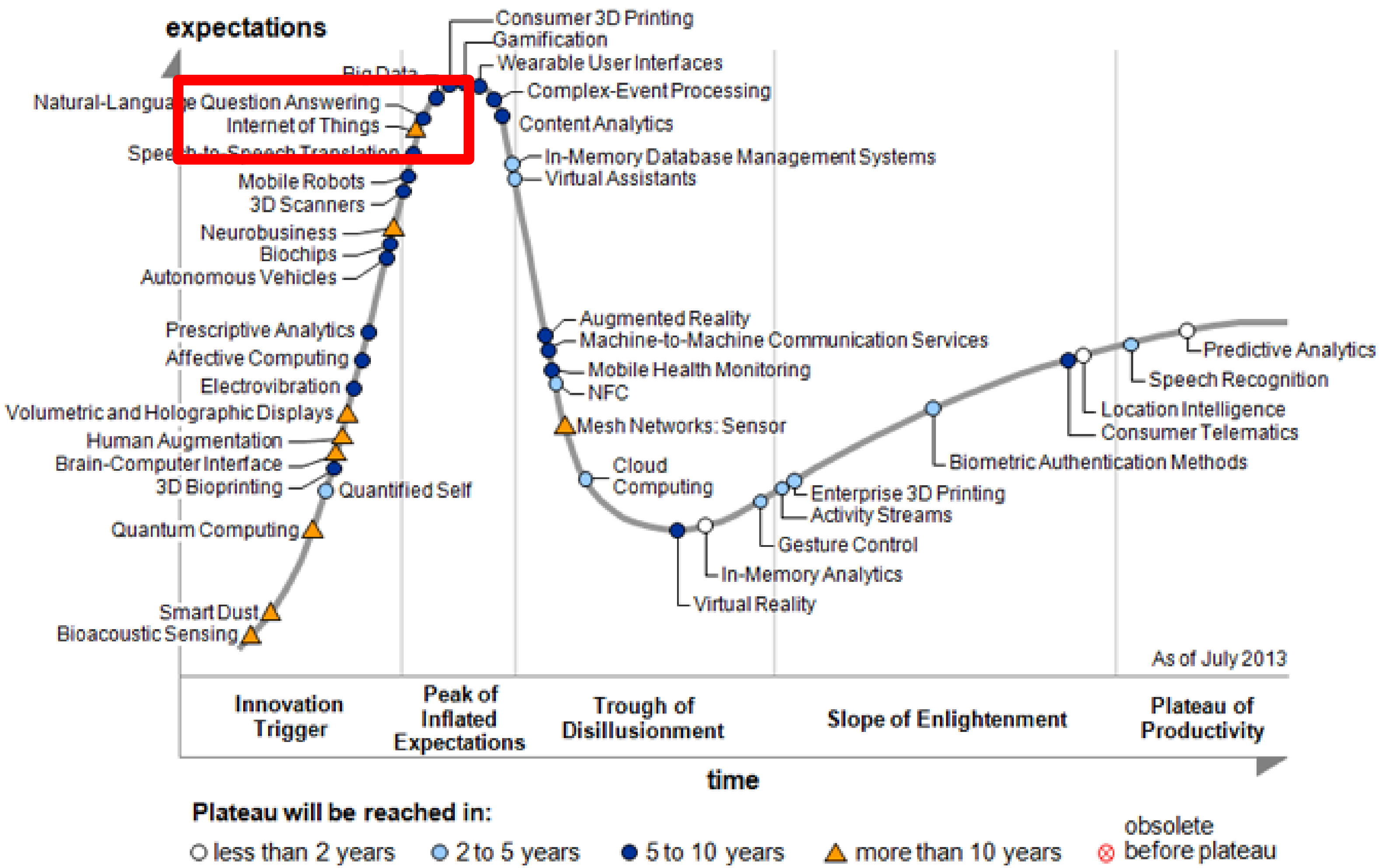
Gartner, 2011



Gartner, 2012



Gartner, 2013



IMHO… IoT is

Interacting with the

Physical World

using **Software**

and **Networking**

Internet Of Things

1: IPv6

- Open Standard
- Widespread
- Scalable
- Auto-configuration
- Simple

Contiki: OS for the IoT

- Runs on small devices, sensor networks, etc
- Low power, low memory
- Certified IPv6 stack
- Supports a variety of hardware



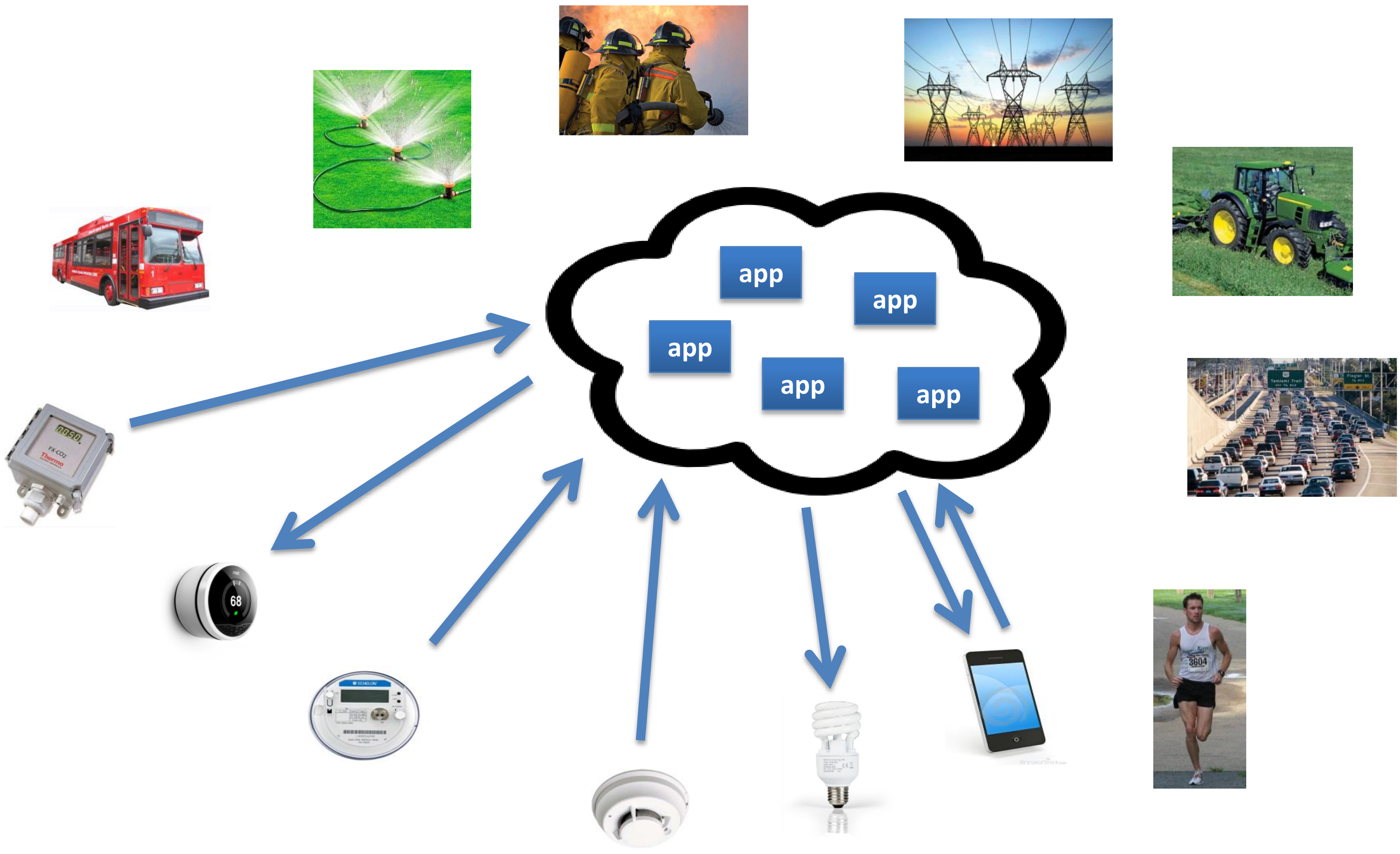

- Open-source, contributors from
  - SICS, Cisco, Redwire LLC, and many others

Internet Of Things

2: Cloud

- Cloud-centric IoT
- Objects as simple as possible
- Intelligence in the Cloud
- Standard interfaces
- No complex application logic

Cloud-Centric IoT

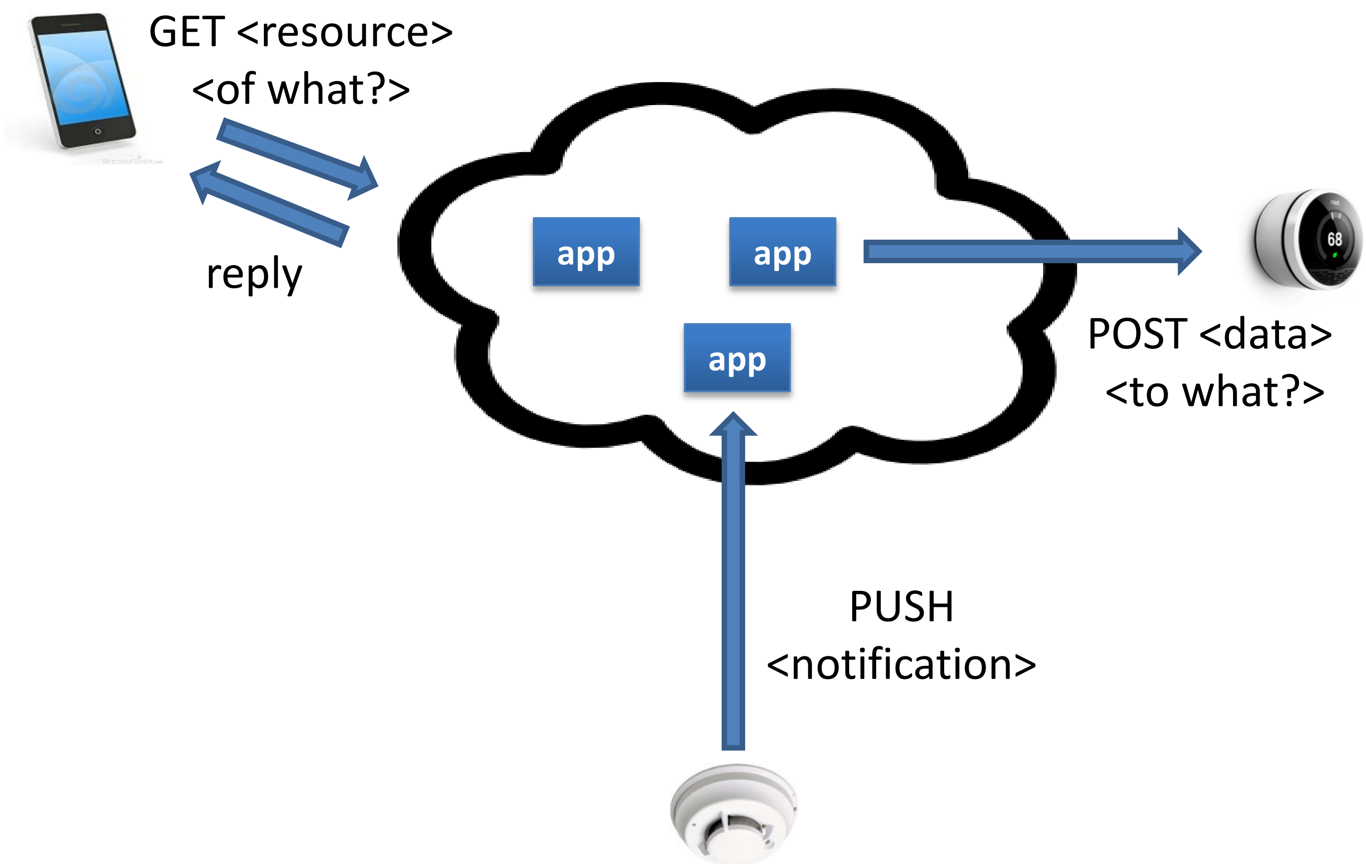


Internet Of Things

3: Semantics

- Object identity
- Virtual/Physical duality
- Service & Role discovery
- CoAP

IoT will be based on semantics



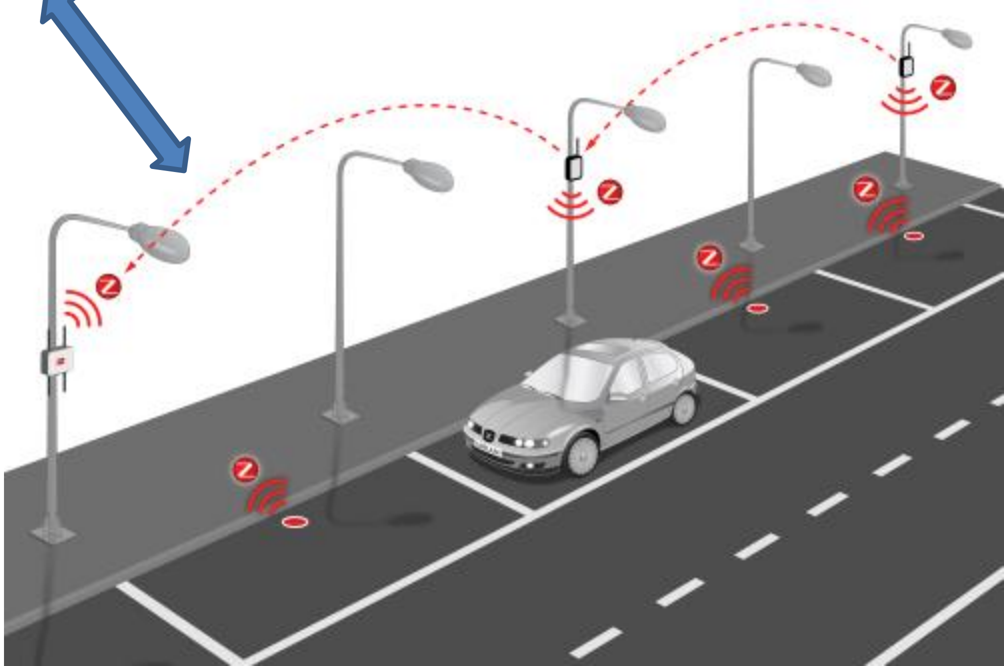
Some Applications

Smart Parking

Cloud App:

- APIs
- Path Calculation
- WSN Mngt
- Billing

Keep objects and clients simple



http://www.libelium.com/smart_parking/

Smart Grid

- Clever electricity management
  - Generation, distribution, consumption
  - Green, efficient, secured

- Dynamic pricing
- Anticipation of needs

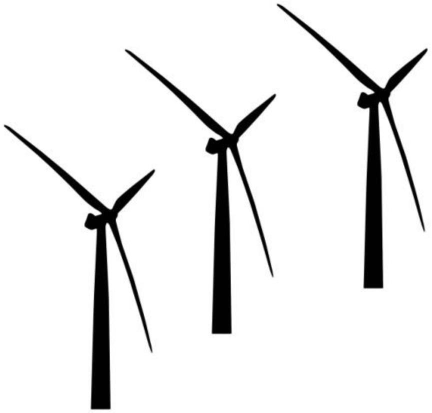
Smart Grid in a Zoning

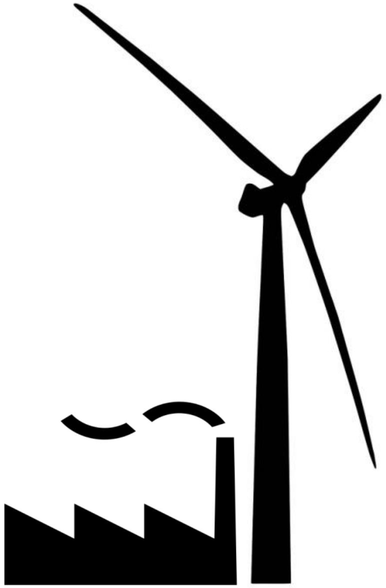
Smart Grid in a Zoning

= €

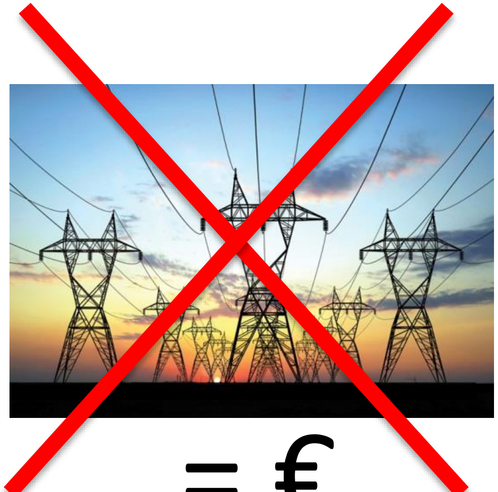
Smart Grid in a Zoning

= €

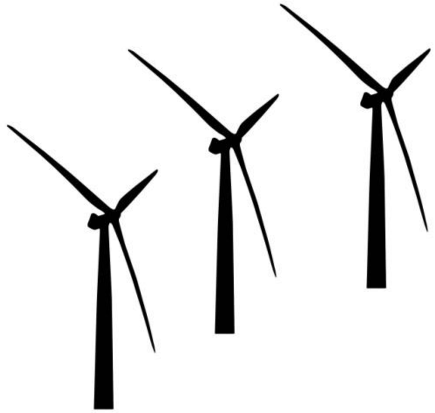




Smart Grid in a Zoning



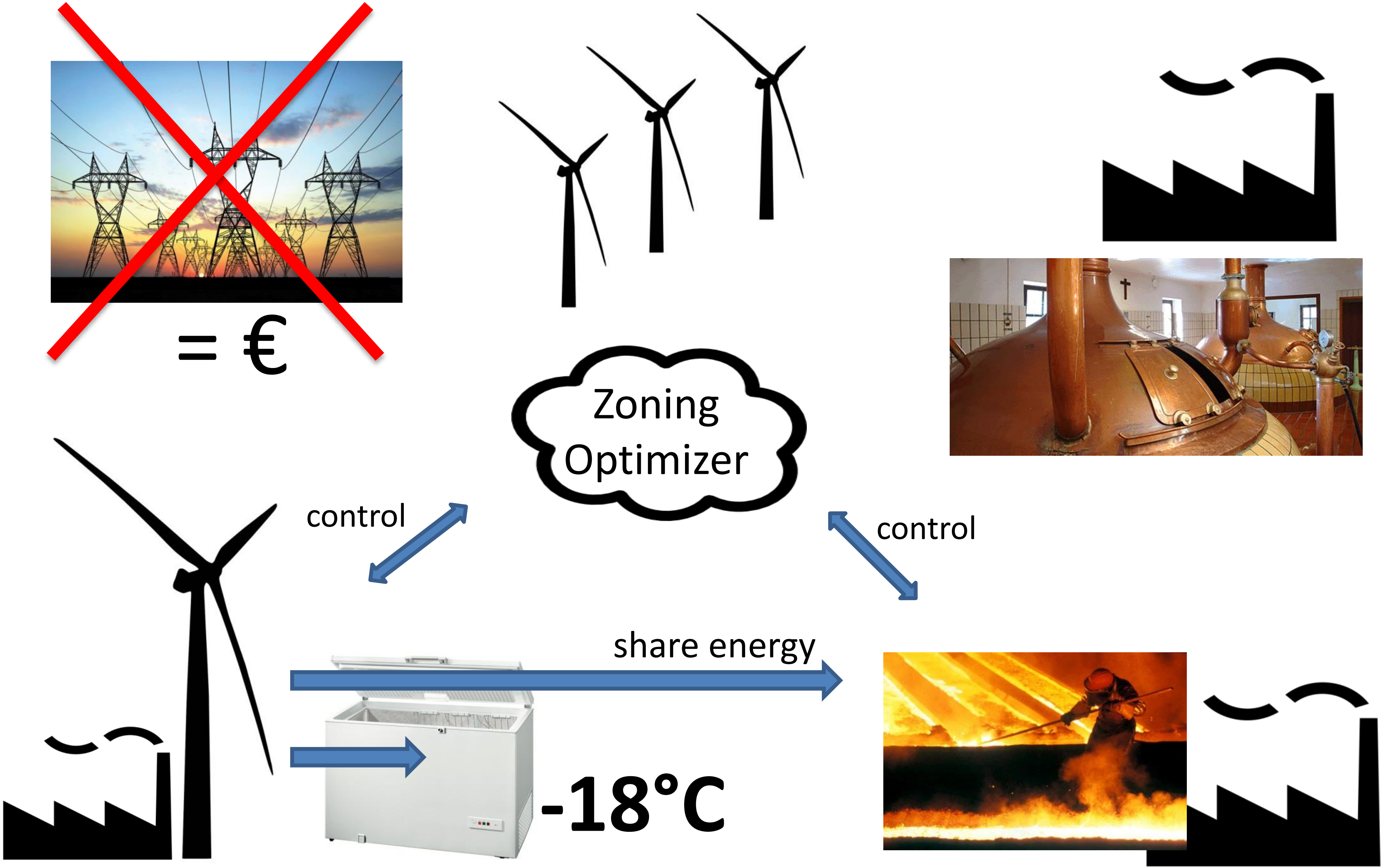
= €



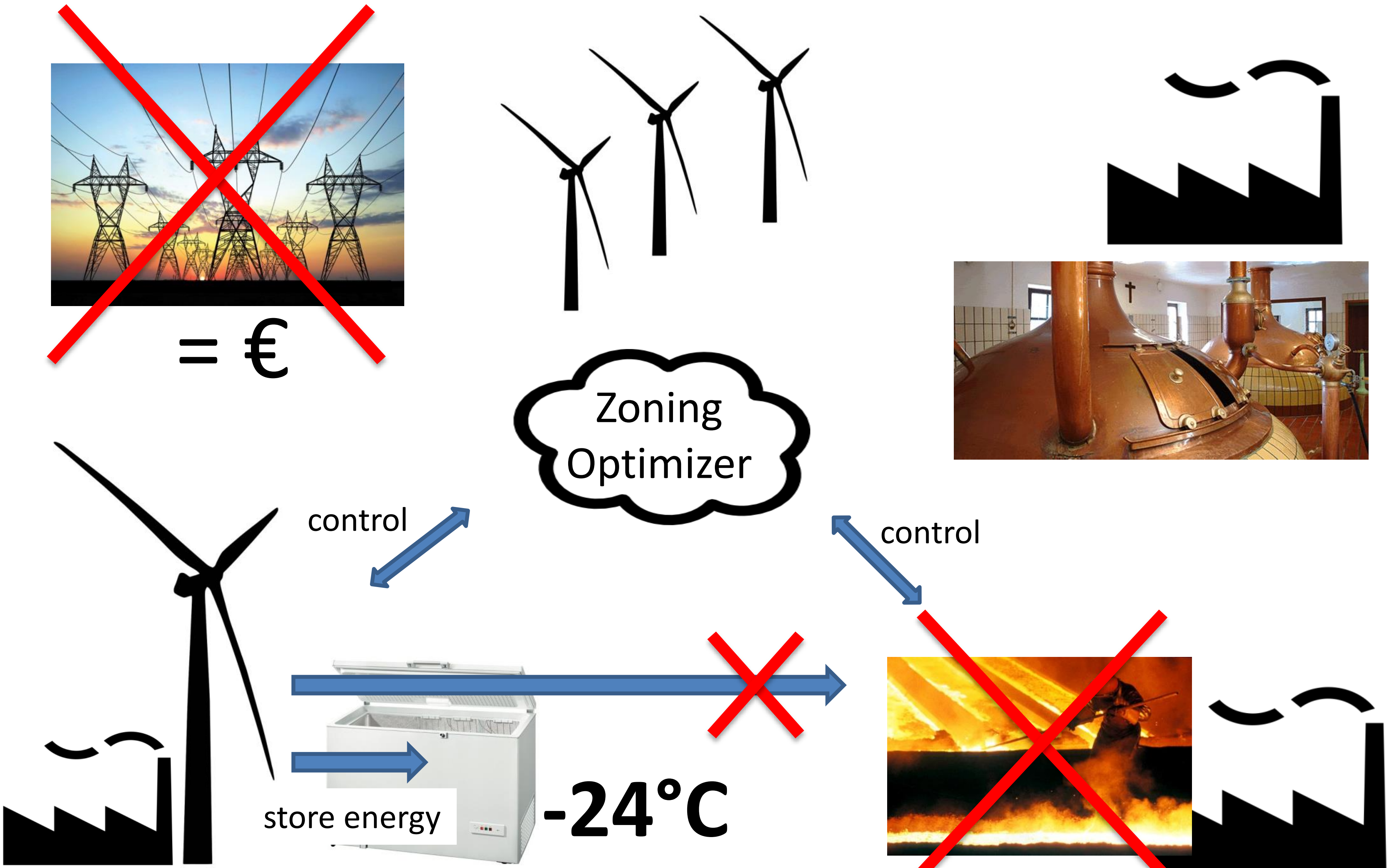
Let’s optimize locally

Self-sufficiency

Smart Grid in a Zoning



Smart Grid in a Zoning



Smart Grid in a Zoning

Inputs:

- Production forecast
- Consumption forecast
- Urgency of tasks
- Optimization Criteria

Outputs:

- Schedule proposition
- Announce supply/demand
- Global optimization
    - Instead of local optim.
- Calculate « cost » of kWh

WEB OF THINGS

architecting the web of things for tinkerers and hackers

- Is one vision of the IoT
- Inspired by the World Wide Web
- Objects connect through the Web
- Objects use Web protocols (& technologies & tools)
- Adoption of the **RESTful architecture**

Designing The

IoT

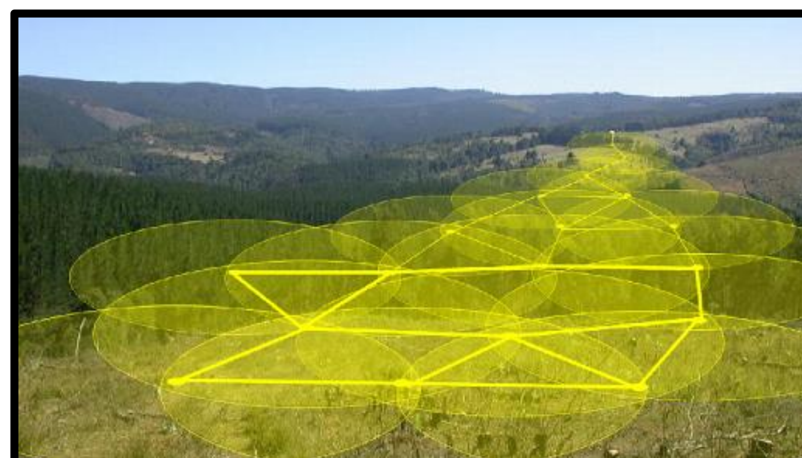
Open Standards

- What we already had
  - IPv4/6
  - A massive internet
  - Isolated Wireless Sensor Networks (WSN)

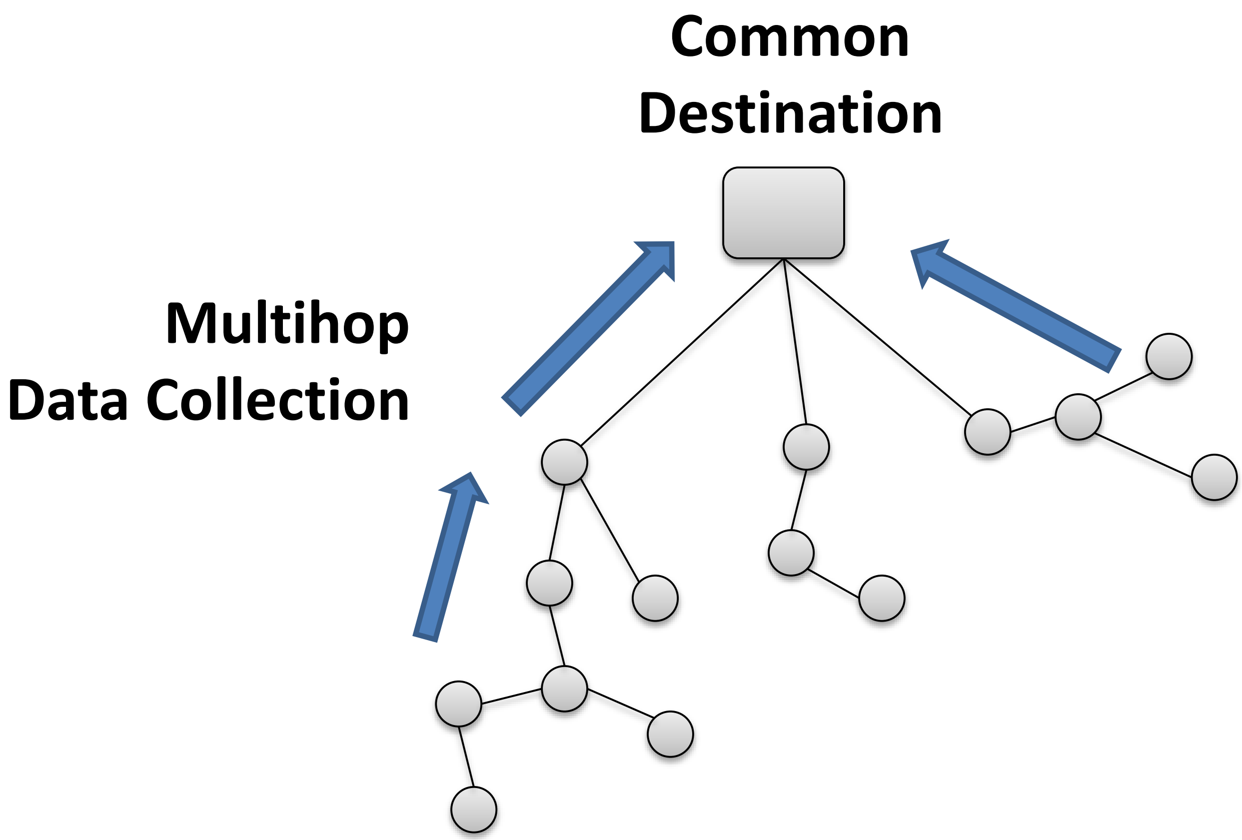
Wireless Sensor Networks (WSN)

Traditionally:

- Multihop
- Autonomous
  - Low-Power
  - Low-Throughput
  - No Maintenance
- Fault-tolerant
- Data-collect
- Gateways



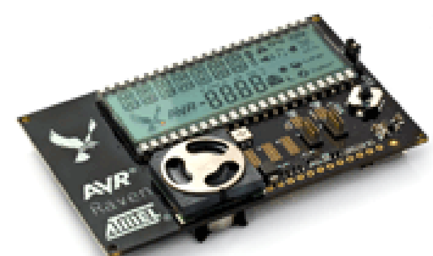
Wireless Sensor Networks (WSN)



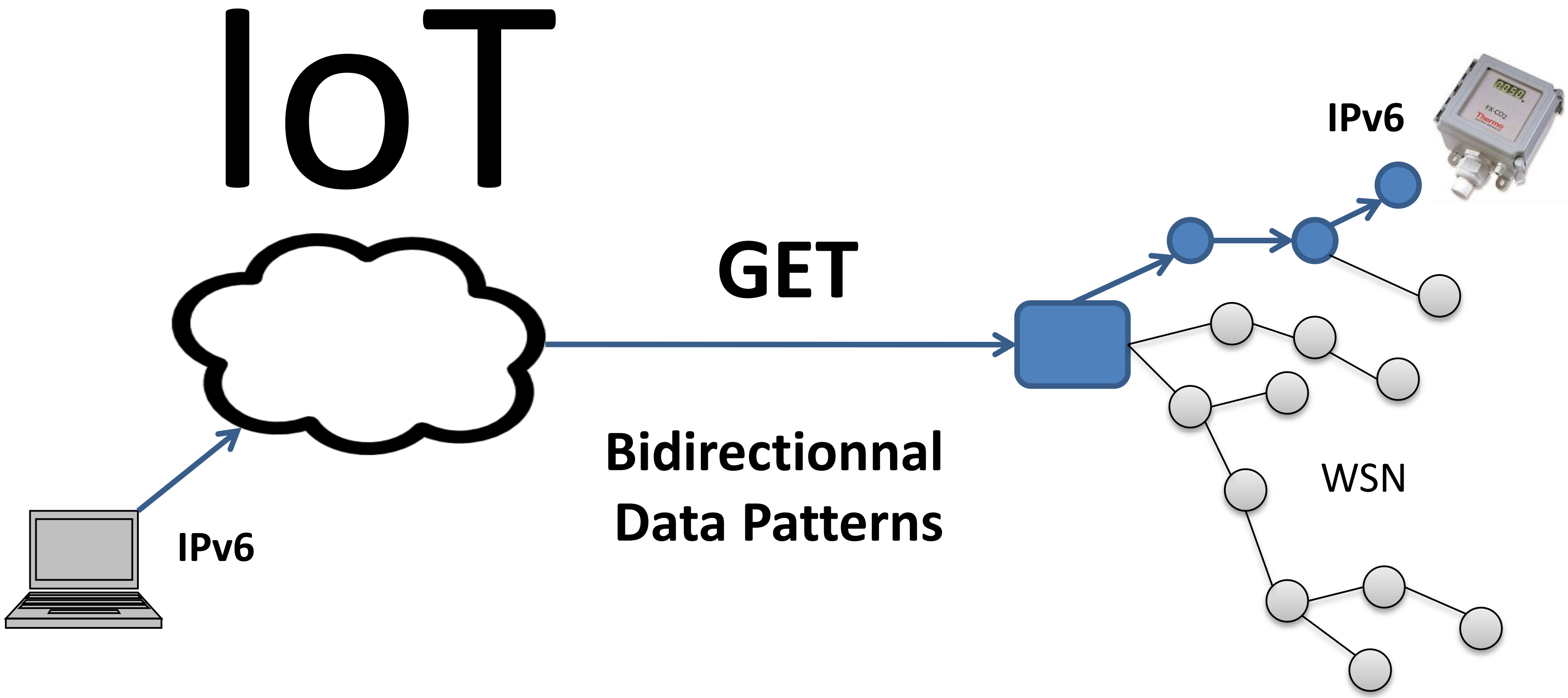
Typical WSN Hardware

- µC
  - 8-16kB RAM, 48-128kB Flash
  - A few MHz
- 802.15.4 radio
- a few sensors
- GPIO
- Some external flash (1-8 Mbit)
- Batteries
- Cheap

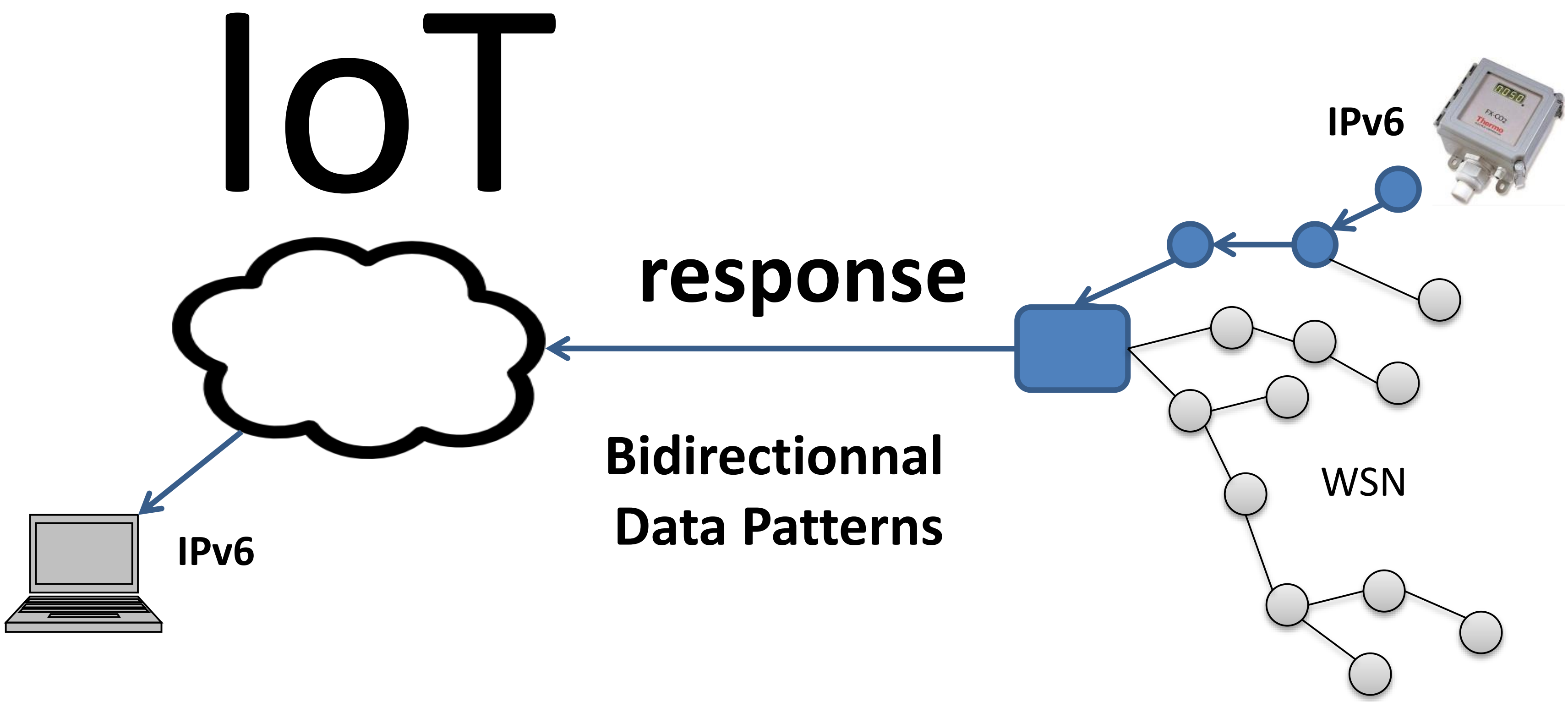


Wireless Sensor Networks (WSN)
Now becoming a subset of IoT

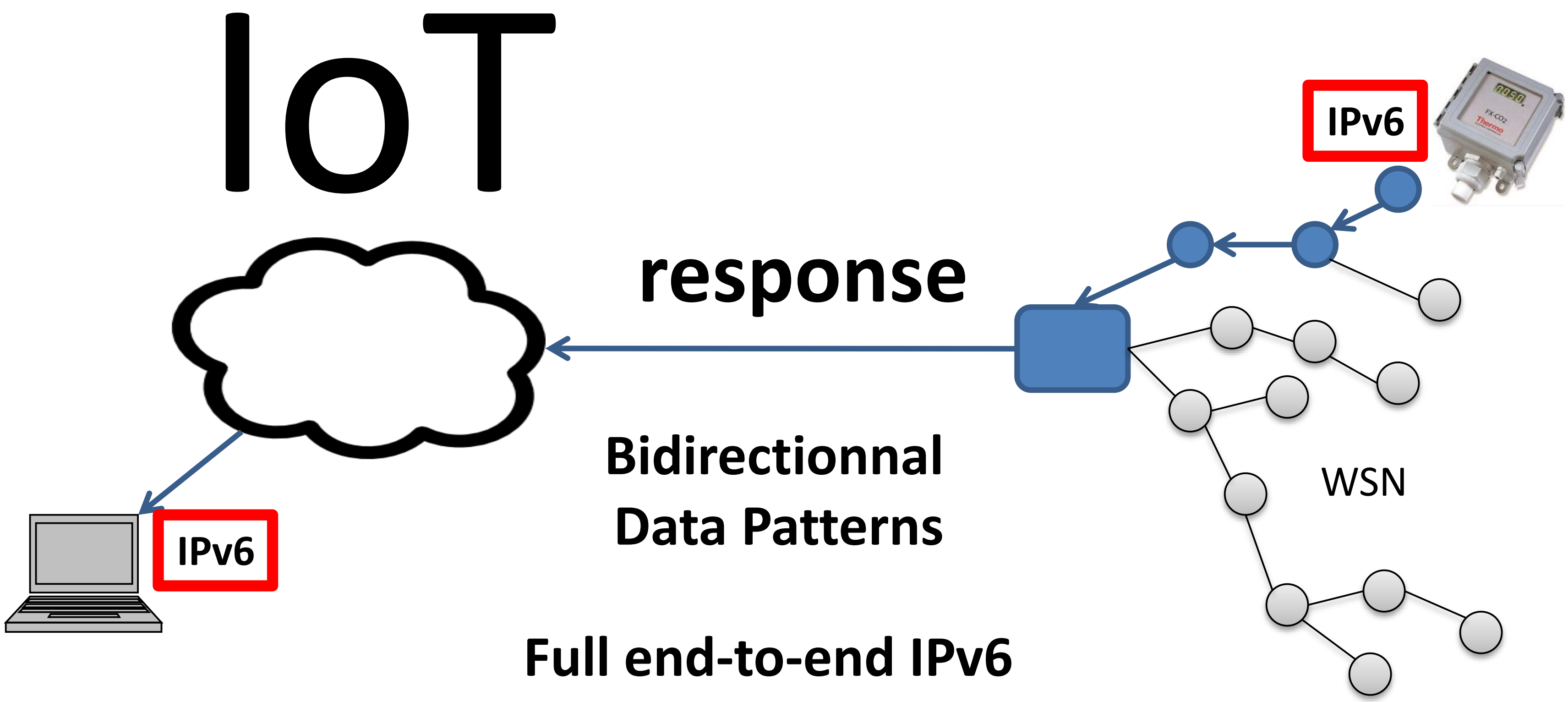


Wireless Sensor Networks (WSN)
Now becoming a subset of IoT



Wireless Sensor Networks (WSN)

Now becoming a subset of IoT

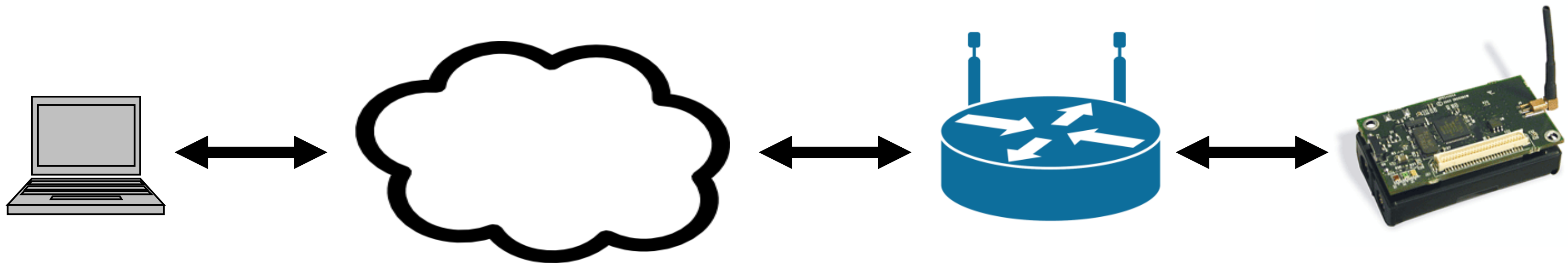


Open Standards

- What we already had
  - IPv4/6
  - A massive internet
  - Isolated Wireless Sensor Networks (WSN)

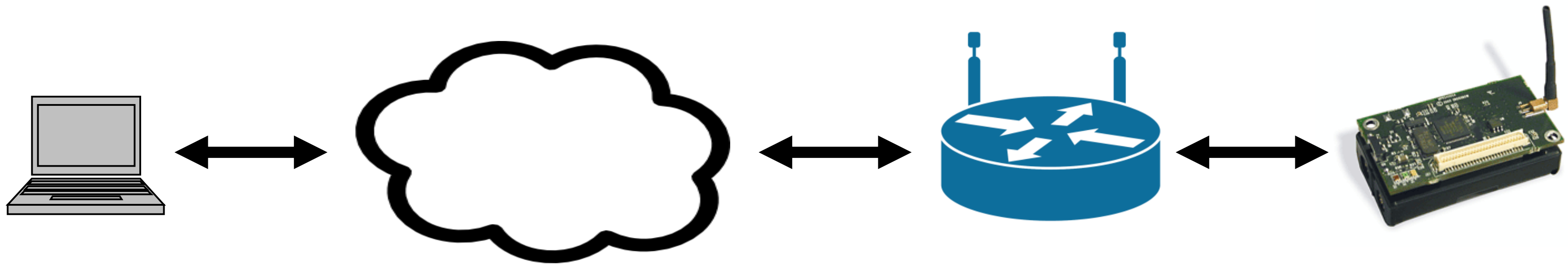
- **What was added (IETF)**
  - **6LoWPAN: IPv6 for 802.15.4 WSNs**
  - **RPL: Routing protocol for 6LoWPAN**
  - **CoAP: RESTful protocol for constrained applications**

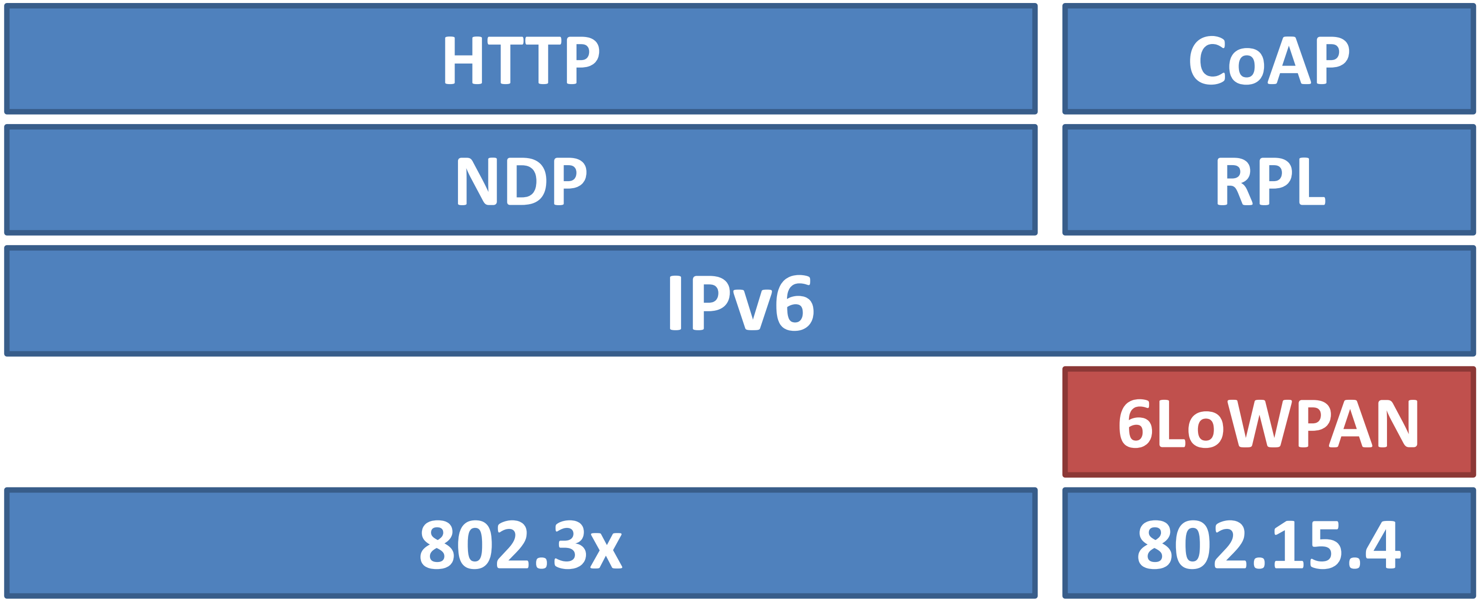
IPv6: the Narrow Waist



HTTP	CoAP
NDP	RPL
IPv6	
	6LoWPAN
802.3x	802.15.4

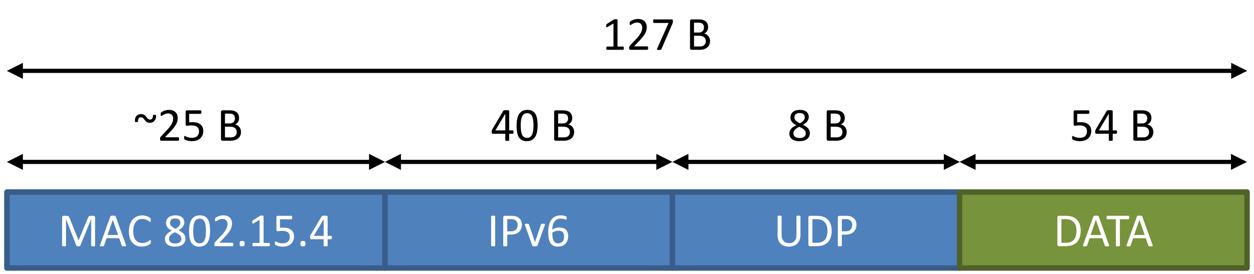
IPv6: the Narrow Waist





6LoWPAN: IPv6 for 802.15.4

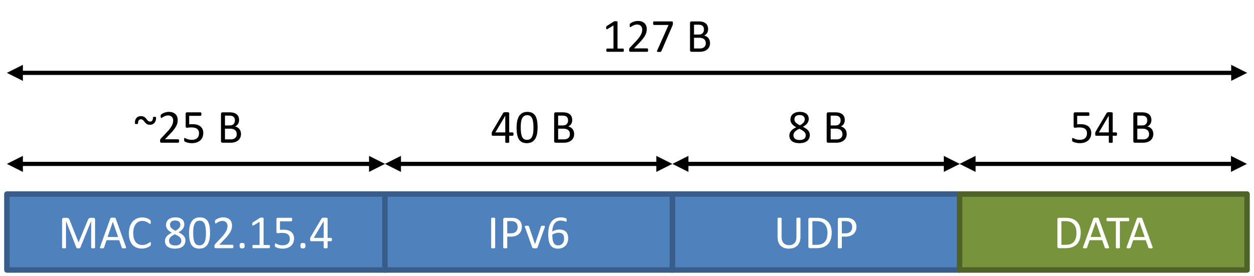
- Maximum Transmission Unit (MTU)
  - Minimum 1280 bytes in IPv6
  - But Maximum 127 bytes MAC frames in 802.15.4



56% overhead
44% data

6LoWPAN: IPv6 for 802.15.4

- Maximum Transmission Unit (MTU)
  – Minimum 1280 bytes in IPv6
  – But Maximum 127 bytes MAC frames in 802.15.4

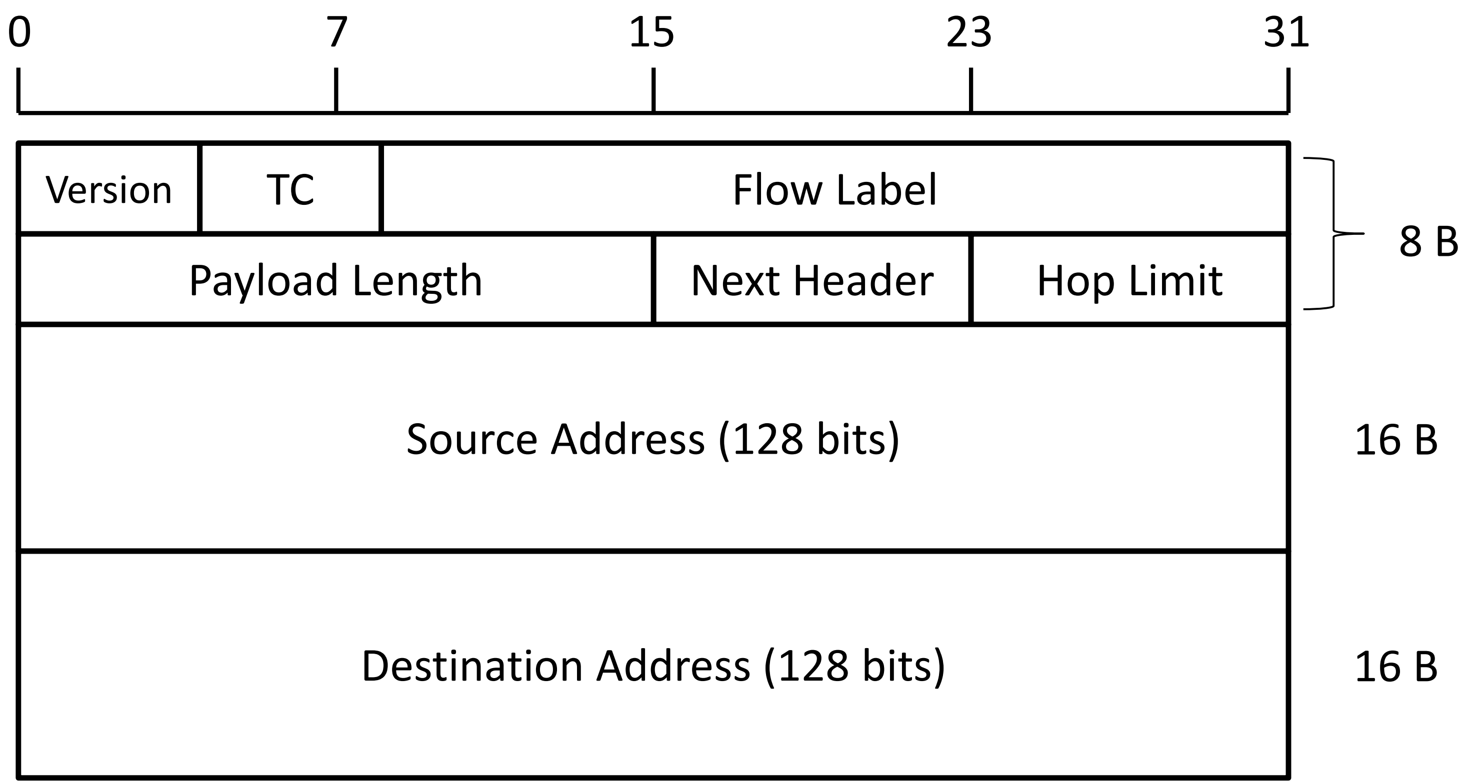


56% overhead
44% data

- 6LoWPAN: Reduce the IPv6 packet size
  – Header Compression
  – Fragmentation

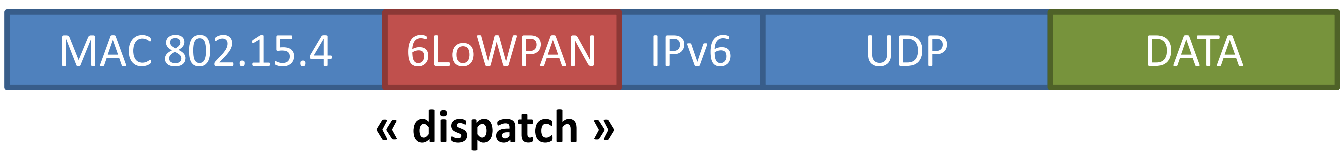
6LoWPAN Header Compression

- Goal: Compress as much of the IPv6 Header



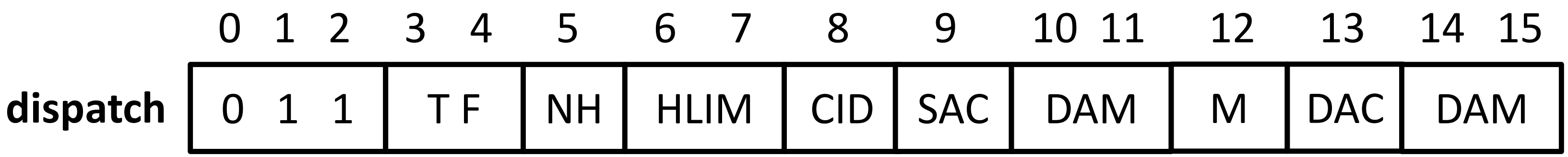
6LoWPAN Header Compression

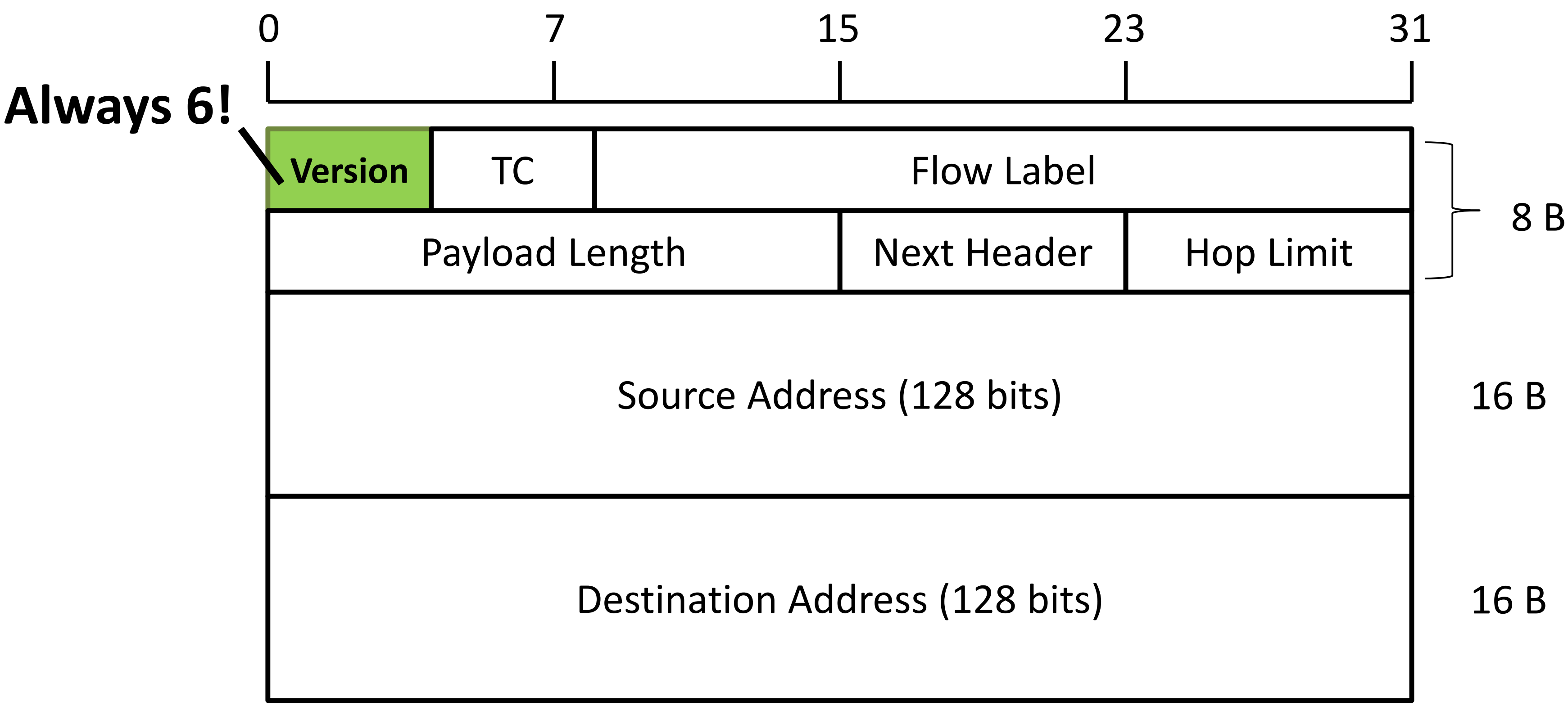
- Goal: Compress as much of the IPv6 Header
- Insert 6LoWPAN Header with compression information



- 3 mechanisms
  – Make assumptions
  – Infer Information from 802.15.4 Header
  – Compress possible values to a subset

6LoWPAN Header Compression



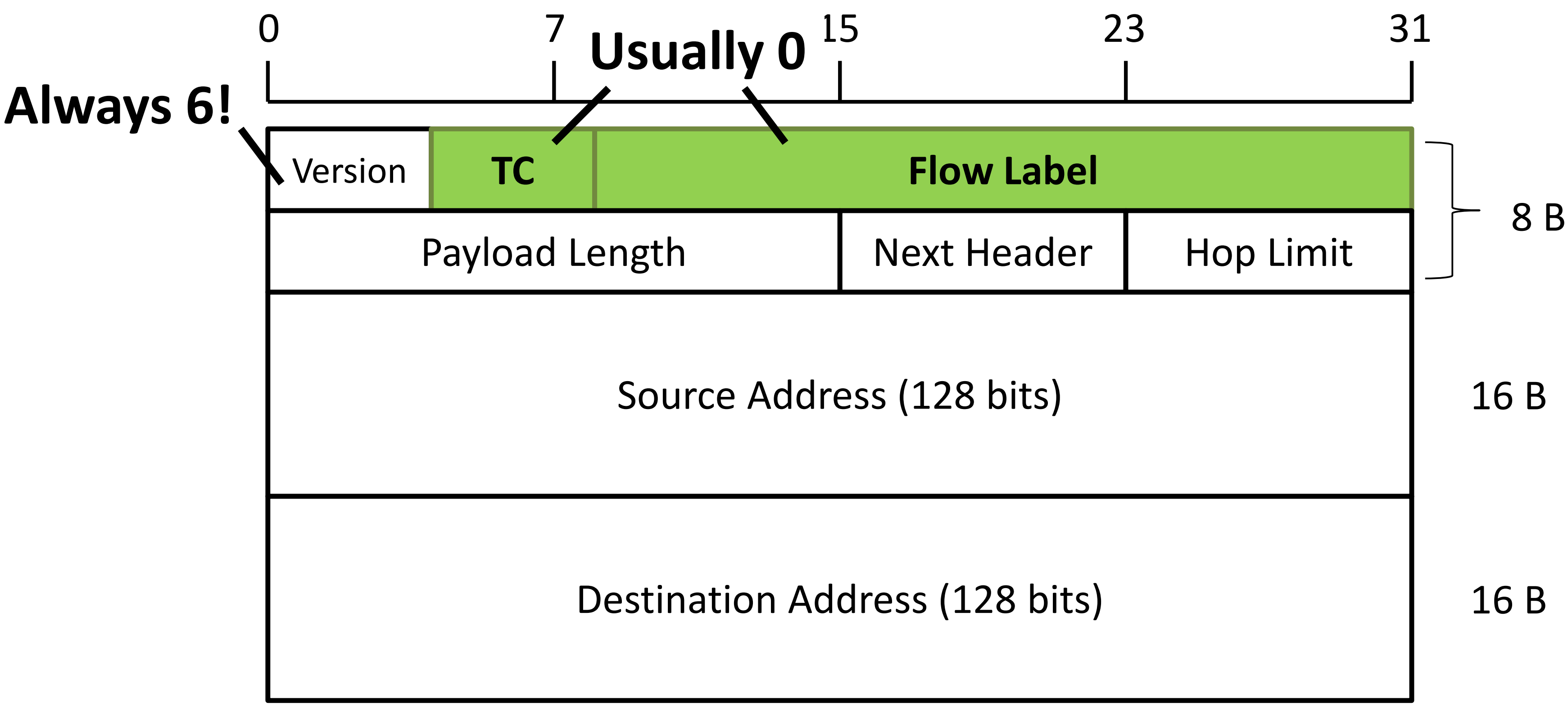


6LoWPAN Header Compression

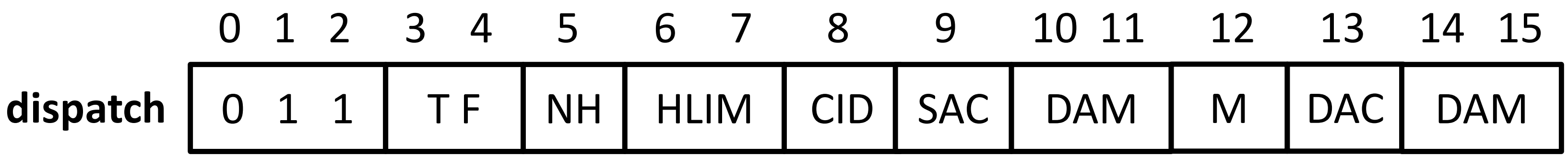
	0 1 2	3 4	5	6 7	8	9	10 11	12	13	14 15
dispatch	0 1 1	**T F**	NH	HLIM	CID	SAC	DAM	M	DAC	DAM

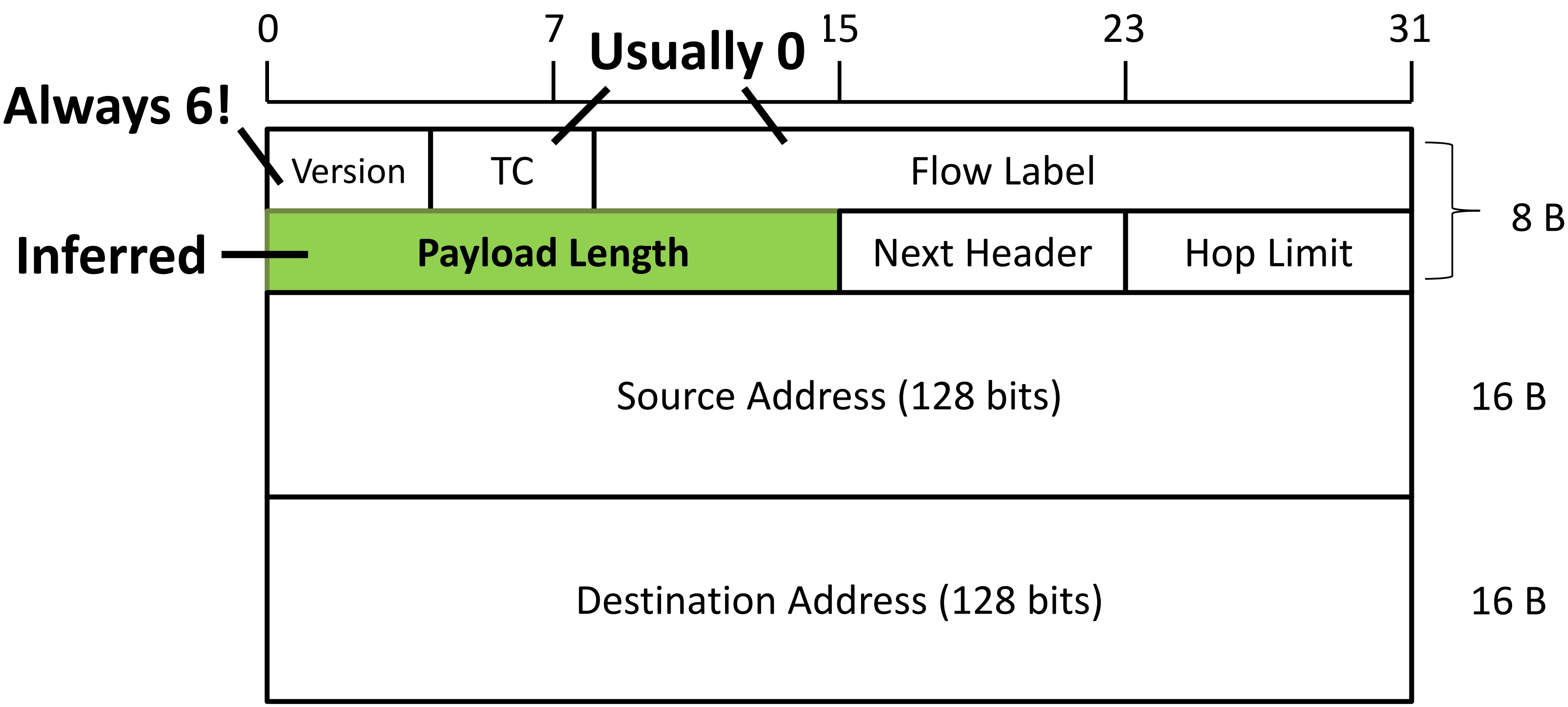
4 possible states of TF: carried inline or not, etc

→ Best case (11): assume all 0, save 3 bytes



6LoWPAN Header Compression



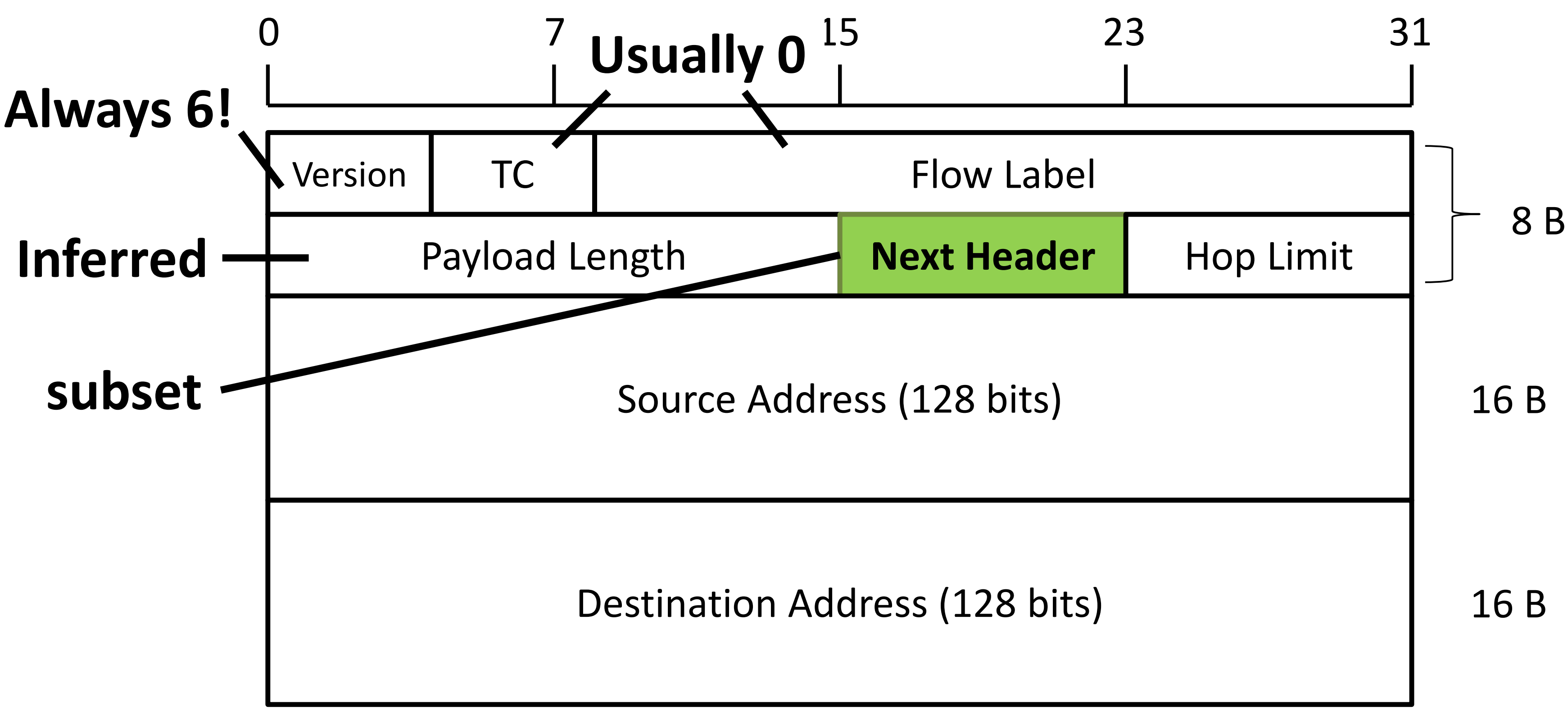


6LoWPAN Header Compression

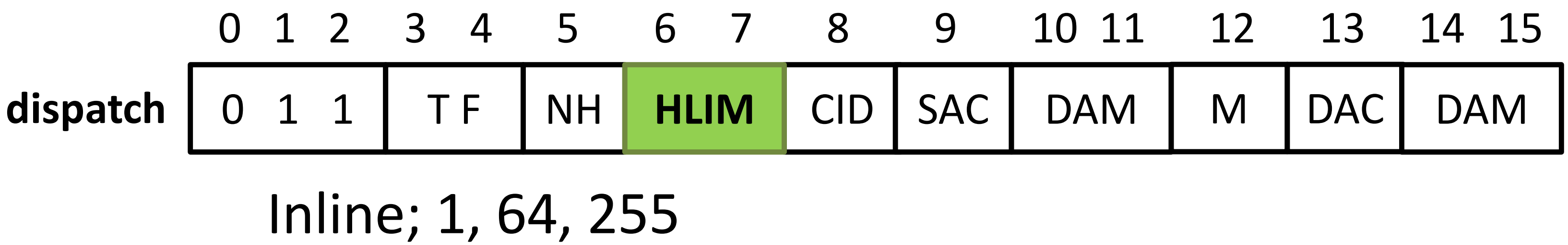
	0 1 2	3 4	5	6 7	8	9	10 11	12	13	14 15
dispatch	0 1 1	T F	**NH**	HLIM	CID	SAC	DAM	M	DAC	DAM

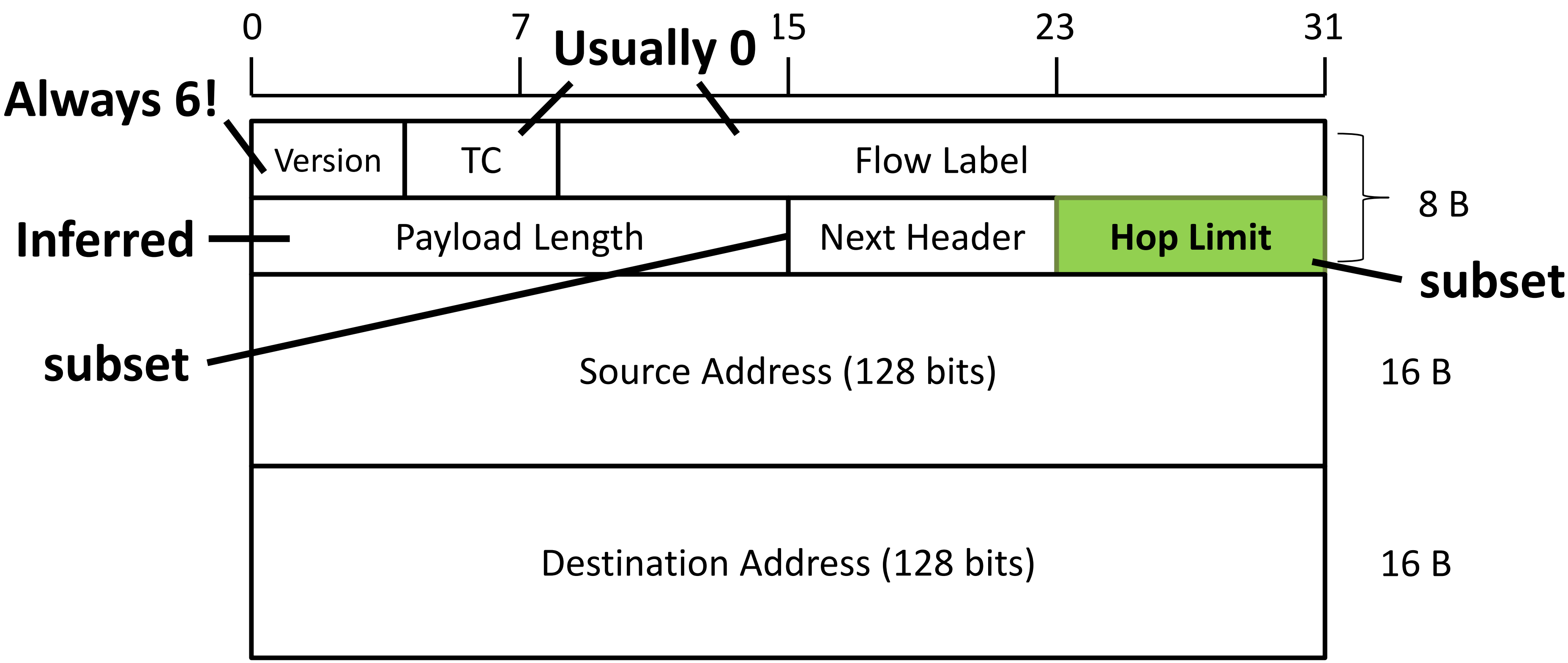
0: inline = 1 byte in IPv6 + next header inline

1: compressed = 1 byte defining NH + compression of next header

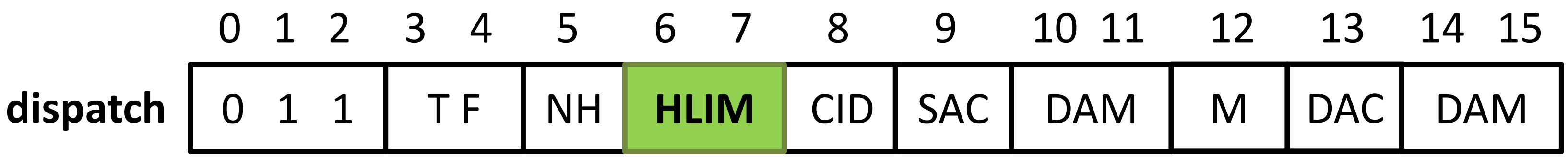


6LoWPAN Header Compression



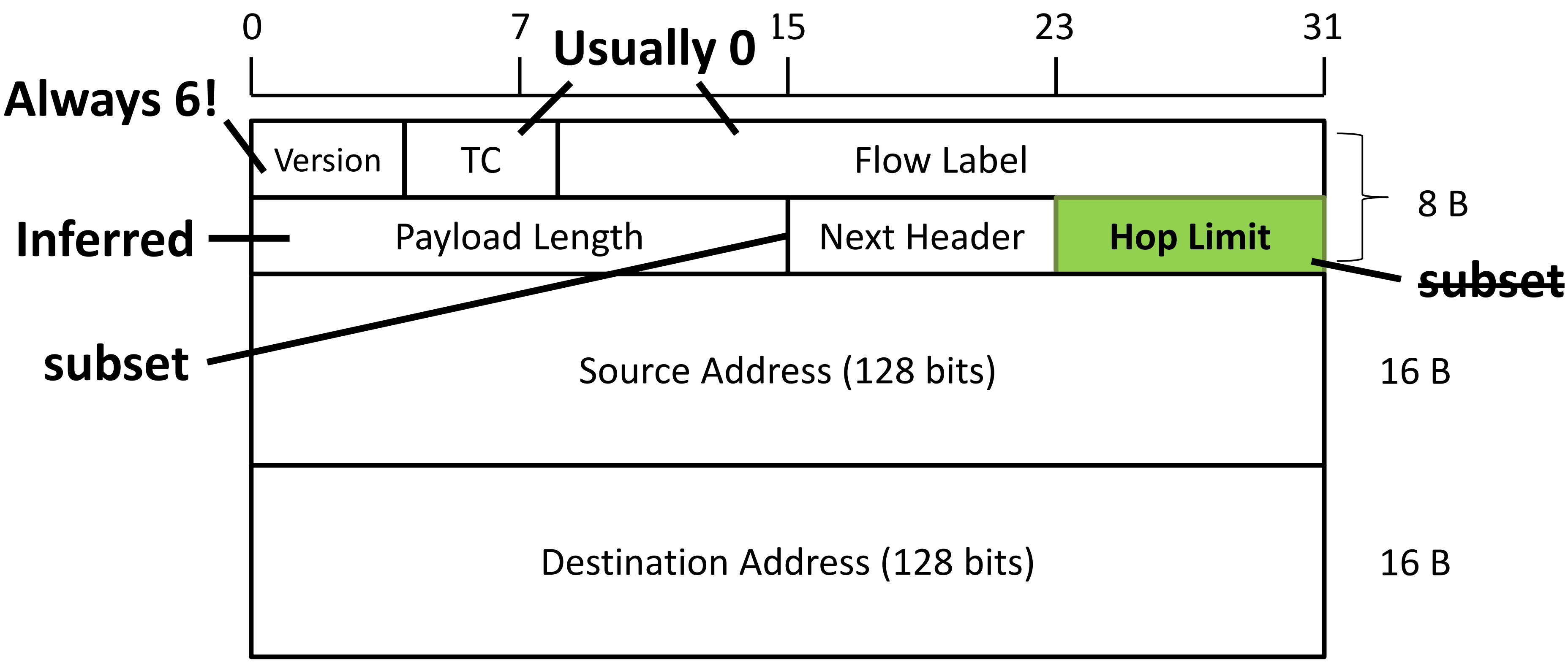


6LoWPAN Header Compression

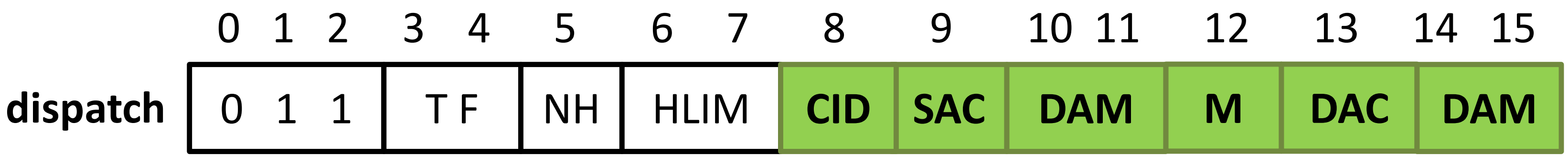


Inline; ~~1, 64, 255~~

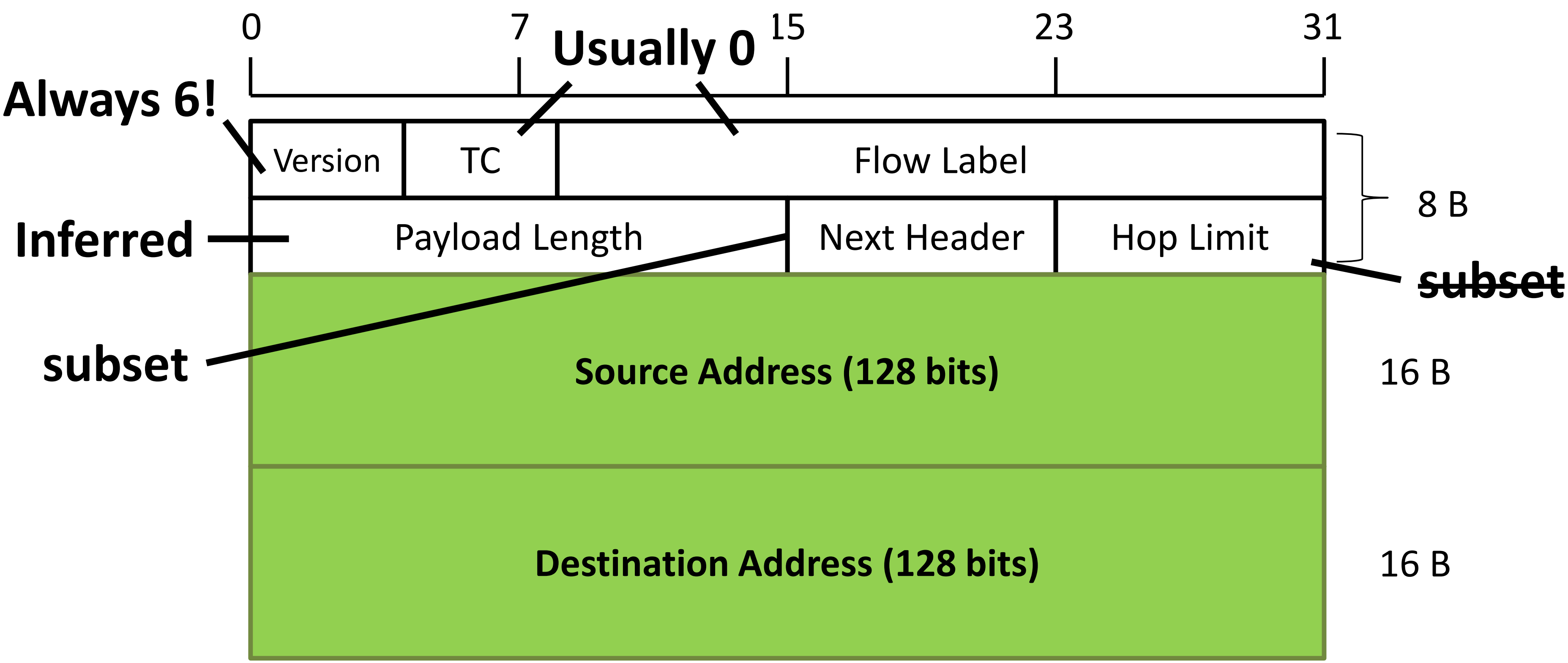
We need it for multihop



6LoWPAN Header Compression



Most common: infer 64-bit prefix from link-local prefix
CID: stateful compression (more gain)

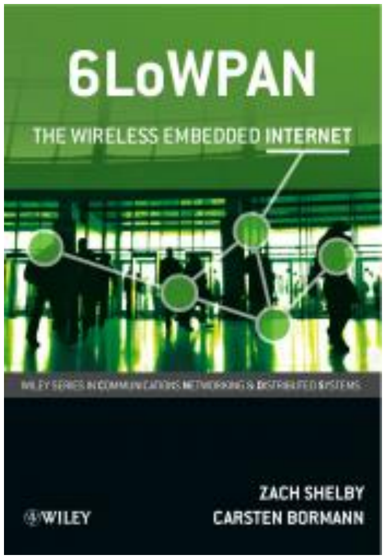


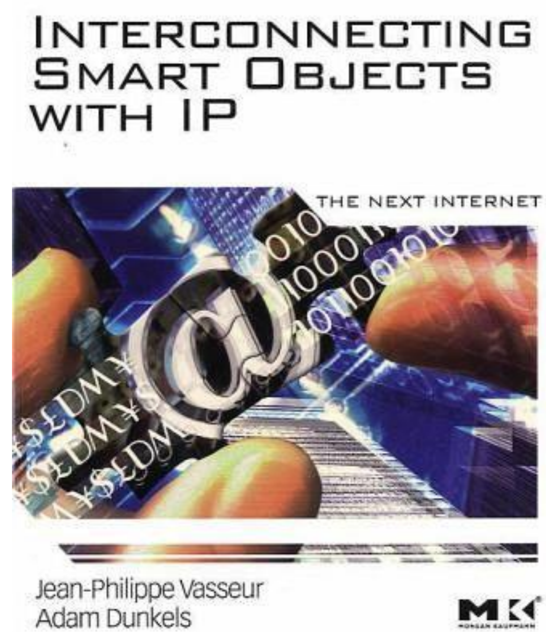
6LoWPAN Header Compression

- Result:
  - IPv6 Header decreases from 40 bytes to about 7
  - 80-90 bytes for data
- Fragmentation takes care of larger packets
  - Fragmentation header with fragment ##

More on 6LoWPAN

6LoWPAN: The Wireless Embedded Internet





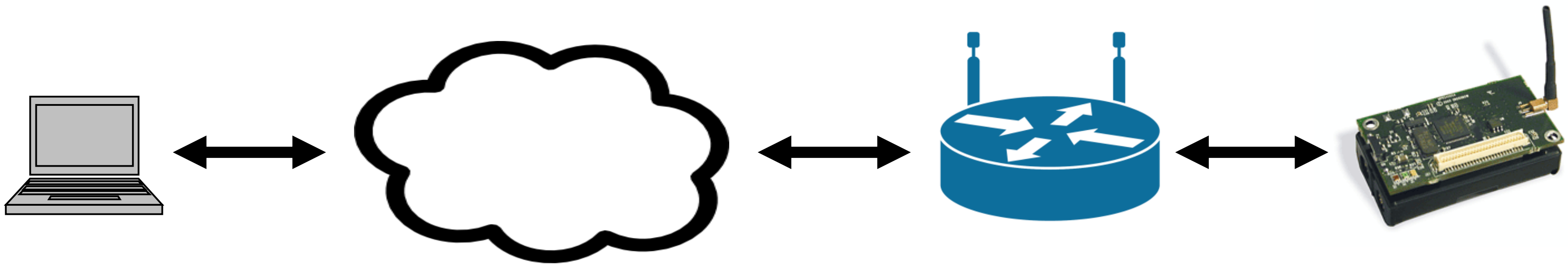
by Zach Shelby (~~Sensinode~~, ARM)
and Carsten Bormann (Universität Bremen)

by Adam Dunkels (~~SICS~~, Thingsquare)
and Jean-Philippe Vasseur (Cisco)

Great Wikipedia Page in FR! (forget the EN one)

http://fr.wikipedia.org/wiki/6LoWPAN

IPv6: the Narrow Waist



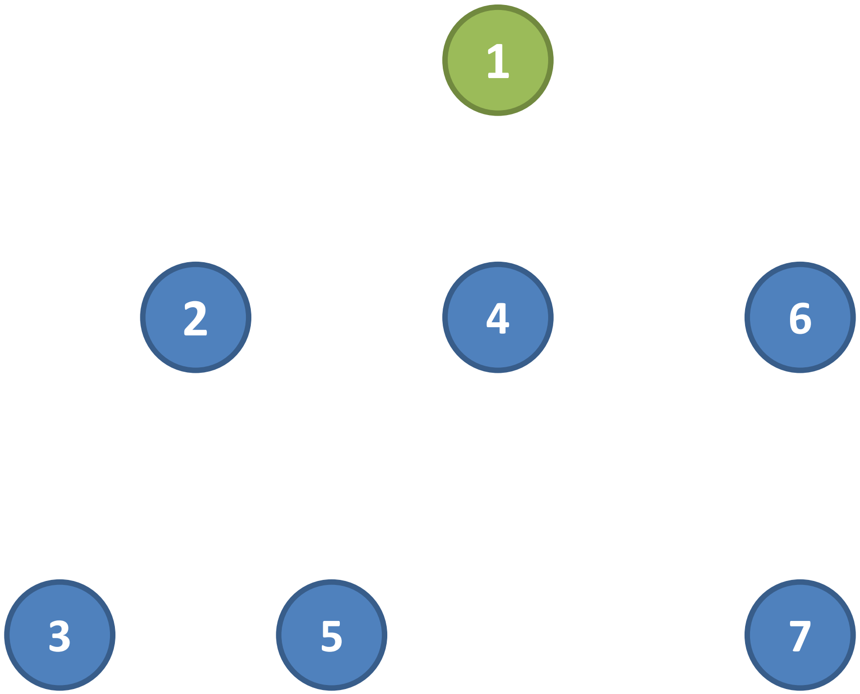
HTTP	CoAP
NDP	RPL
IPv6	IPv6
	6LoWPAN
802.3x	802.15.4

RPL: Routing Protocol for 6LoWPAN

- Facts
  - Low Power & Lossy Networks (LLN)
  - Links are highly dynamic
  - Must support bidirectional traffic
    - Optimized for collect (MP2P)
    - Basic support of root → node (P2MP)
    - Basic support of node → node (P2P)
  - Energy / availability / latency compromise
    - Must be flexible
  - Scalable (destinations, density)

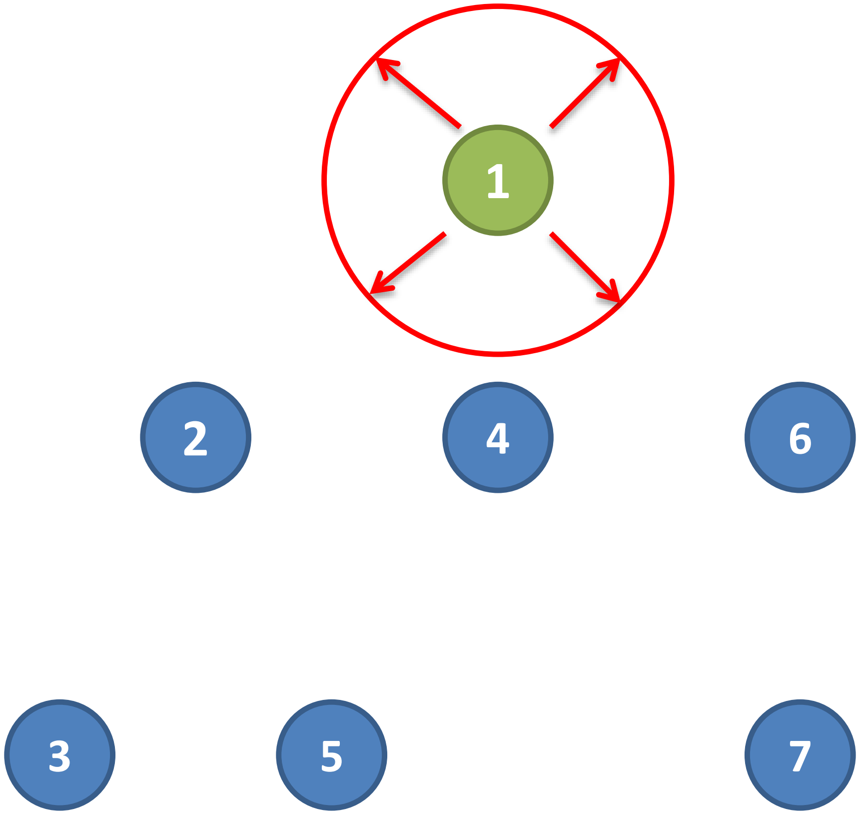
RPL

DODAG: inspired by data-collect protocols



RPL

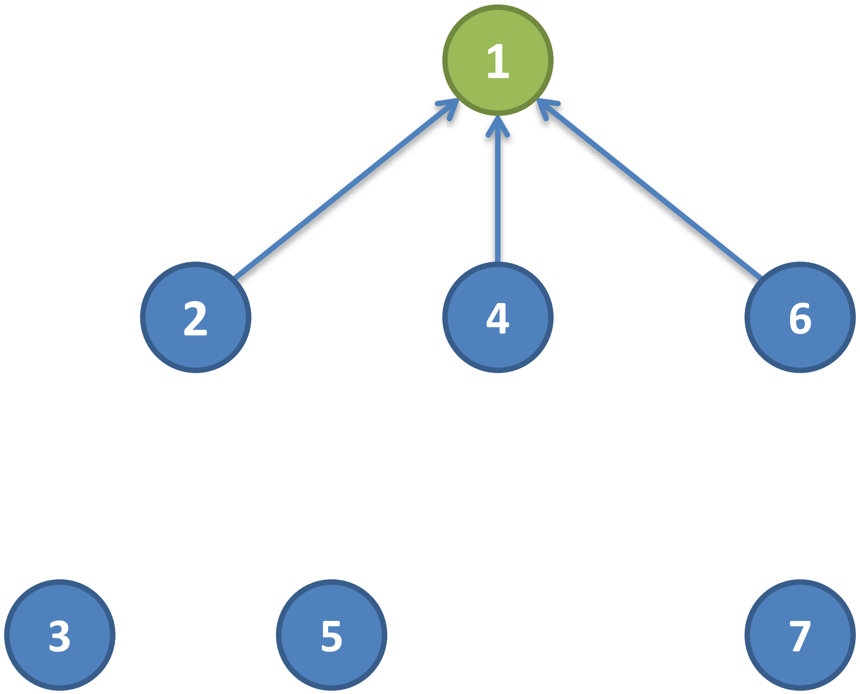
DODAG: inspired by data-collect protocols



- Periodic Broadcasts (DIO)

RPL

DODAG: inspired by data-collect protocols

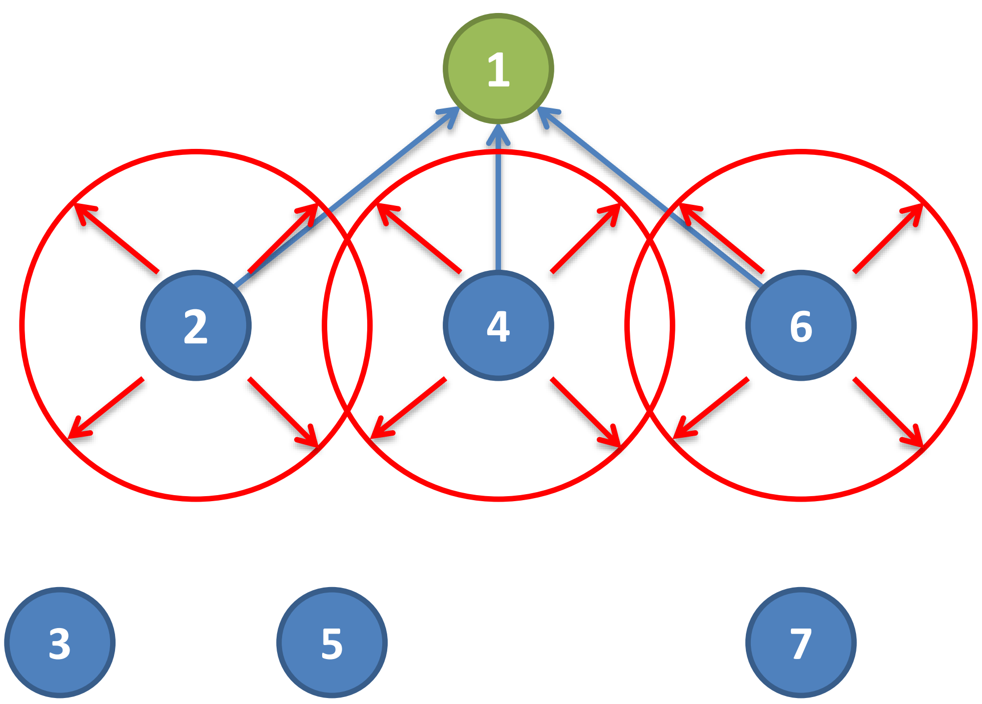


- Periodic Broadcasts (DIO)

→Uproutes

RPL

DODAG: inspired by data-collect protocols

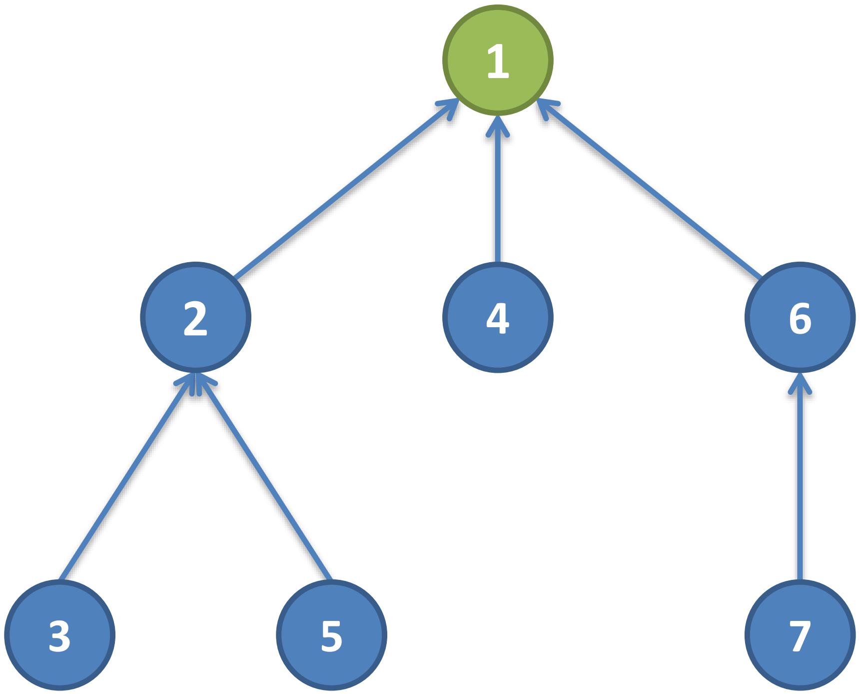


- Periodic Broadcasts (DIO)

→Uproutes

RPL

DODAG: inspired by data-collect protocols

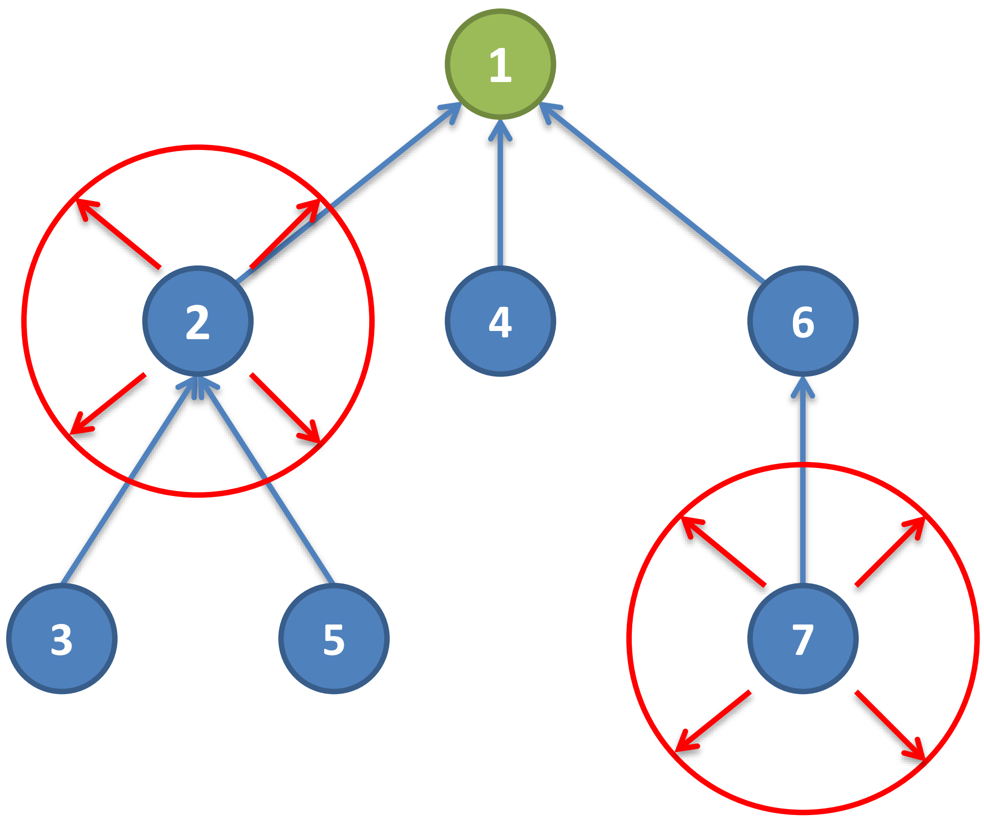


- Periodic Broadcasts (DIO)

→Uproutes

RPL

DODAG: inspired by data-collect protocols



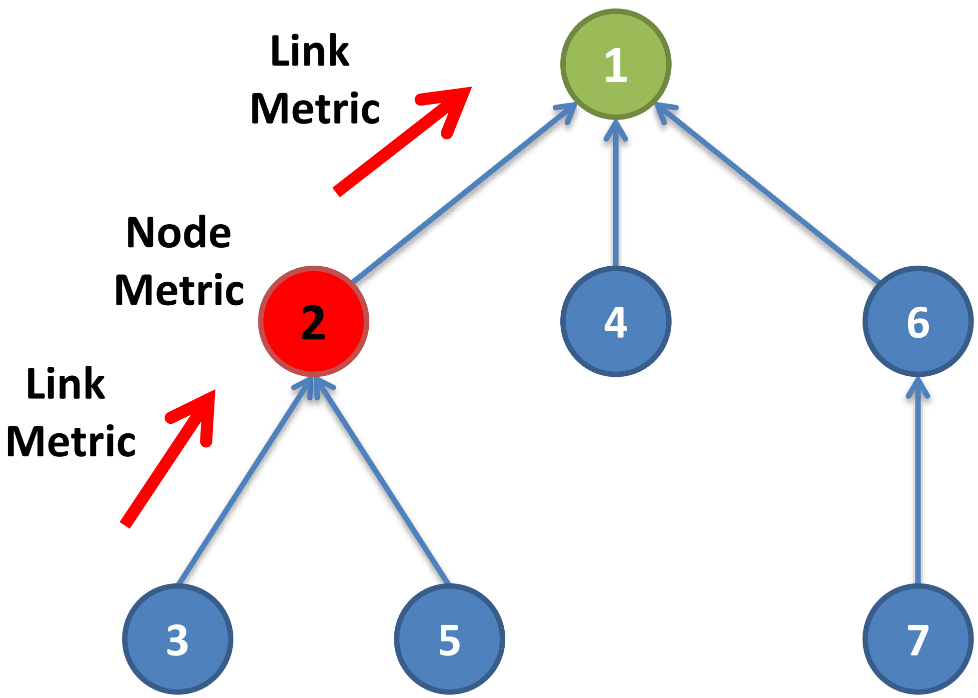
- Periodic Broadcasts (DIO)

→Uproutes

→Keep broadcasting to update ranks (with backoff)

RPL

Distance-vector routing protocol



- Periodic Broadcasts (DIO)

→Uproutes

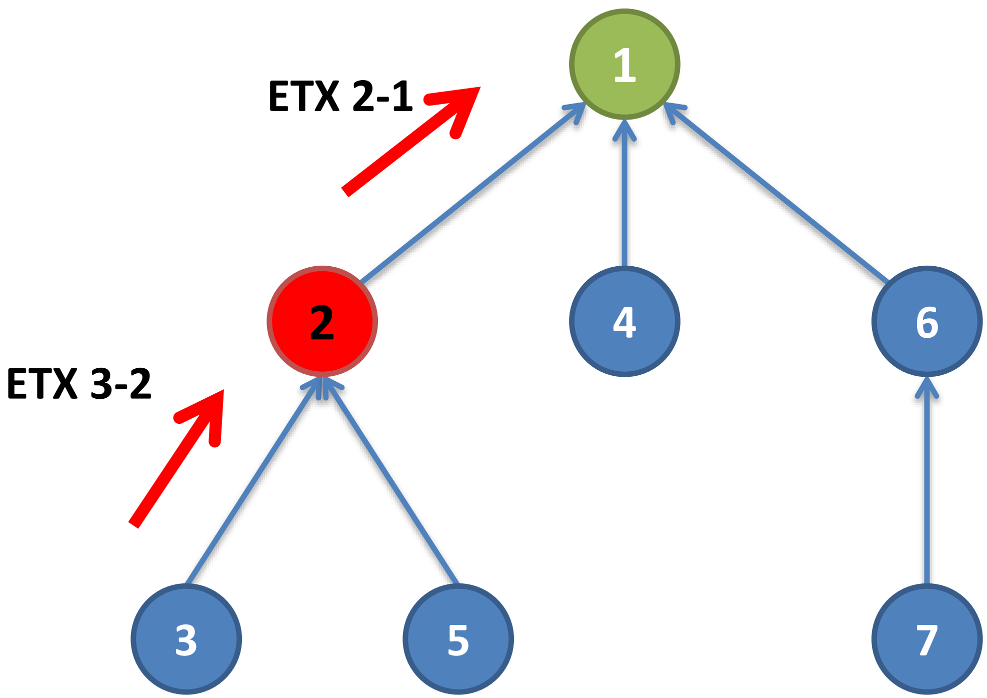
→Keep broadcasting to update **ranks** (with backoff)

- Rank = cost to sink

Rank 3, through 3→2→1 = Link 3→2 + Rank 2

RPL

ETX: Estimated Transmission Count (Link metric)
→ Cumulative TX to reach sink, through a parent

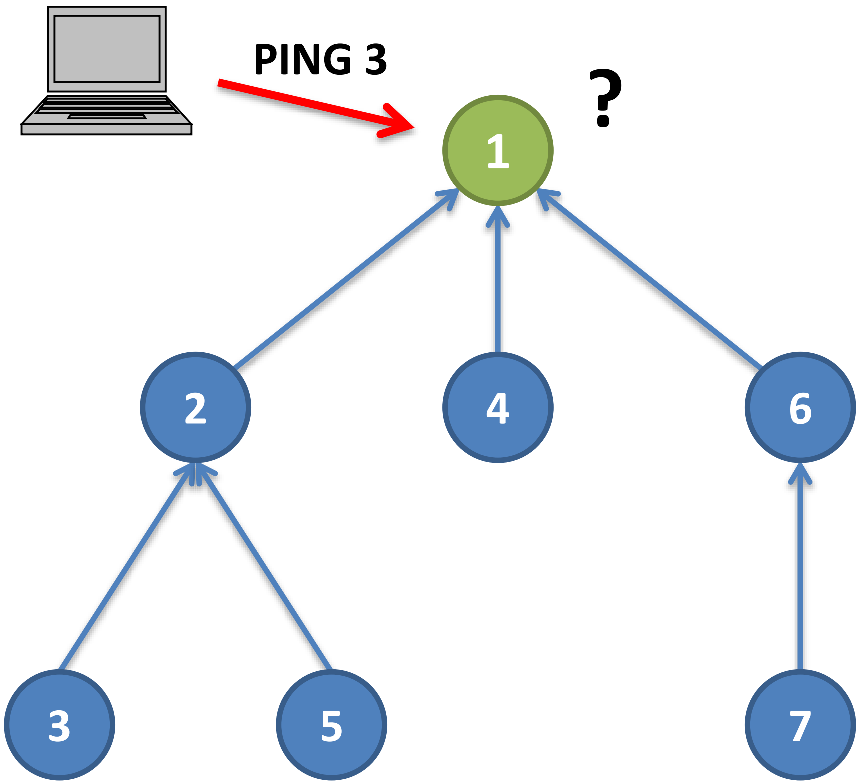


- Periodic Broadcasts (DIO)
→Uproutes
→Keep broadcasting to update **ranks** (with backoff)
- Rank = cost to sink
- **Metric example**: **ETX**, based on dynamic measurements
- Datapath Validation

Rank 3, through 3→2→1 = Link 3→2 + Rank 2

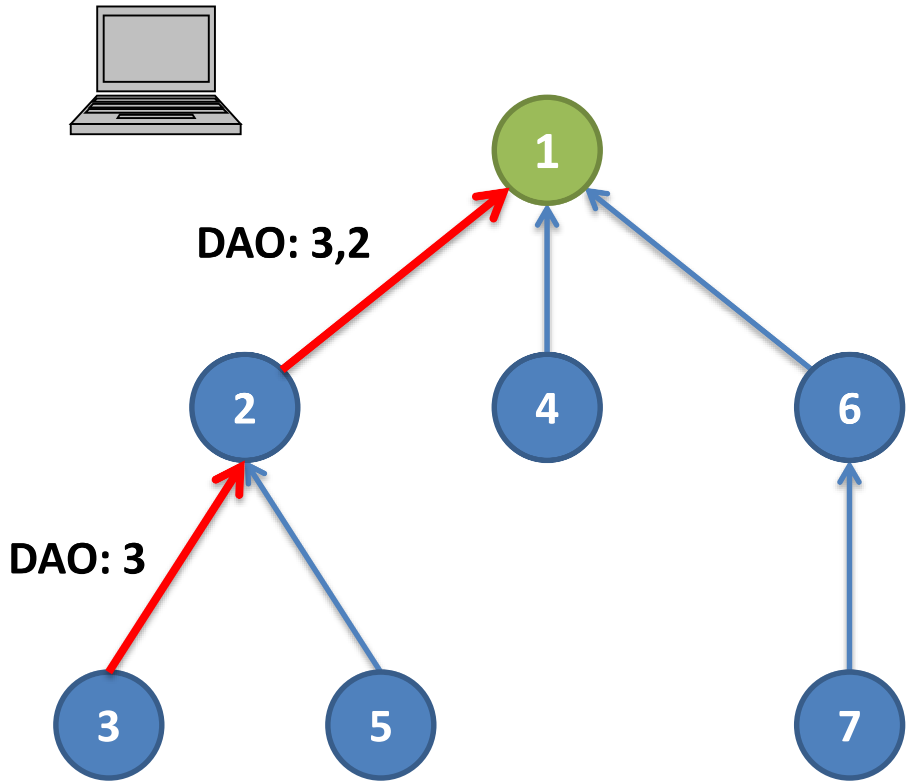
RPL

Remember, IoT → Down Traffic
Down Traffic needs routing tables



RPL

Remember, IoT → Down Traffic
Down Traffic needs routing tables

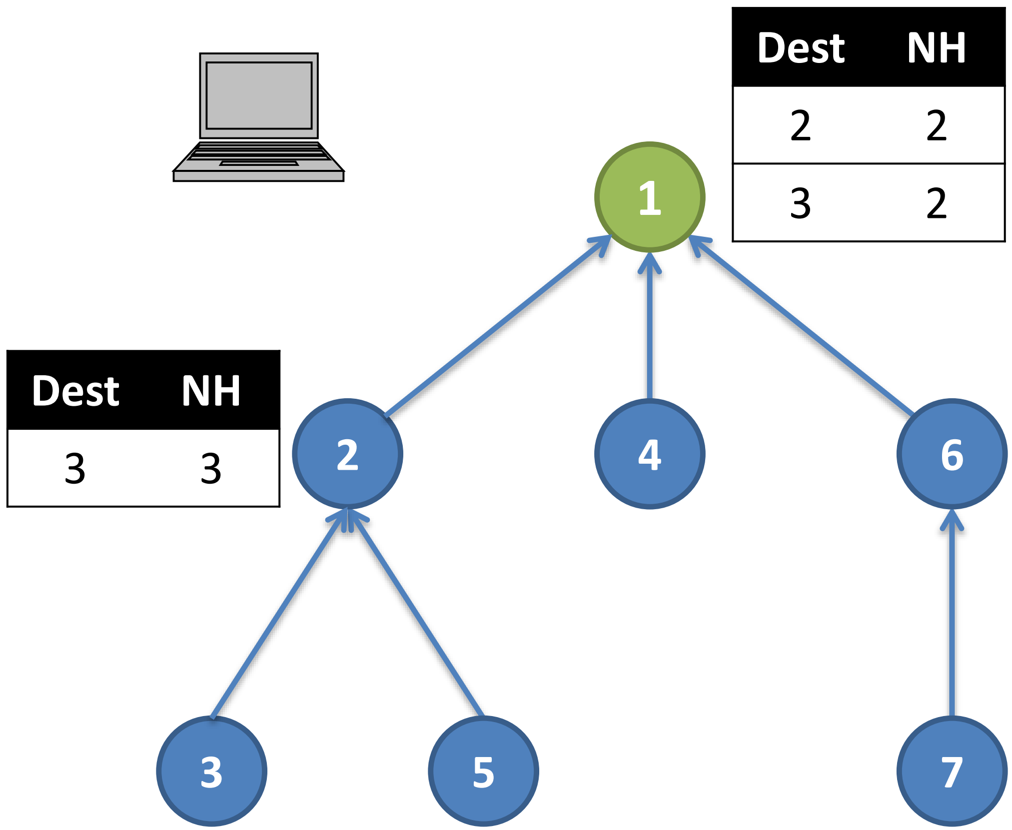


- Destination Advertisement (DAO)

RPL

Remember, IoT → Down Traffic

Down Traffic needs routing tables



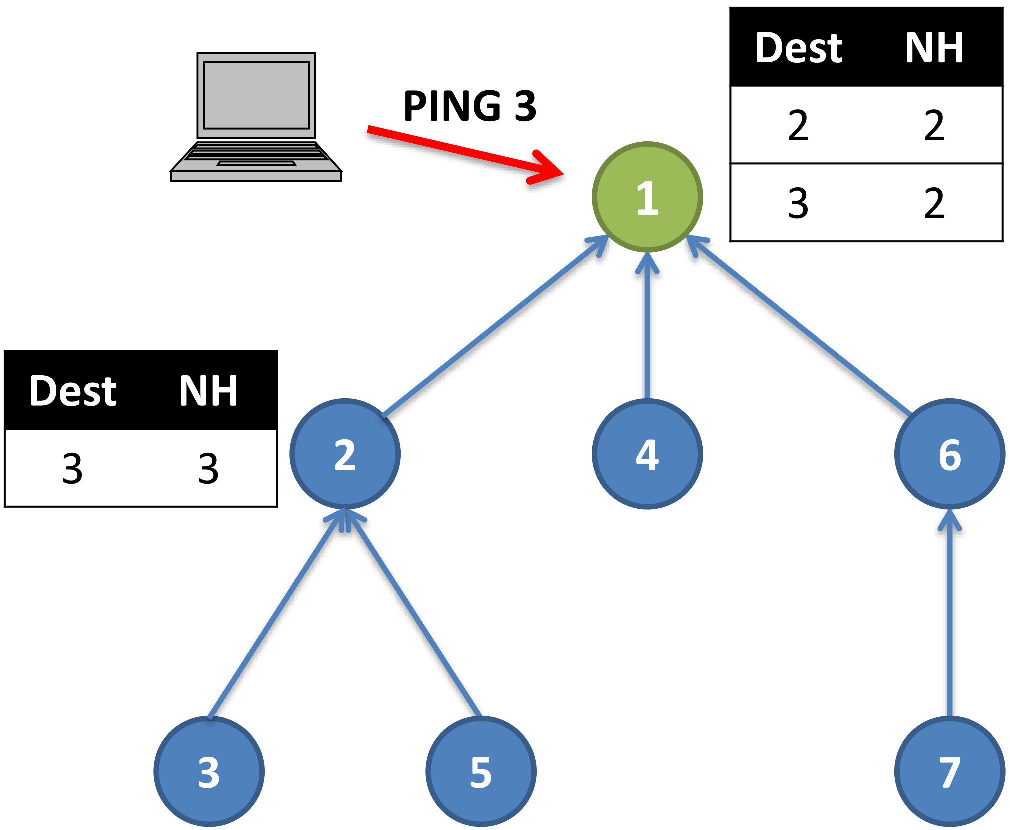
Dest	NH
2	2
3	2

Dest	NH
3	3

- Destination Advertisement (DAO)

RPL

Remember, IoT → Down Traffic
Down Traffic needs routing tables



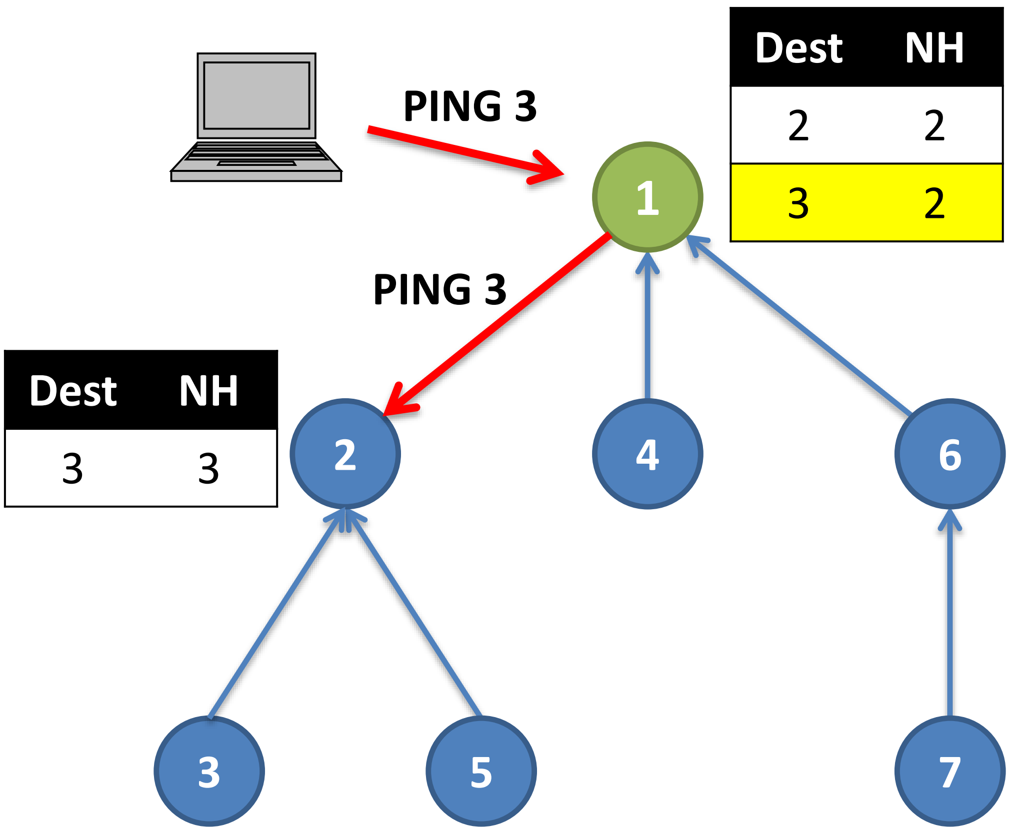
Dest	NH
2	2
3	2

Dest	NH
3	3

- Destination Advertisement (DAO)

RPL

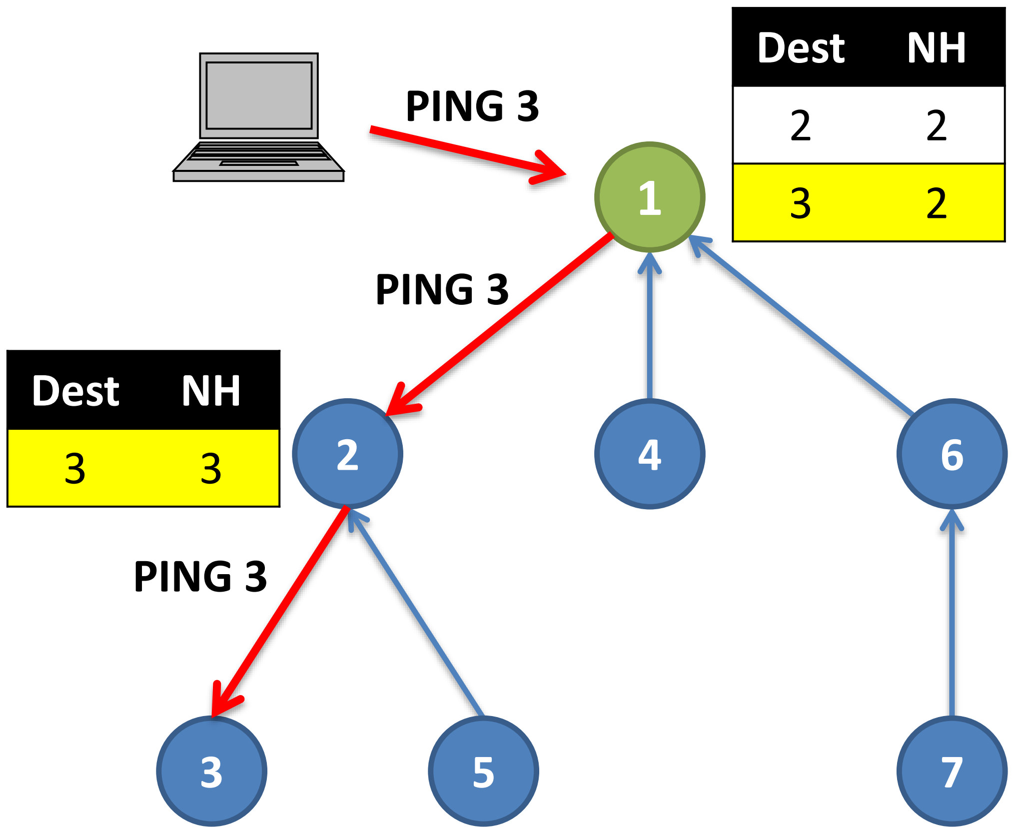
Remember, IoT → Down Traffic
Down Traffic needs routing tables



- Destination Advertisement (DAO)

RPL

Remember, IoT → Down Traffic
Down Traffic needs routing tables



Dest	NH
2	2
3	2

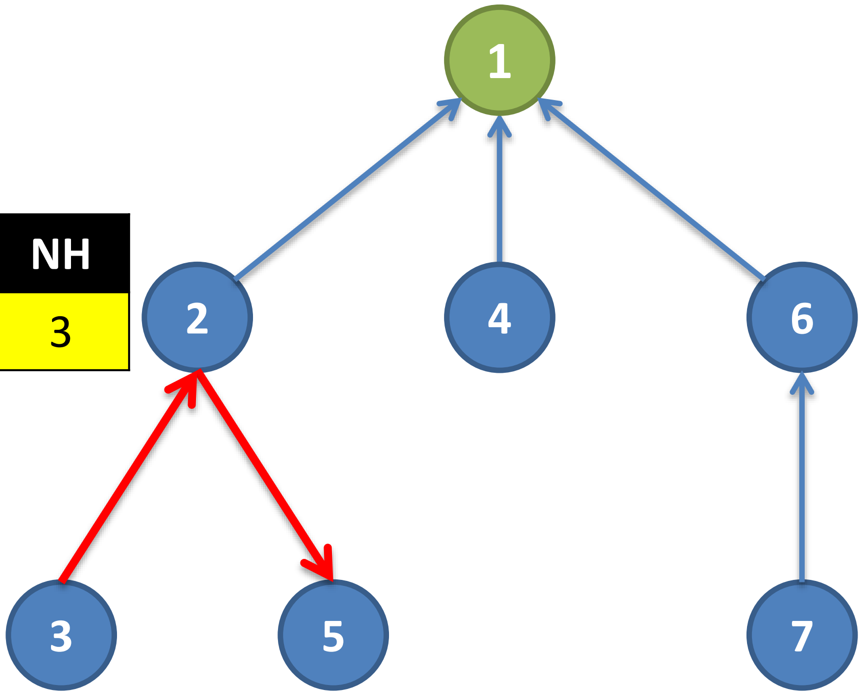
Dest	NH
3	3

- Destination Advertisement (DAO)

RPL

P2P traffic is possible

Dest	NH
3	3

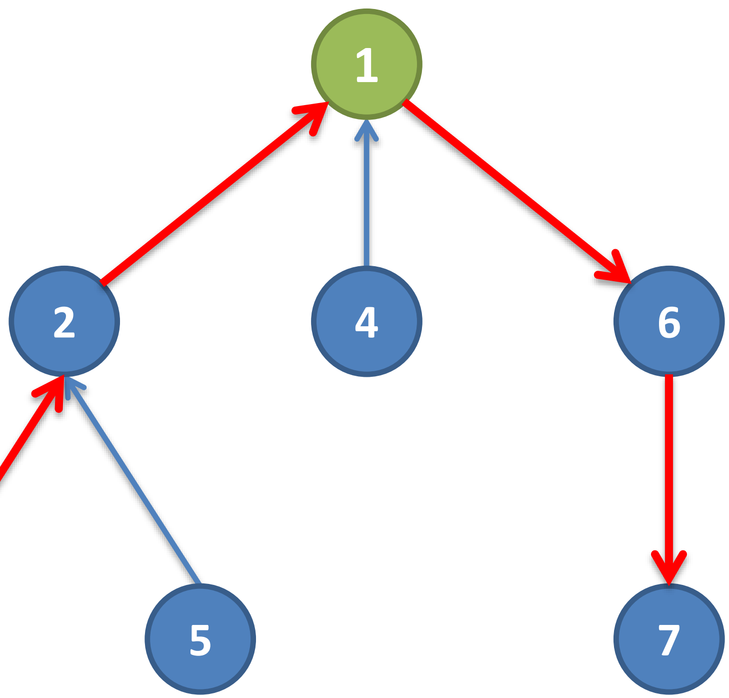


RPL

P2P traffic is possible

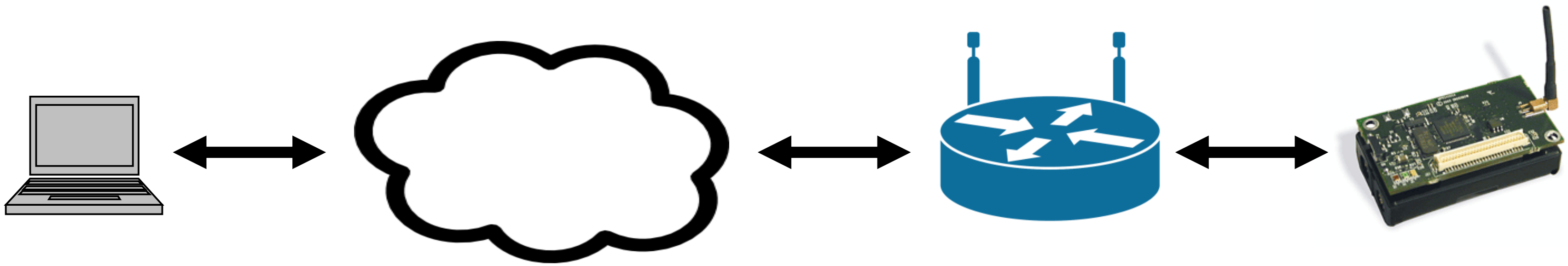
Dest	NH
2	2
3	2
7	6
...	
5	2

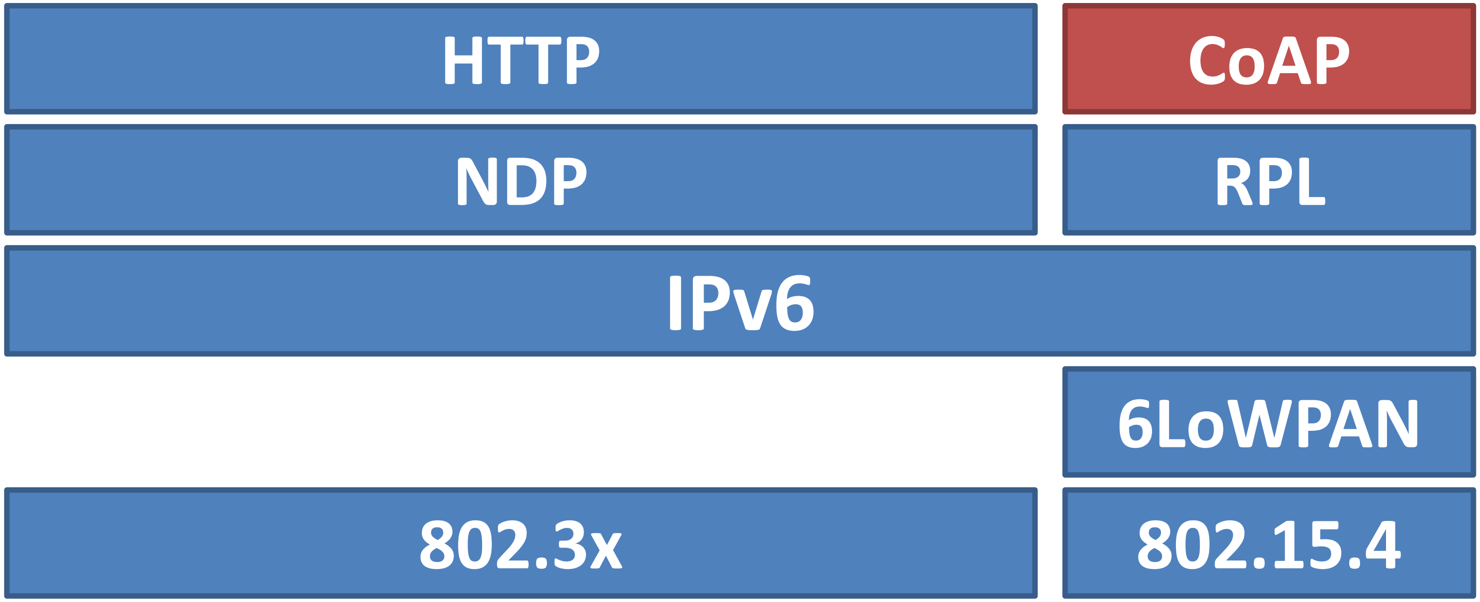
Dest	NH
3	3



Dest	NH
7	7

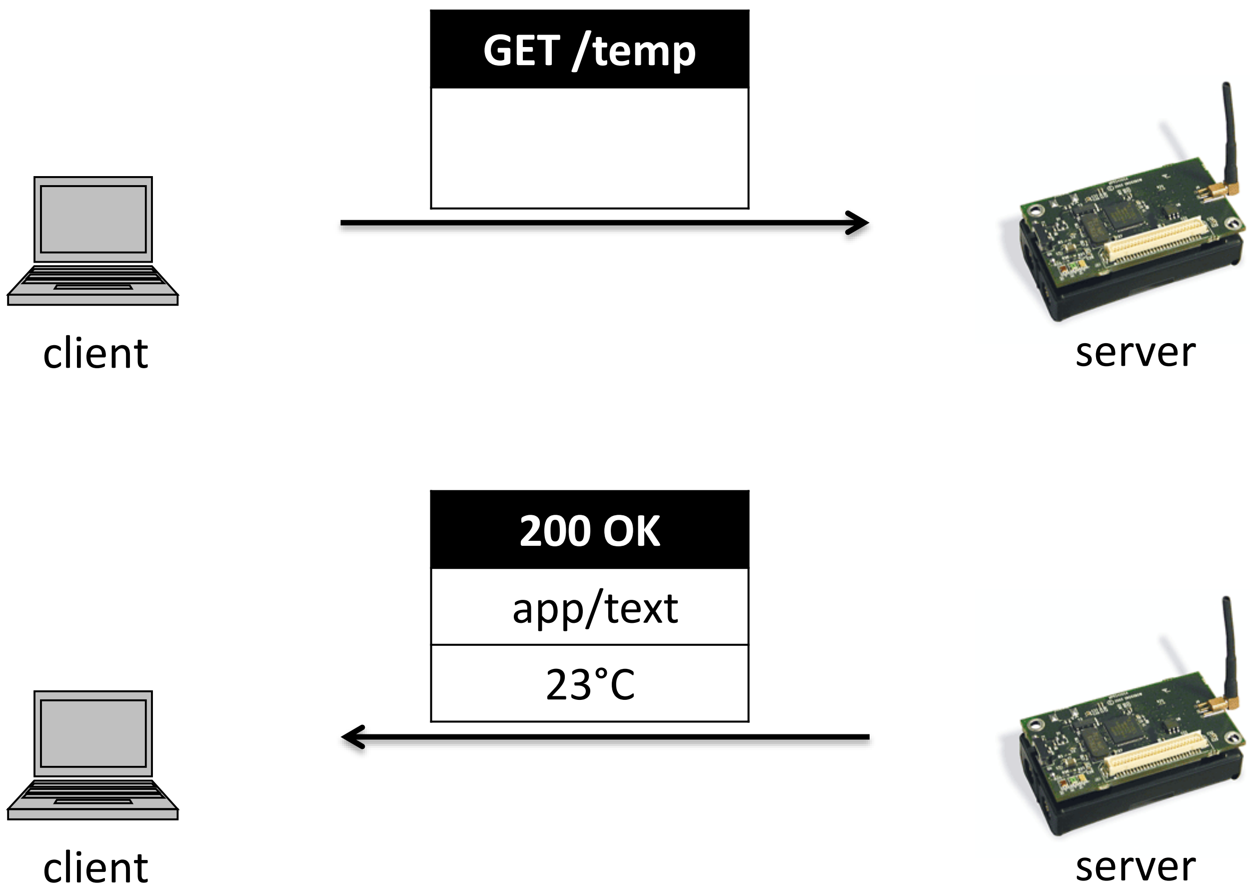
IPv6: the Narrow Waist



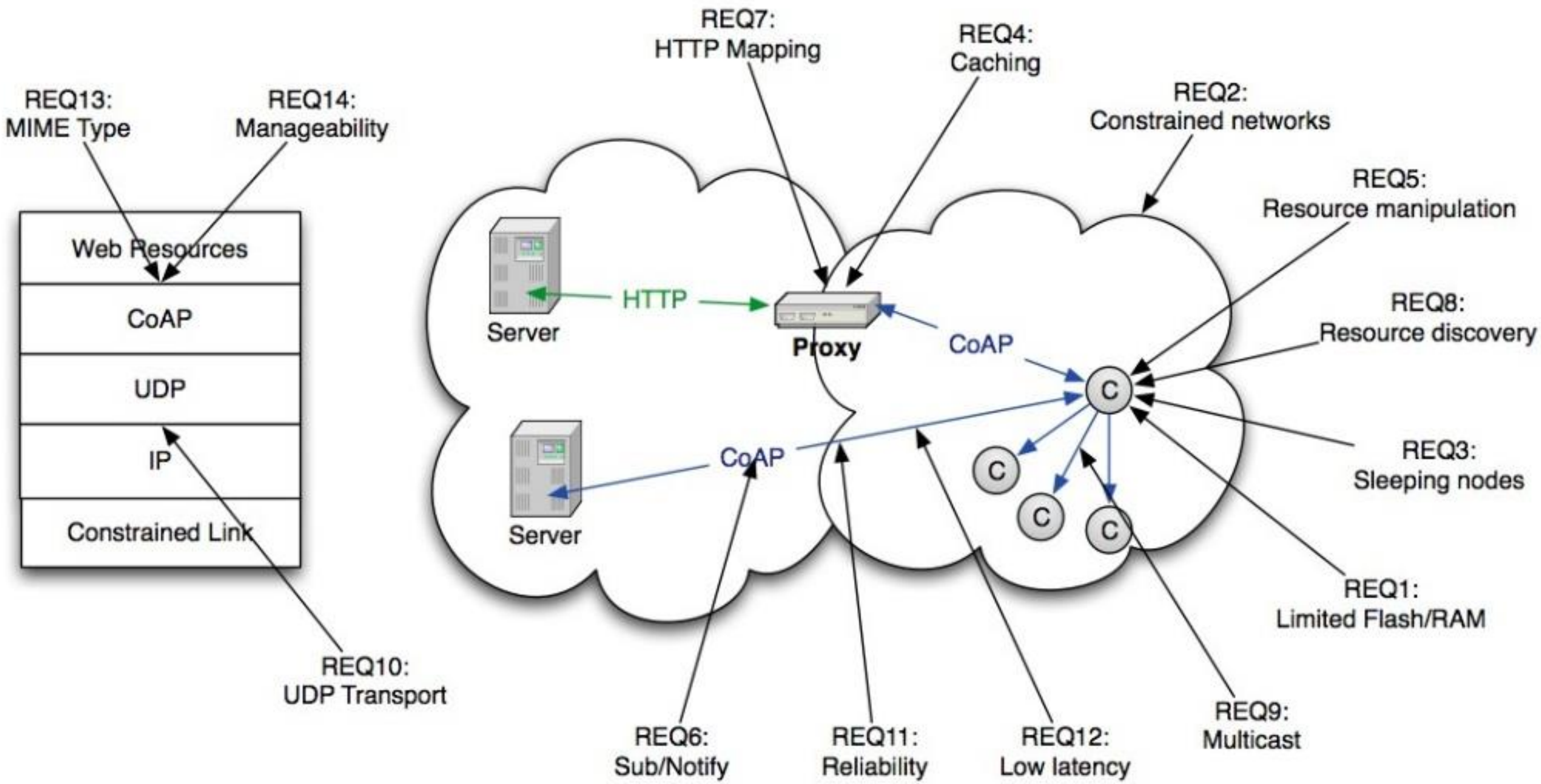


CoAP

A RESTful protocol

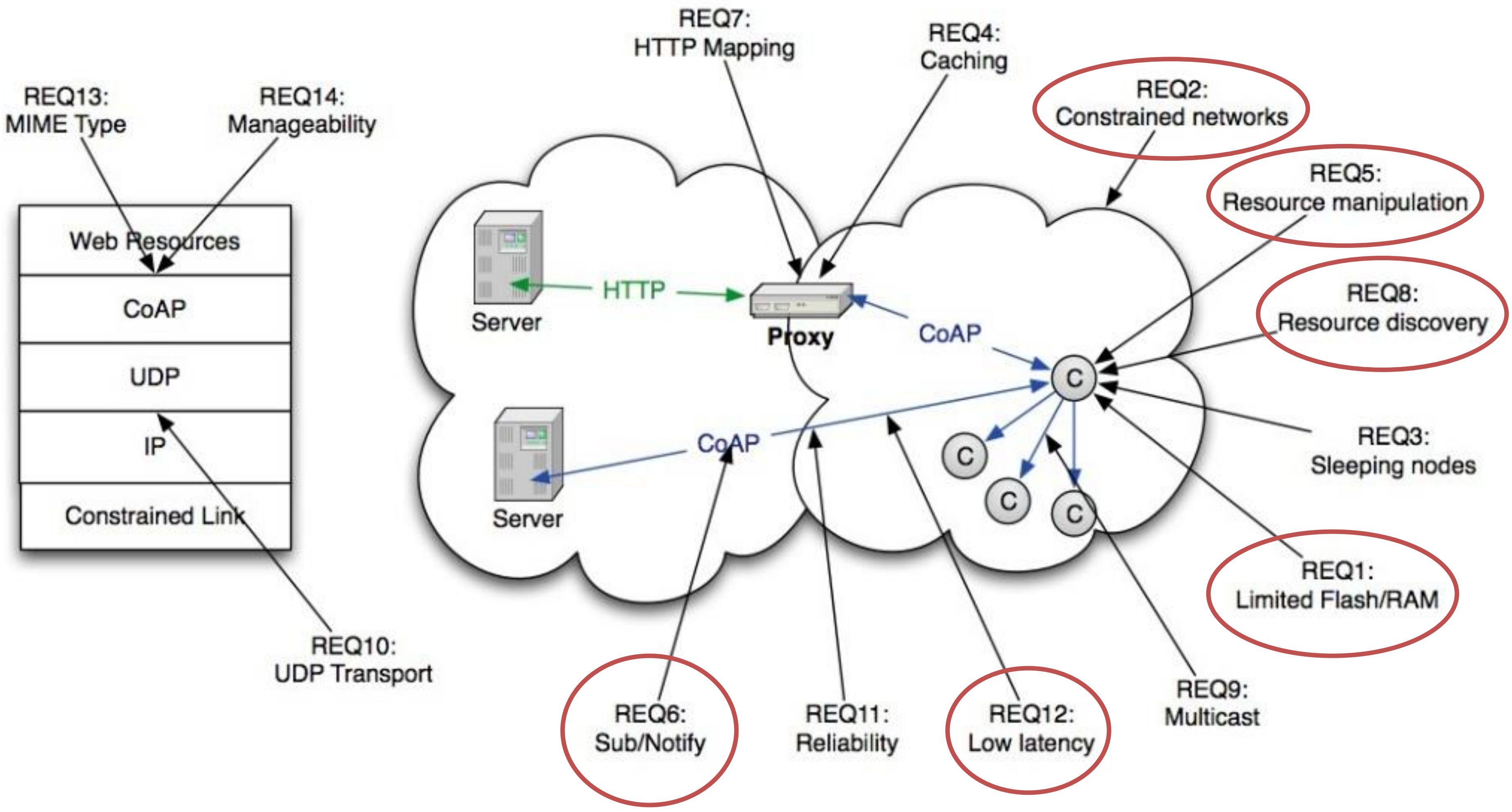


CoAP Design Requirements

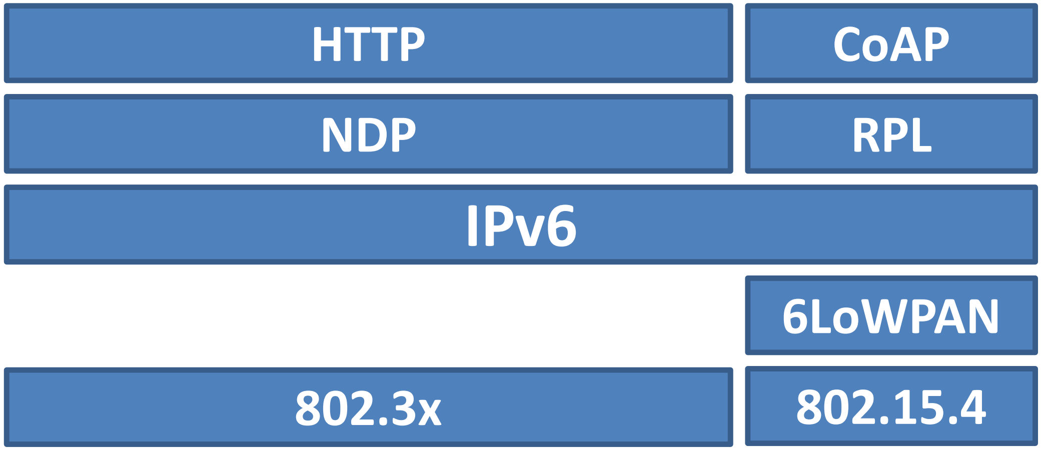


http://fr.slideshare.net/zdshelby/coap-tutorial

CoAP: Time for a Demo



http://fr.slideshare.net/zdshelby/coap-tutorial



Putting it all together

with

Contiki

www.contiki-os.org

www.github.com/contiki-os/contiki

2001

uIP

Contiki

uIP:

- TCP/IP stack for 8 & 16-bit µC
- Single buffer
- Standalone until v0.9
- Tiny
  - Code size < 5k
  - RAM ~1k (= mostly the packet buffer)

Contiki

2001 **2003-2004**

uIP **Contiki, Prototthreads**

Protothreads

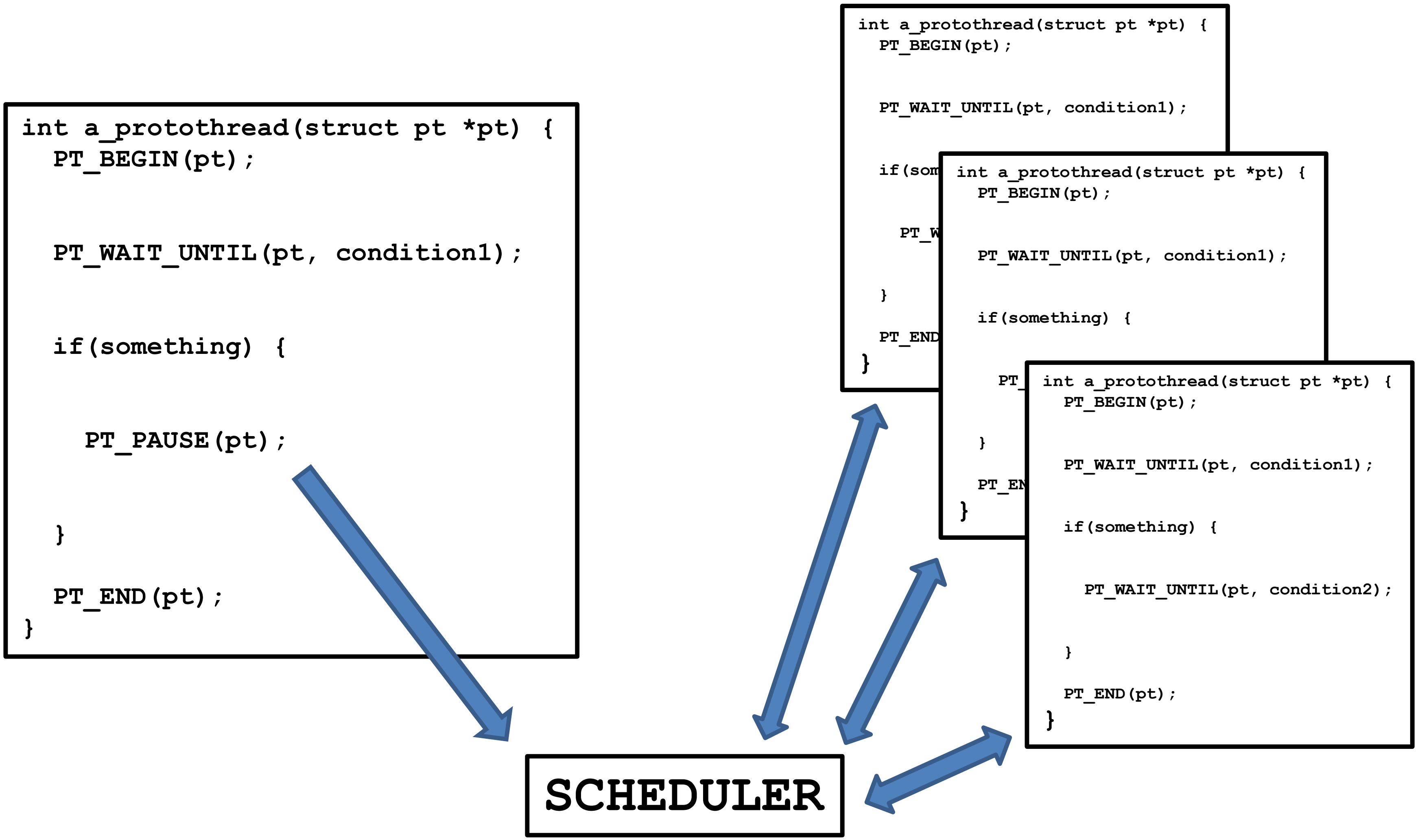
- Non pre-emptive systems
- Event-Driven model (TinyOS)
  + Good for interfacing with hardware & reactive processing
  - Difficult to sequence high-level operations, needs complex state machines
  - Lots of functions → overhead
- Protothreads (Contiki)
  + Linear code execution for event-driven systems
  + Low overhead (stackless threads)

Protothreads: wait for events

```
int a_protothread(struct pt *pt) {
  PT_BEGIN(pt);

  /* … */

  PT_WAIT_UNTIL(pt, condition1);

  /* … */

  if(something) {

    /* … */

    PT_WAIT_UNTIL(pt, condition2);

    /* … */

  }

  PT_END(pt);
}
```

http://dunkels.com/adam/dunkels06protothreads.ppt

Protothreads: yielding



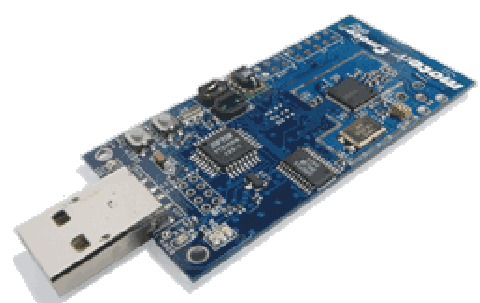
Protothreads

Bottom line

=

Let’s you program sequentially with multi-threading in C

On



Contiki

2001 **2003-2004** **2007**

uIP

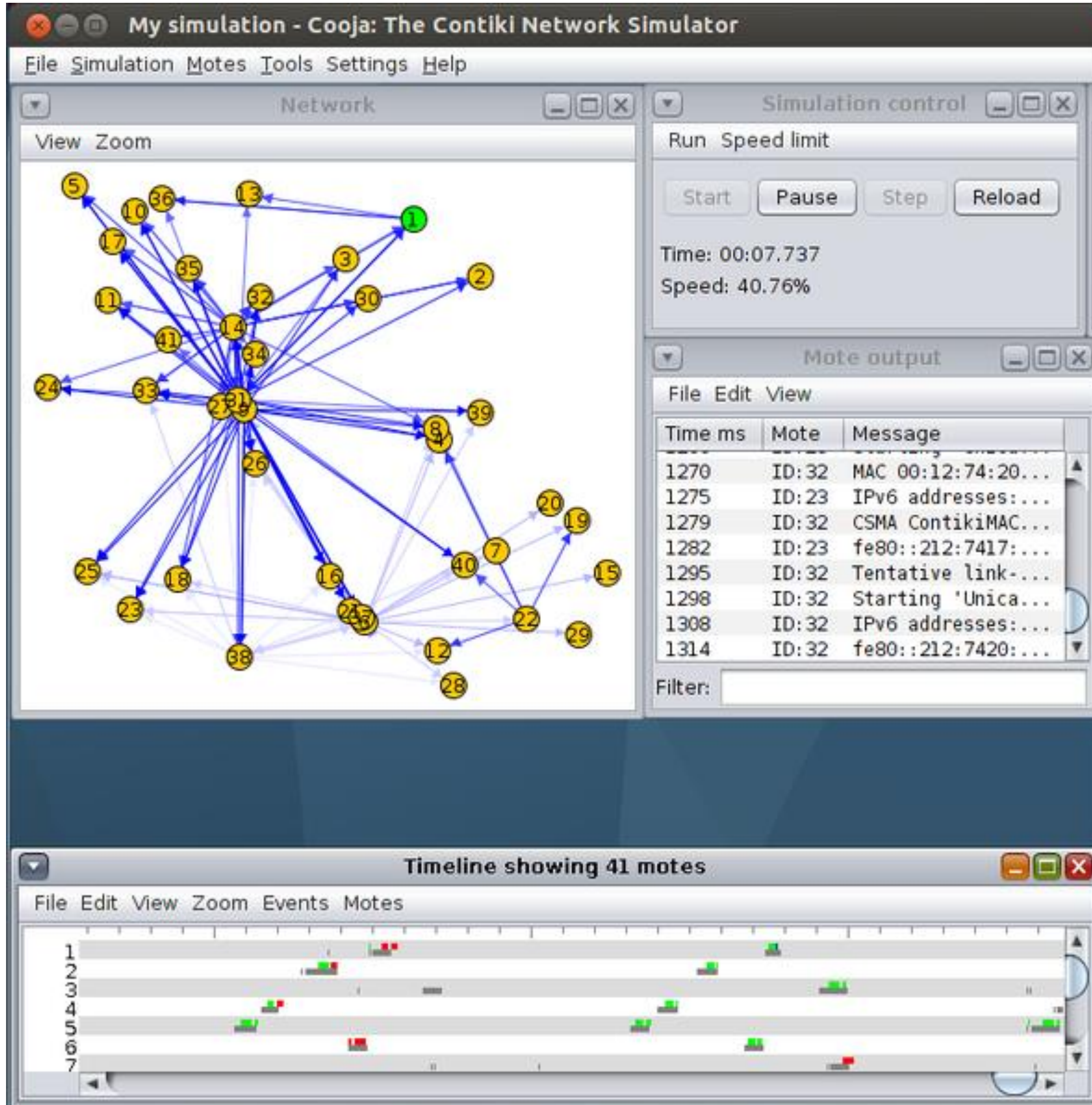
Contiki, Protothreads

COOJA, Power Profiling

COOJA

Run bytecode on emulated HW

Time-accurate WSN simulation



Contiki

2001 **2003-2004** **2007**

uIP

Contiki, Protothreads

COOJA, Power Profiling

2008

SICSLoWPAN

Contiki

2001

uIP

2003-2004

Contiki,
Protothreads

2007

COOJA,
Power Profiling

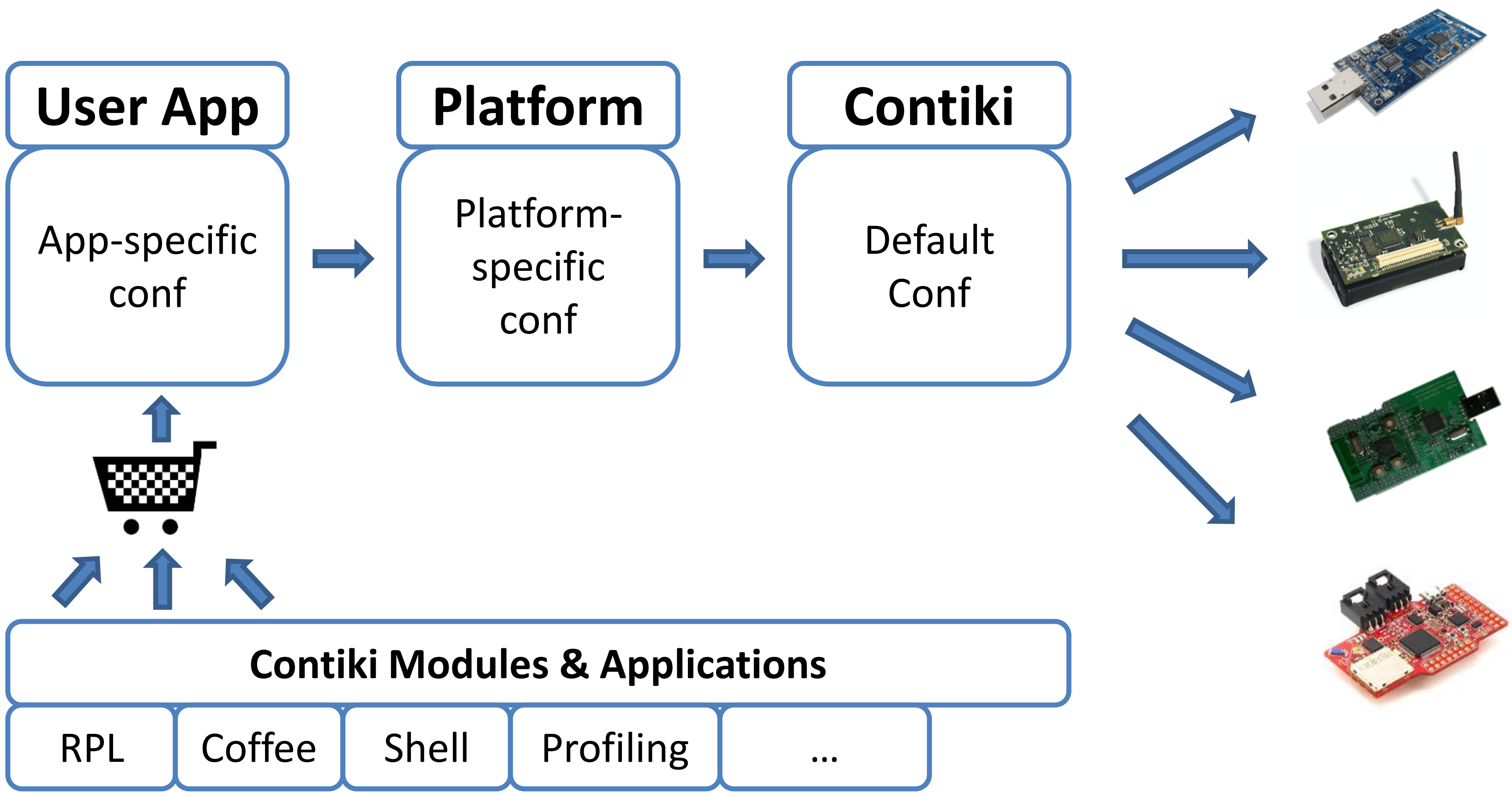
2008

SICSLoWPAN

2009-2010

Many more
Platforms

Multi-Platform Build System



Contiki

2001 — **2003-2004** — **2007**

uIP

Contiki, Protothreads

COOJA, Power Profiling

2008 — **2009-2010** — **2011**

SICSLoWPAN

Many more Platforms

ContikiRPL
ContikiMAC

ContikiMAC: Sleepy Radios

- Radios are power-hungry
- Current Draw of RX +/- same as TX

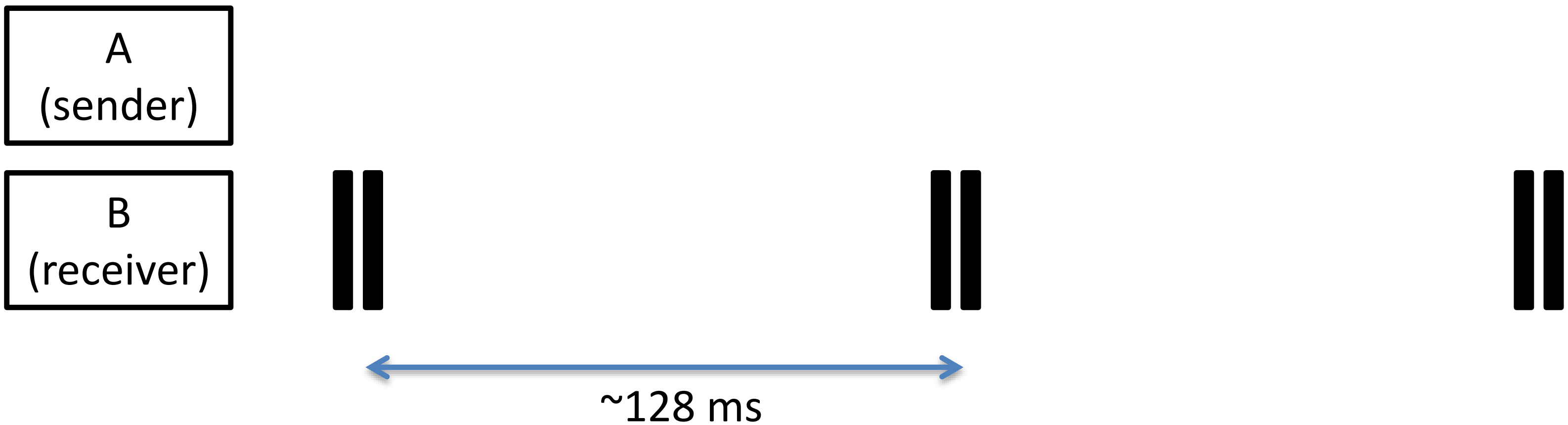
→ContikiMAC: Duty Cycle the Radio

- Builds on previous RDC protocols
- No synchronization
- Supporting Async communications
- < 1% Duty Cycle achieved

http://dunkels.com/adam/dunkels11contikimac.pdf

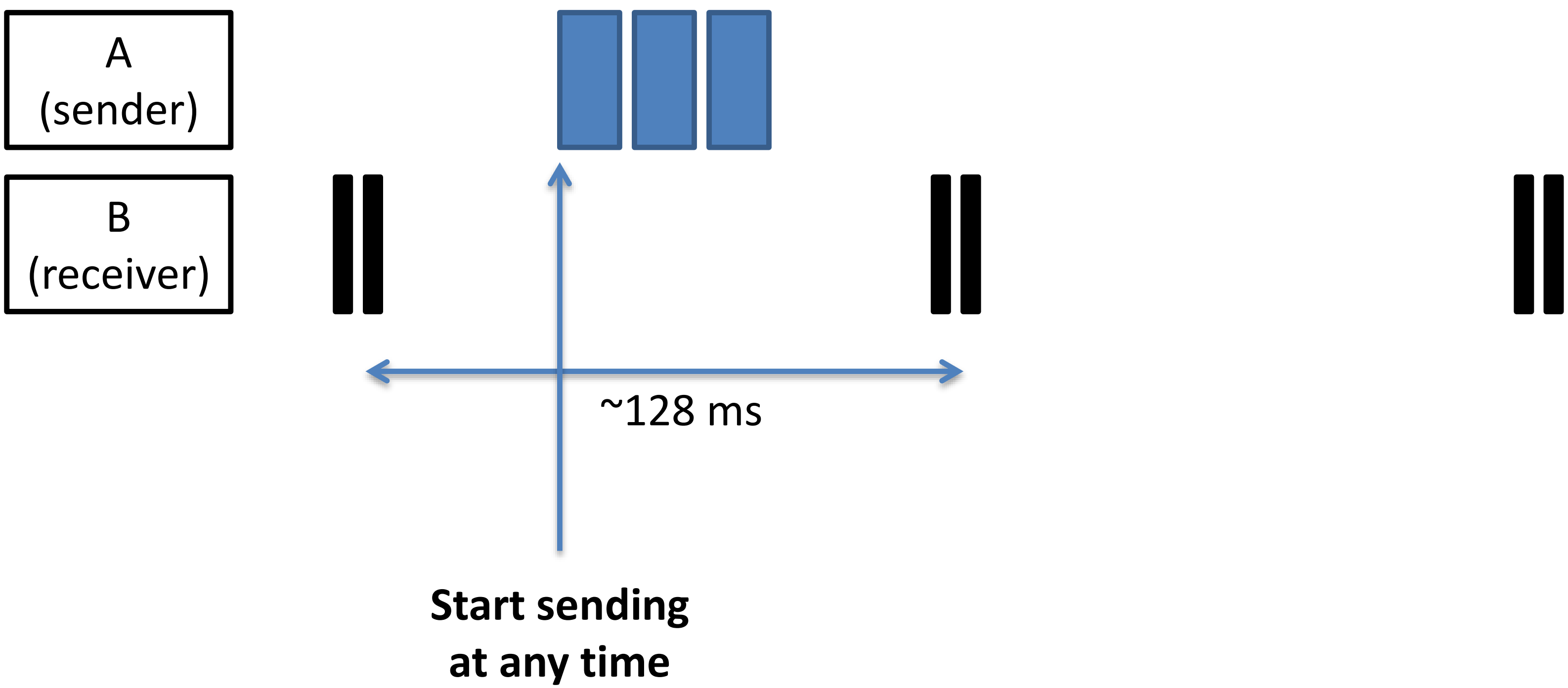
ContikiMAC: Sleepy Radios

Sleep, wake up periodically



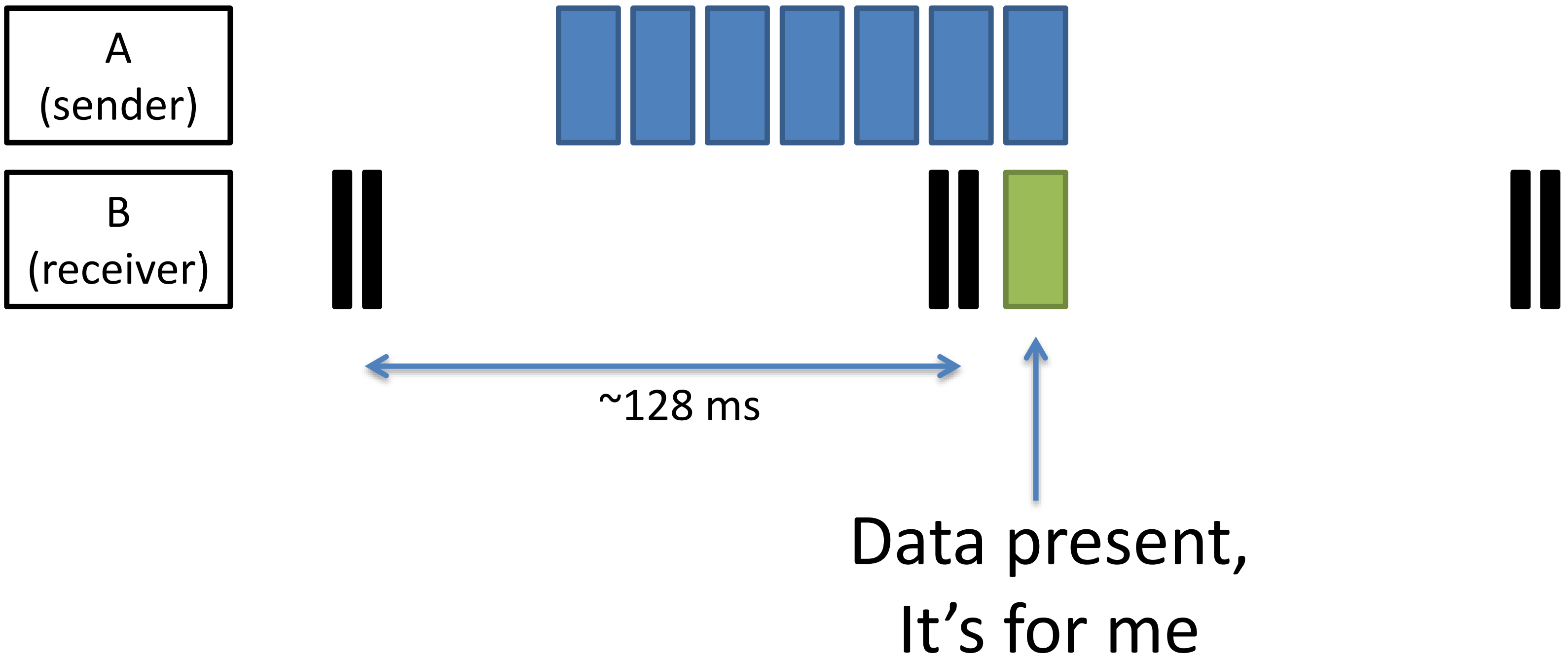
ContikiMAC: Sleepy Radios

Unicasts: strobe until ACK'd



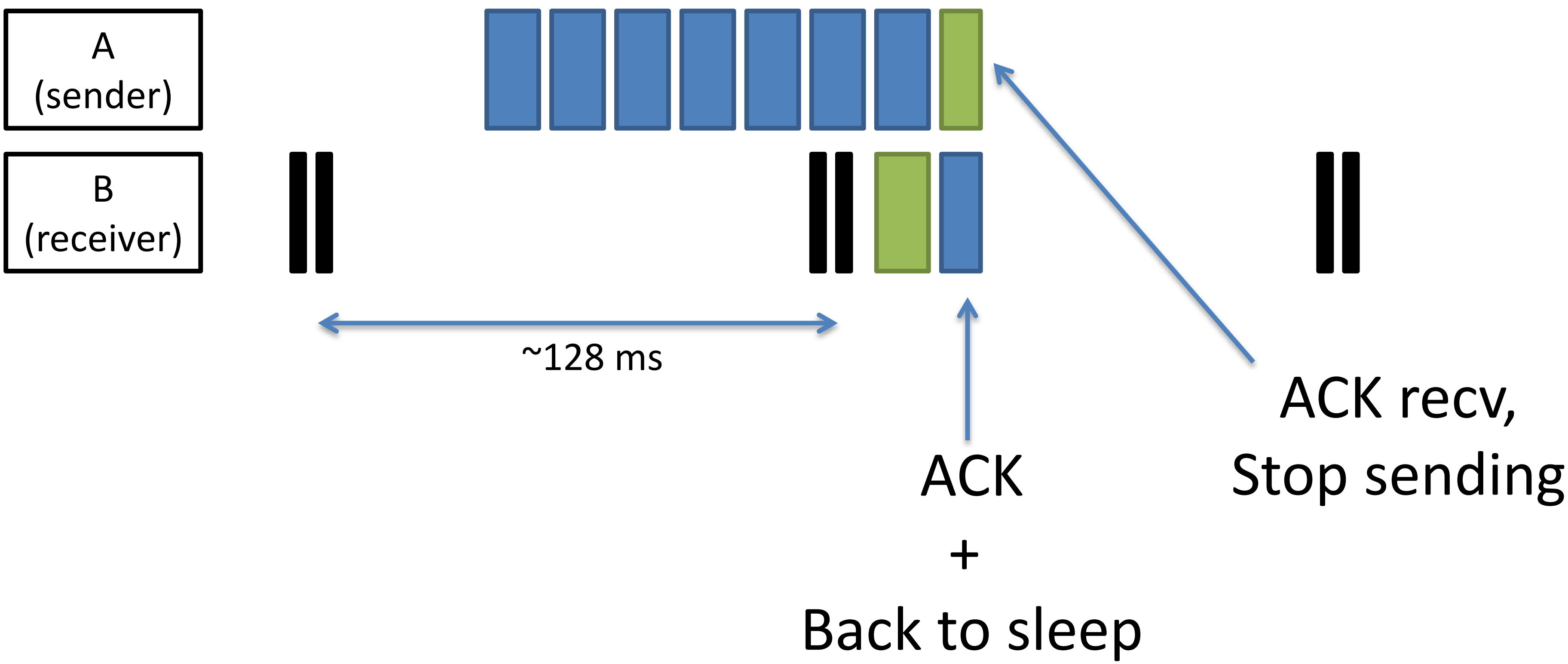
ContikiMAC: Sleepy Radios

Unicasts: strobe until ACK'd



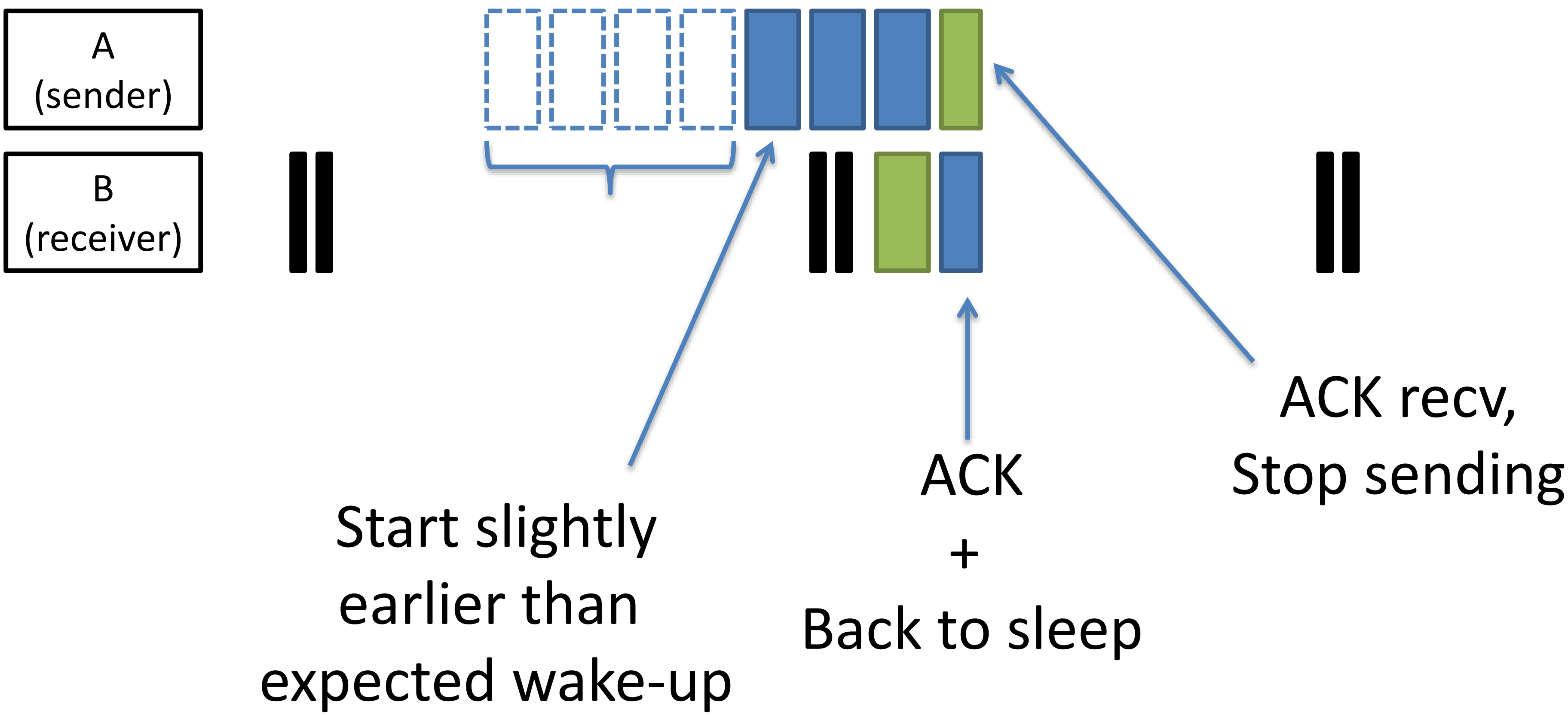
ContikiMAC: Sleepy Radios

Unicasts: strobe until ACK’d



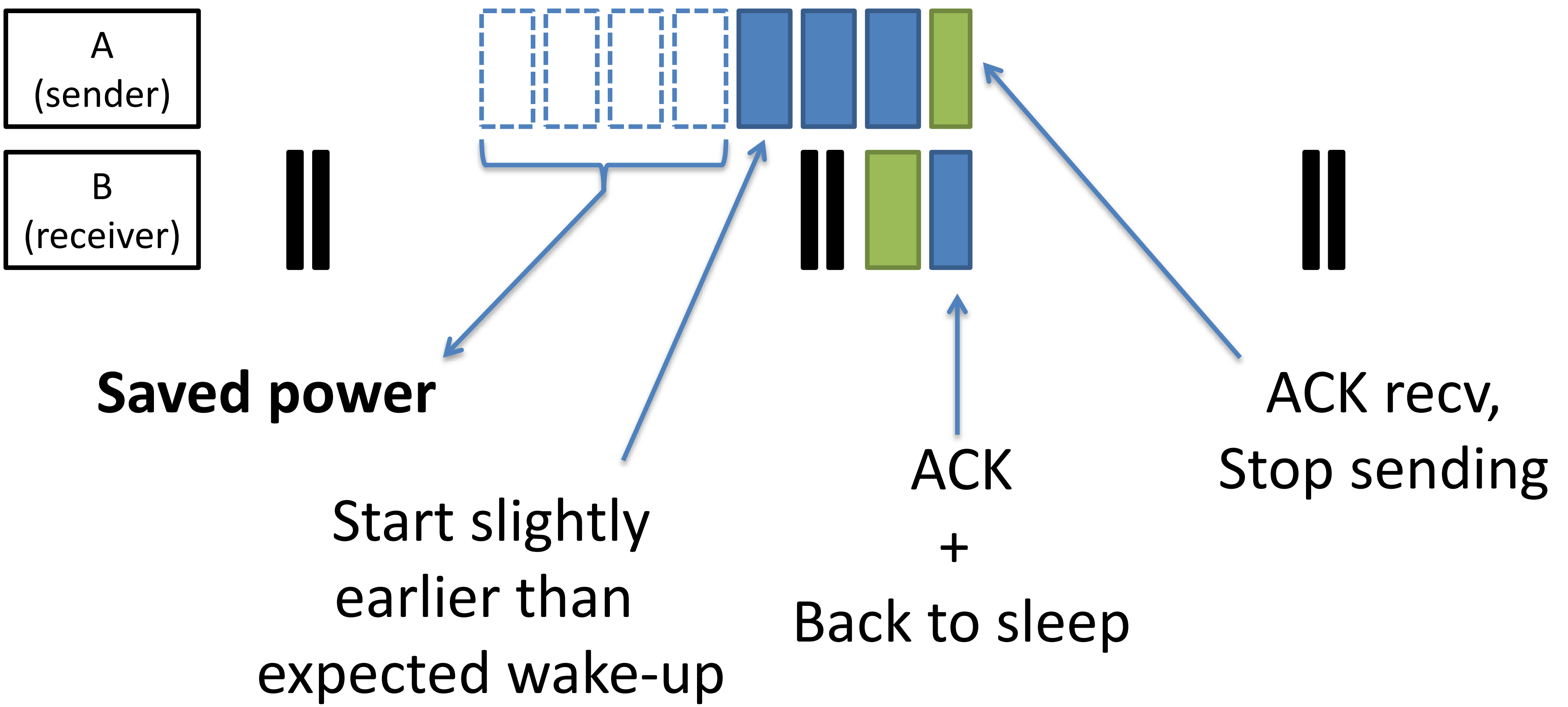
ContikiMAC: Sleepy Radios

Phase-Lock: anticipate wake-up by learning



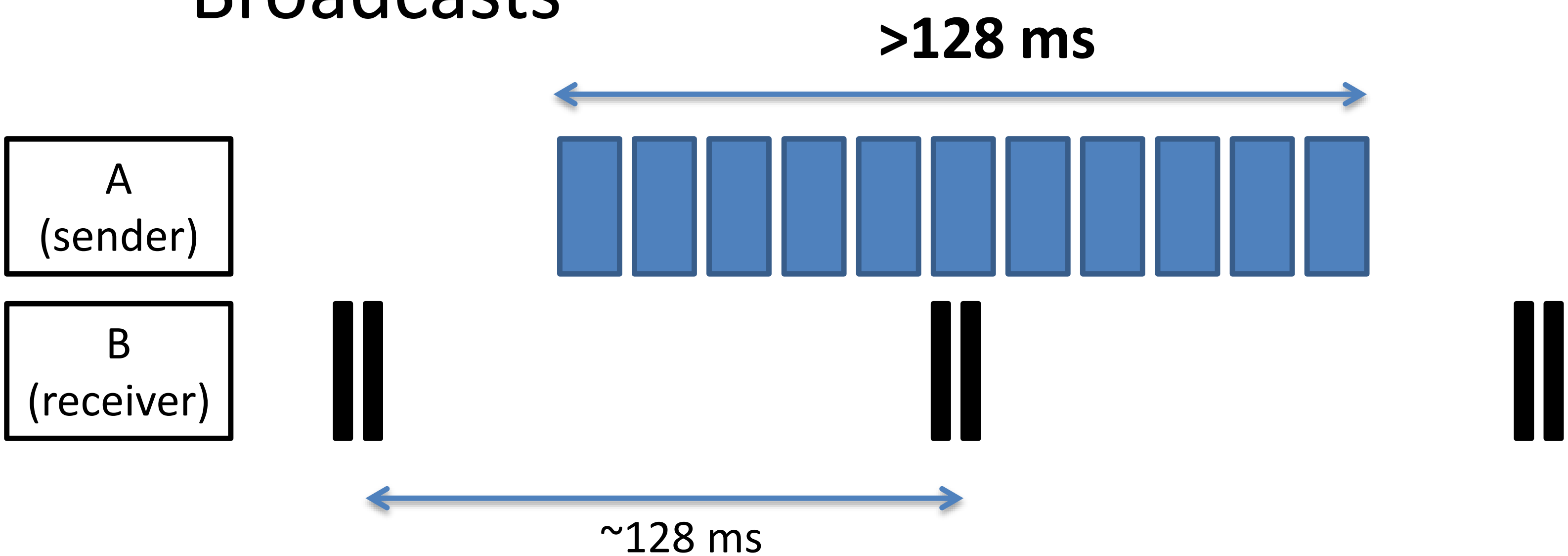
ContikiMAC: Sleepy Radios

Phase-Lock: anticipate wake-up by learning

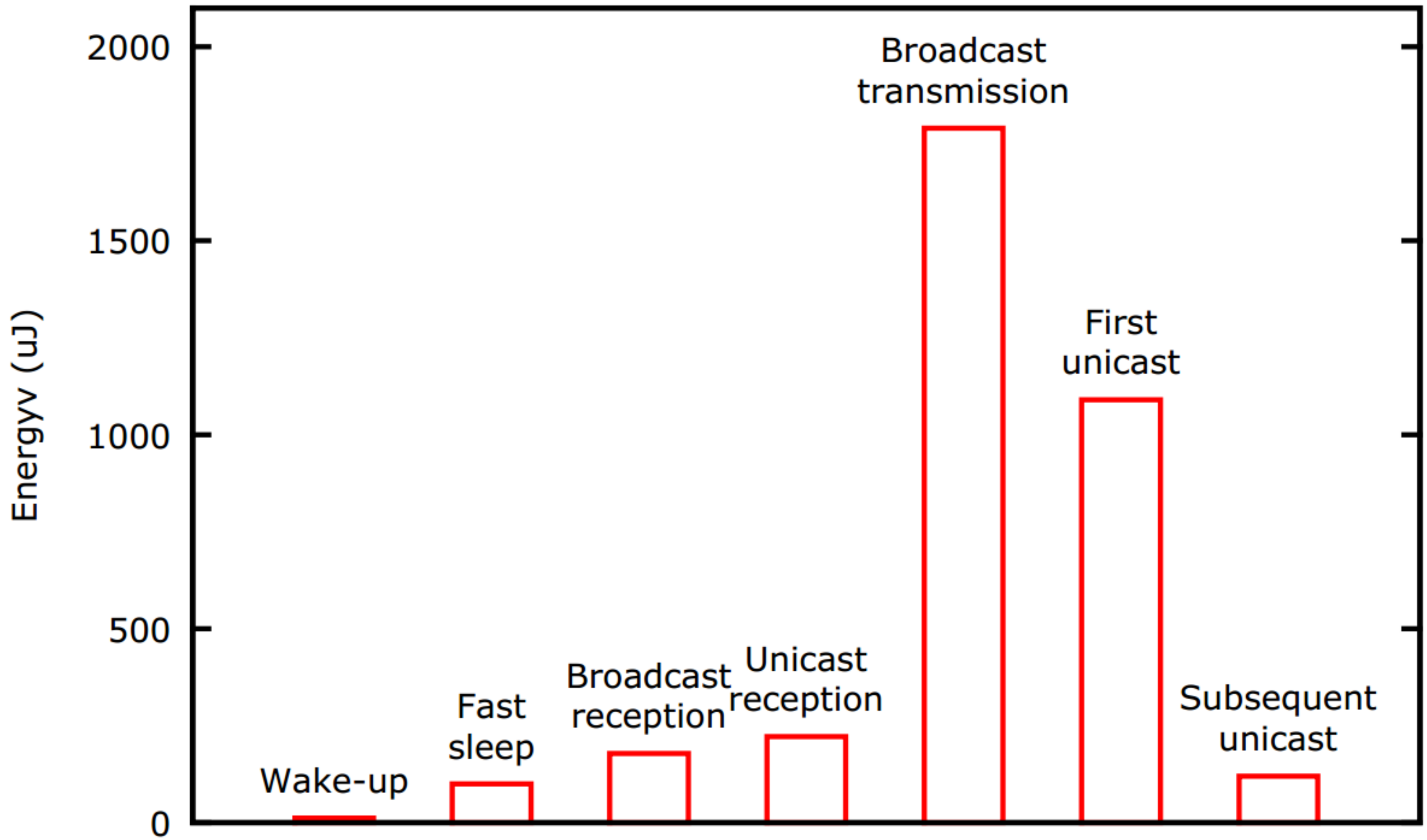


ContikiMAC: Sleepy Radios

Broadcasts



ContikiMAC



http://dunkels.com/adam/dunkels11contikimac.pdf

Contiki

2001 **2003-2004** **2007**

uIP

Contiki, Protothreads

COOJA, Power Profiling

2008 **2009-2010** **2011**

SICSLoWPAN

Many more Platforms

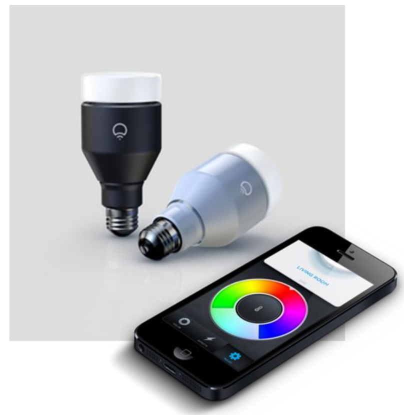
ContikiRPL
ContikiMAC

2012

Connect to the Cloud: Thingsquare

Contiki-based Projects

www.tado.com



www.lifx.co

Redwire self-powered sensor

http://www.redwirellc.com/

http://www.dtectsystems.com

http://www.noolitic.com/

Contiki

2001 — **uIP**

2003-2004 — **Contiki, Protothreads**

2007 — **COOJA, Power Profiling**

2008 — **SICSLoWPAN**

2009-2010 — **Many more Platforms**

2011 — **ContikiRPL ContikiMAC**

2012 — **Connect to the Cloud: Thingsquare**

2013 — **6LBR**

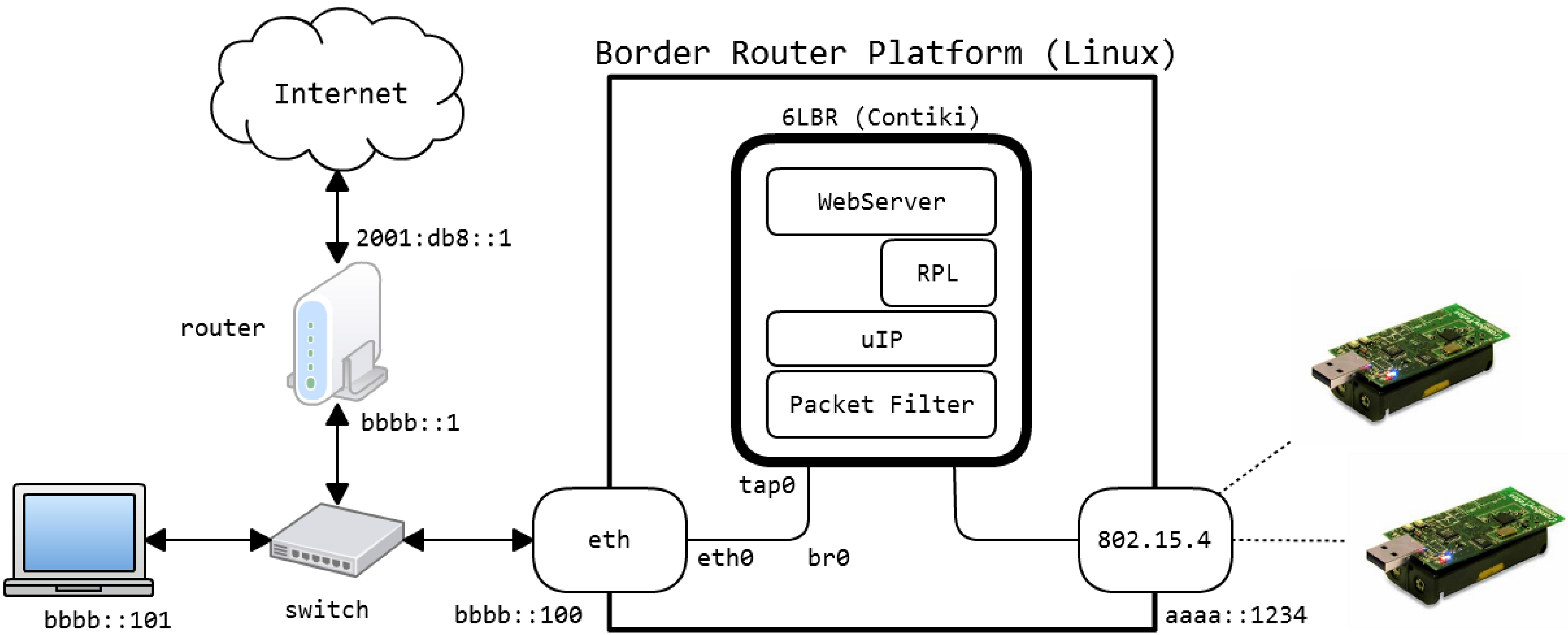
Ongoing Activities @ cetic

6LBR

6LBR: A Deployment-Ready Border Router for Internet of Things

CETIC (Laurent Deru, Sébastien Dawans)
UMons (Maxime Denis, Bruno Quoitin)

6LBR Overview



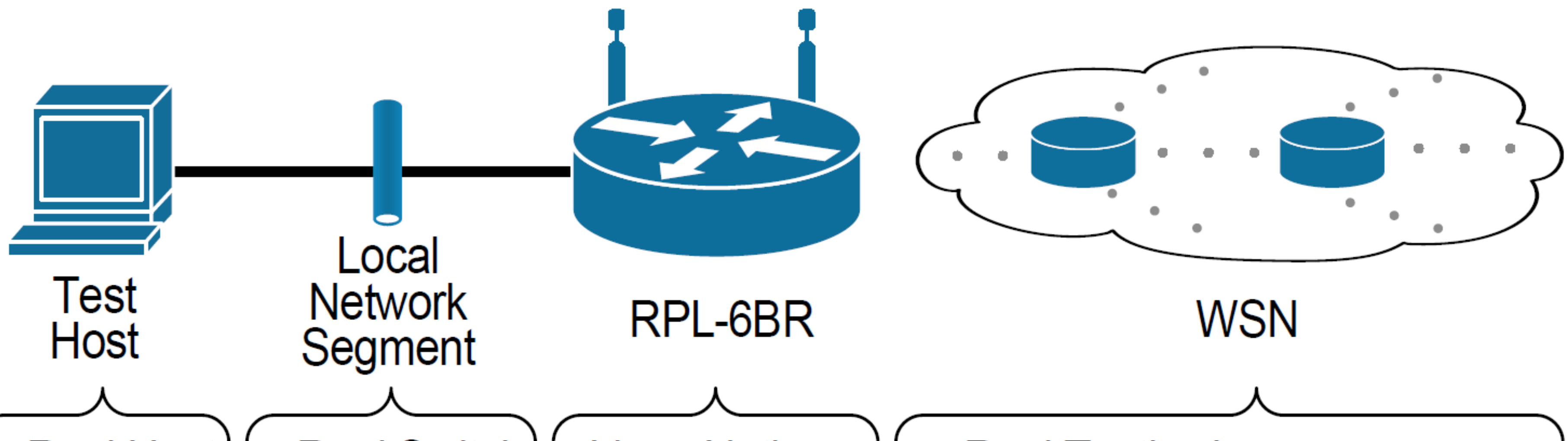
More info: https://github.com/cetic/6lbr/wiki/6LBR-Modes

6LBR Features

- Today
  - Seamless integration of IPv6 and RPL
  - Run smoothly on low-cost hardware
  - Multi-BR (reliability, traffic balancing)
  - Highly modular regression testing framework
- Ongoing
  - Multiple RPL Instances
  - Expand user base
  - Real-World Feedback

6LBR Demo

6LBR Test Framework



+ Real Host
+ VM
+ Local App

+ Real Switch
+ Linux Bridge

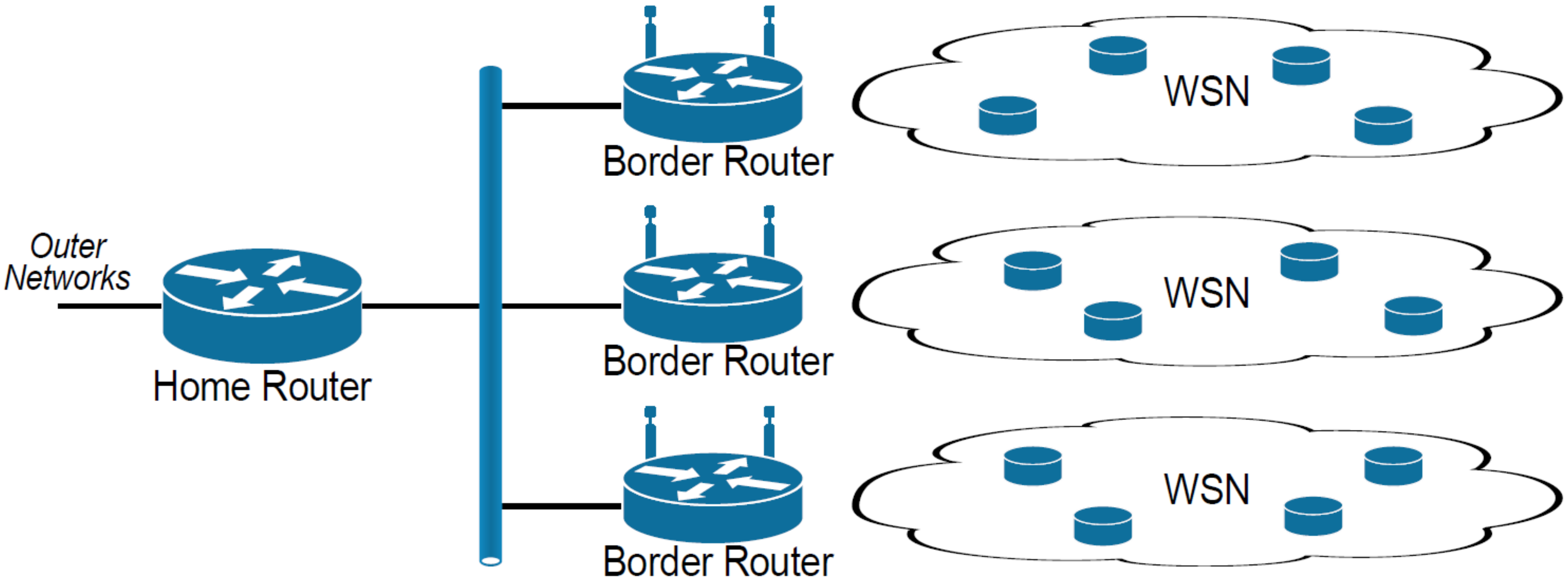
+ Linux Native
  - Std Computer
  - VM
  - Raspberry Pi
  - BeagleBone
+ Bare Metal
  - Econotag

+ Real Testbed
+ COOJA

http://inl.info.ucl.ac.be/system/files/deru13redundant.pdf

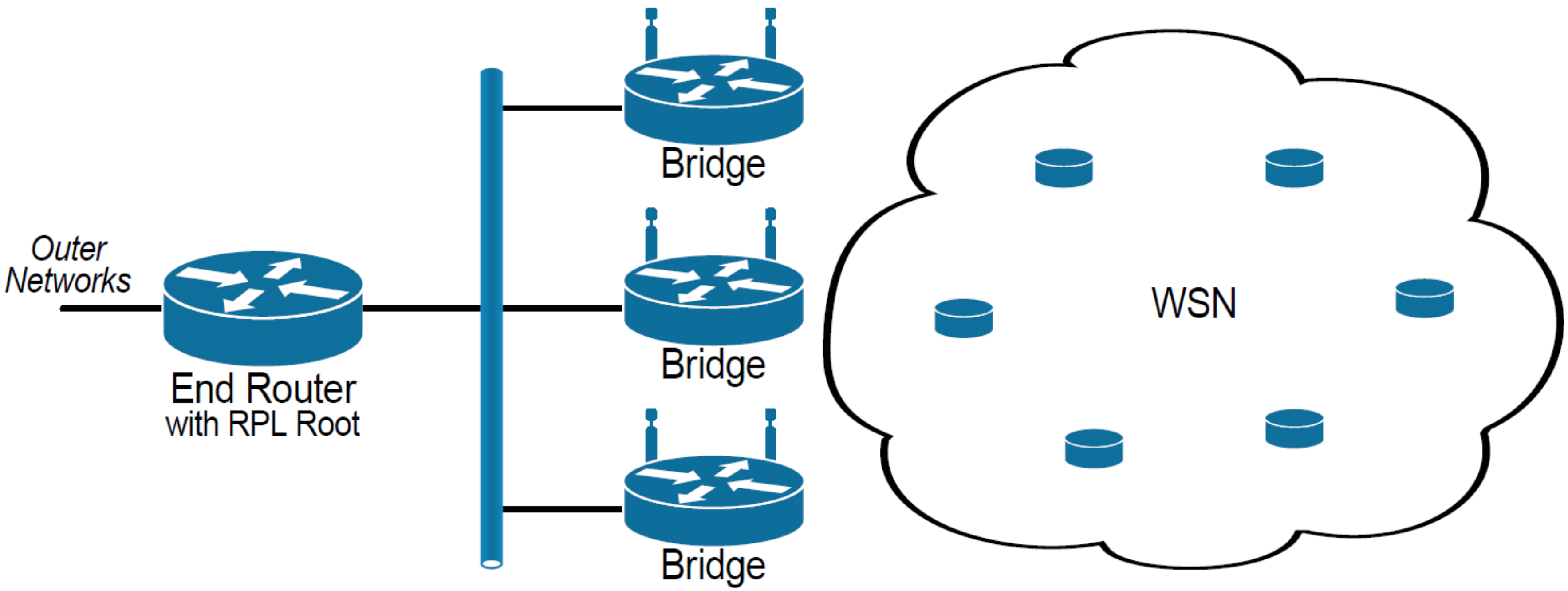
Multiple Border Routers

Spatial Diversity, Redundancy…

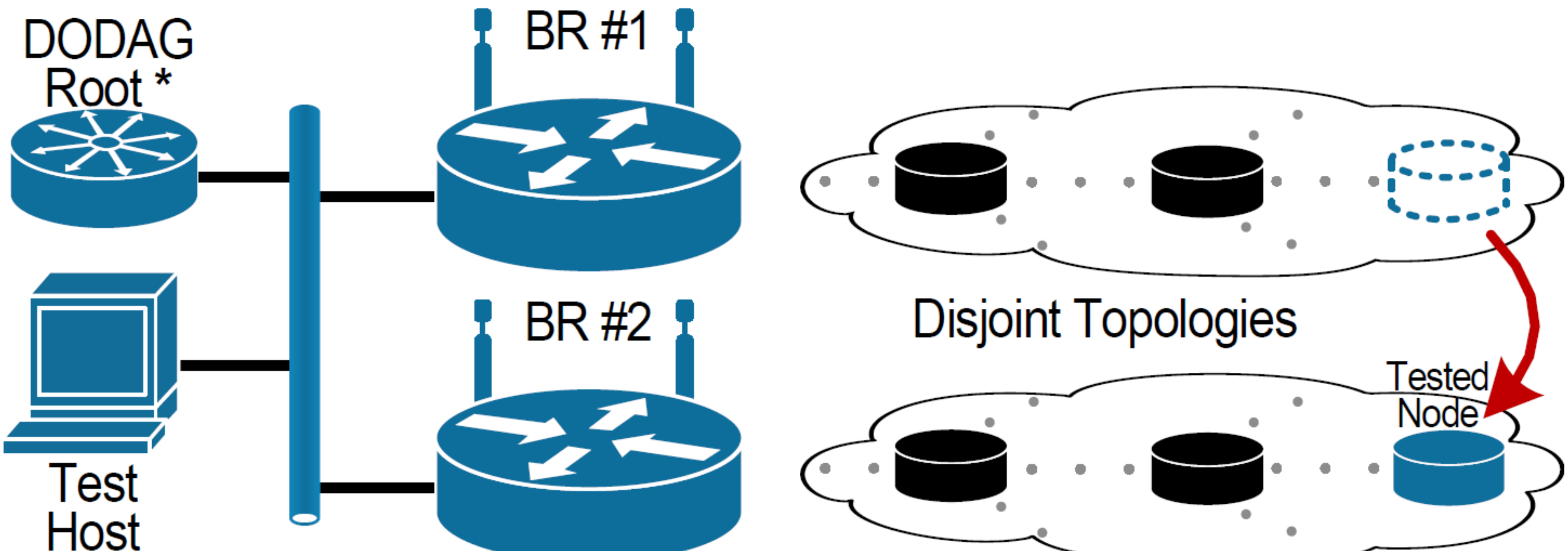


Multiple Border Routers

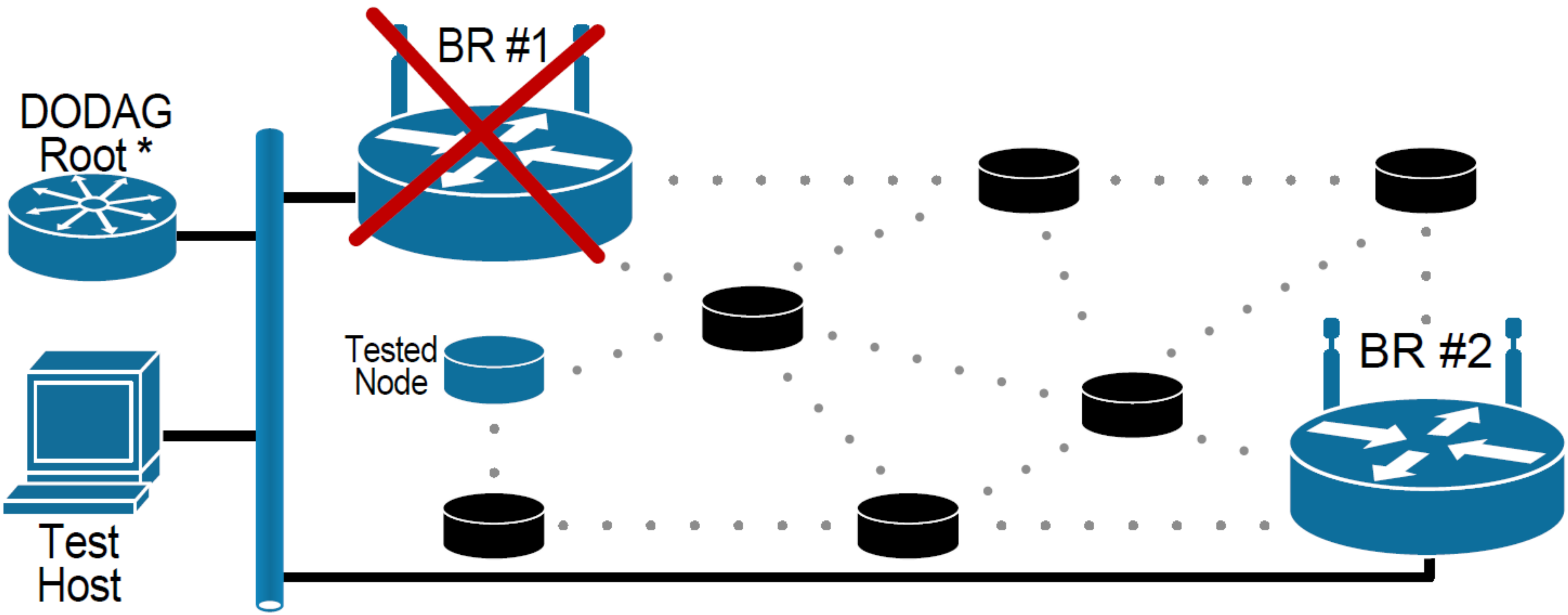
Aggregate the WSN with external RPL Root



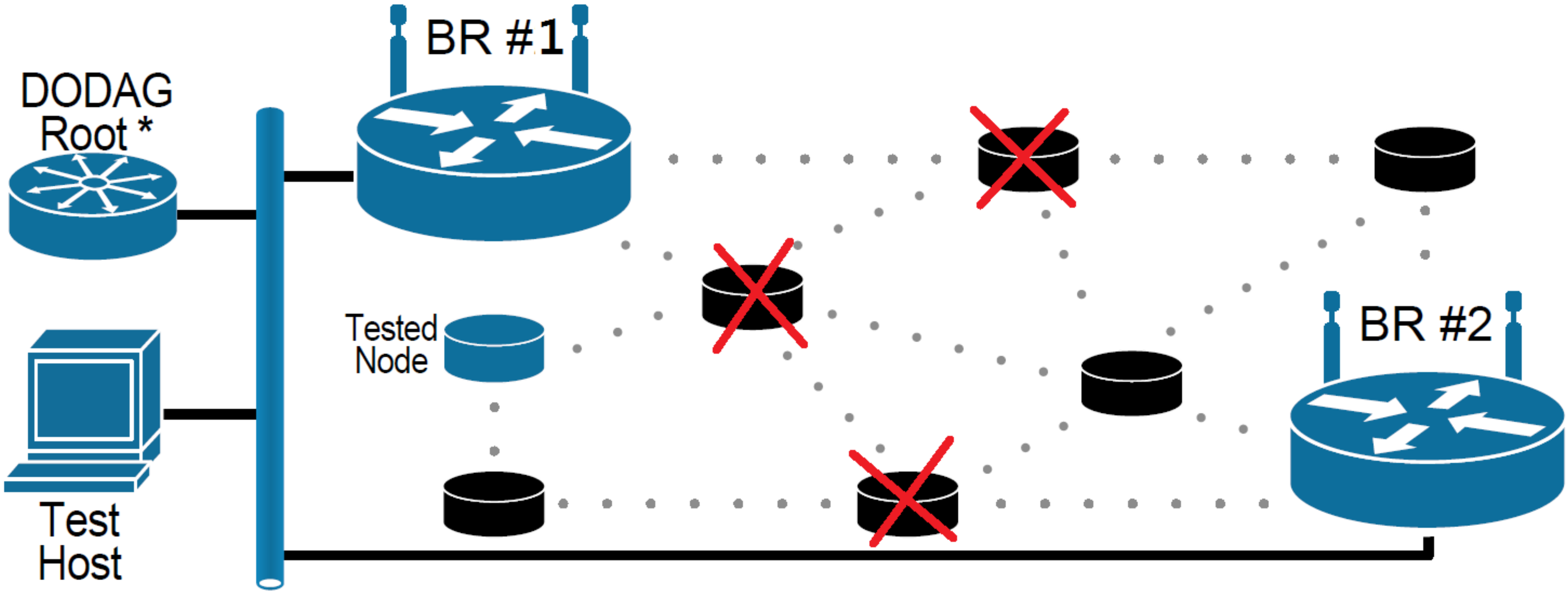
Node Mobility



Border Router Redundancy

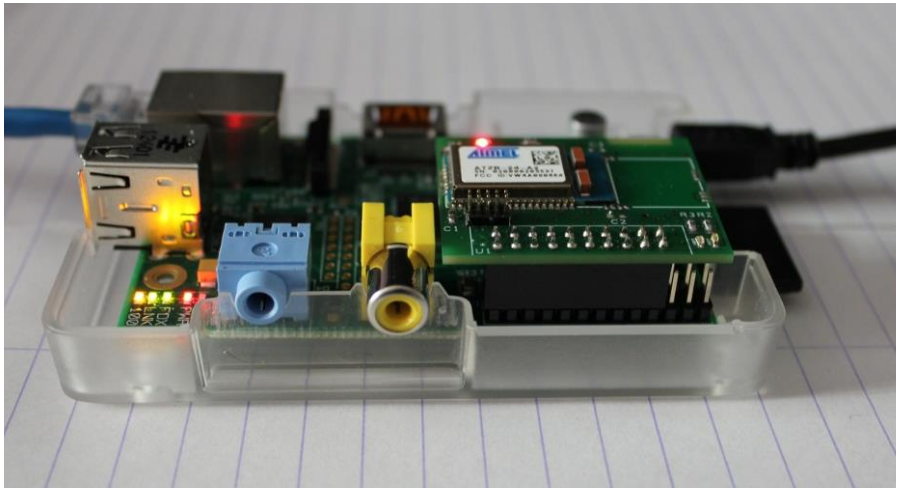


Network Partitioning



6LBR Platforms

RPi + Nooliberry

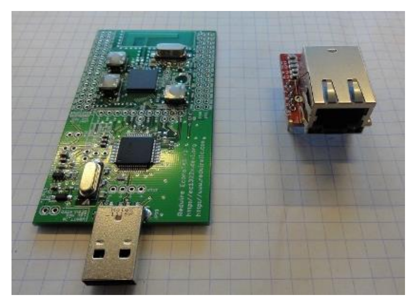


RPi + any « Contiki Mote »

BeagleBone

Redwire, LLC

Econotag



BR12

REDWIRE

IO Embedded Router (Contiki based) (PRE-ORDER $20 OFF: ships June 2013)

Submitted by admin on Mon, 2013-03-11 11:35

Description

The Redwire IO embedded router is perfect for ethernet-to-6LowPAN routing applications based on the Contiki OS.

Features

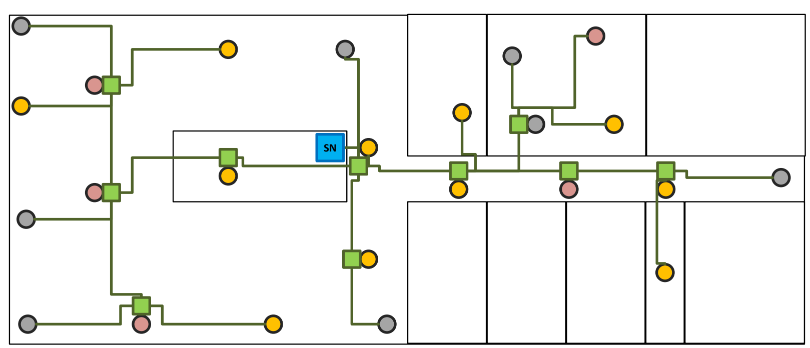
- Preloaded with Contik-based router firmware
    - Thingsquare Mist
        - NAT64/DNS64
    - 6LBR by Cetic
        - Configurable routing options
            - Smart Brigde (layer 4)
            - Router (layer 3)
            - Transparent Bridge (layer 2)
- Program and debug over USB: **no additional hardware necessary**
    - On board serial and JTAG converter
    - programming with mc1322x-load
    - debug with OpenOCD
- Powered by USB cable --- use USB power brick suitable for your country (host connection to PC is not necessary for operation)
    - US
    - UK
    - Euro Plug

Hardware Details

- Redwire M12 mc13224v module
- ENC28J60 SPI-to-ethernet controller
- FT2232 dual channel USB-to-serial and USB-to-JTAG controller
- Link and Activity LEDs
- 2 general purpose LEDs (red and green)

Ongoing Activities @ cetic

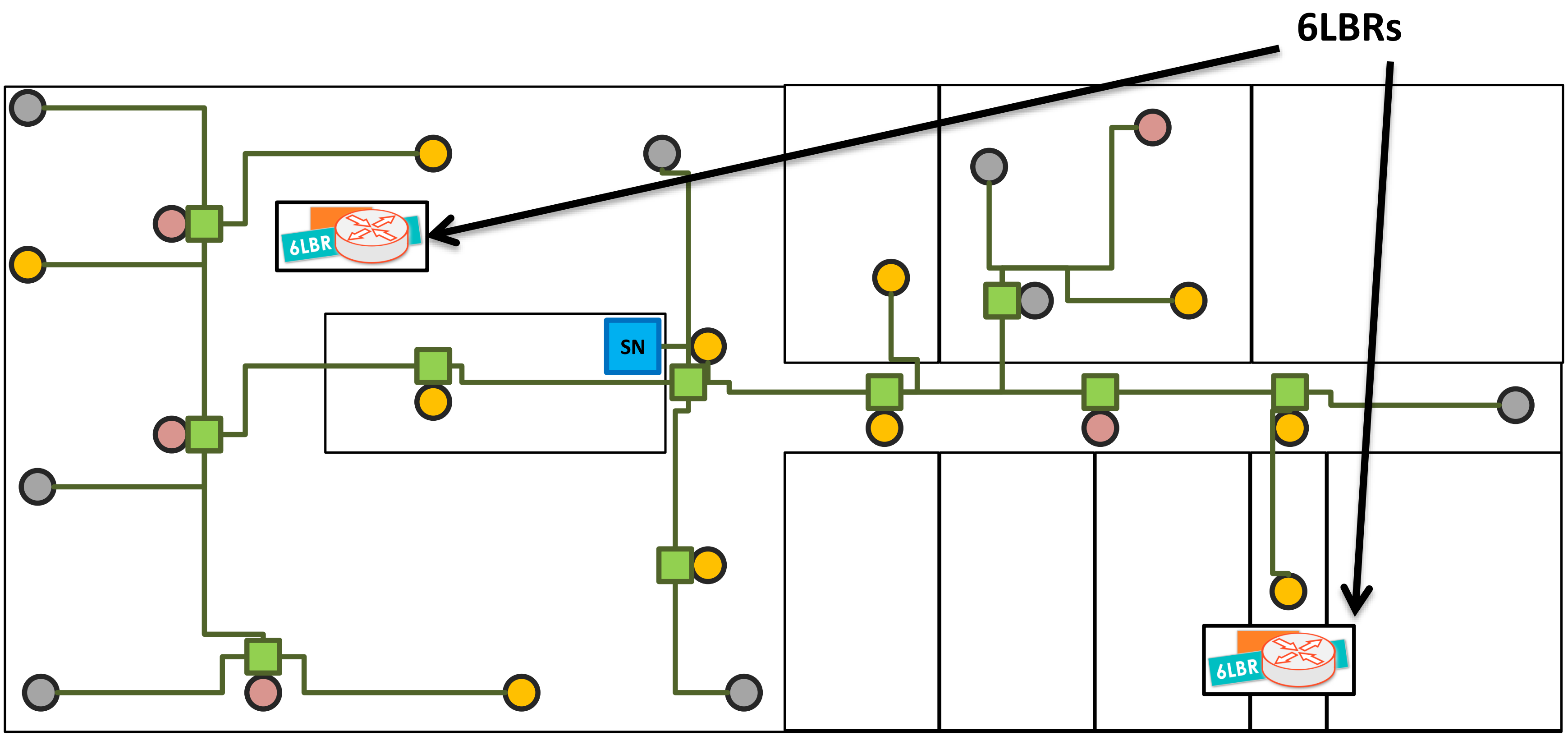
6LBR



IoT Testbed

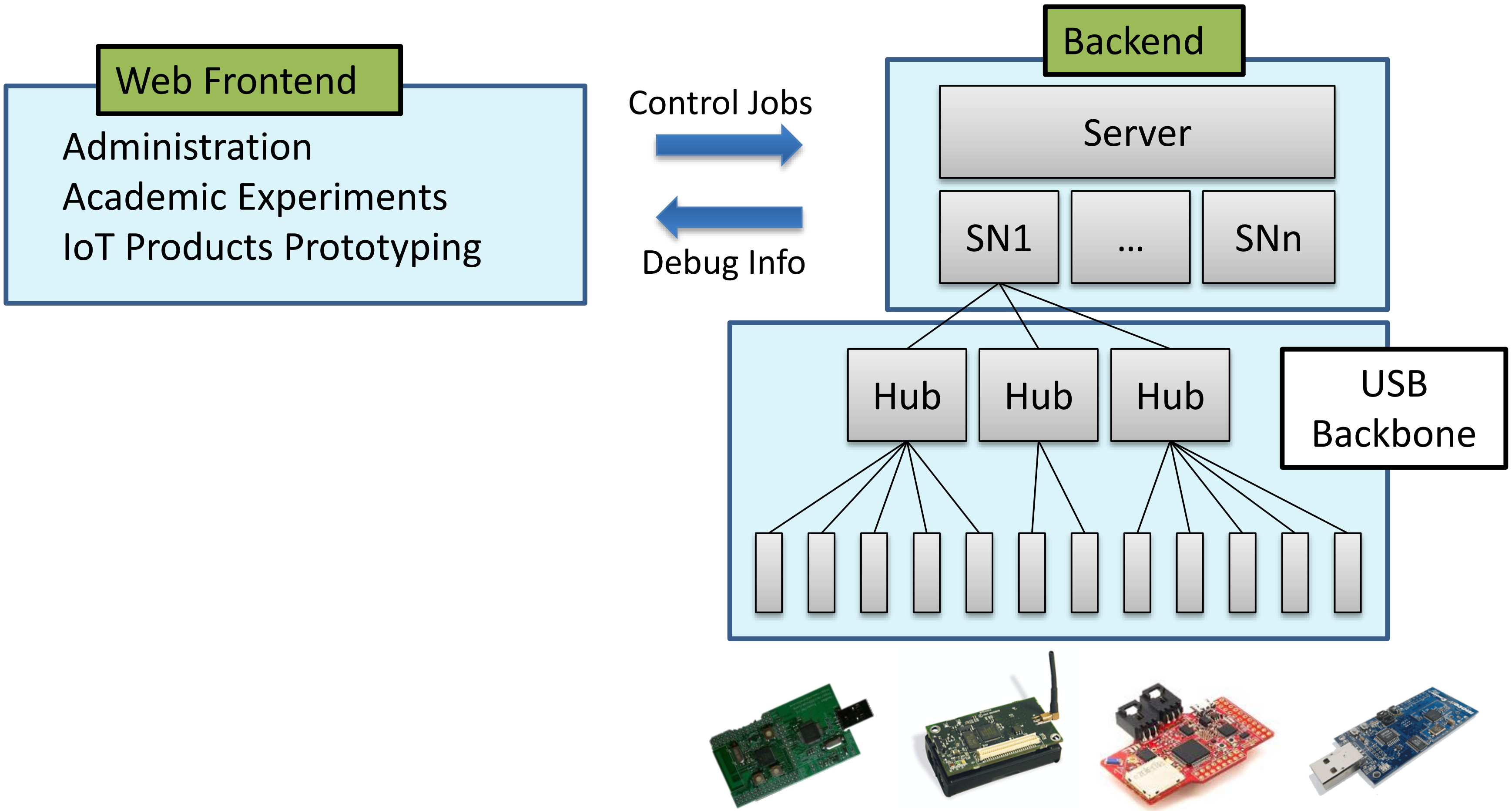
CETIC Testbed Deployment

25 nodes of 3 different models & **border routers**



CETIC Testbed Overview

CETIC WSN Testbed



From Simulations to Reality

Simulation

+ control
+ reproductibility
- poor channel models
- Limited hw platforms

Real World Deployment

- huge debugging issues
- physical access difficult
- many unexpected problems

From Simulations to Reality

Simulation

+ control
+ reproductibility
- poor channel models
- Limited hw platforms

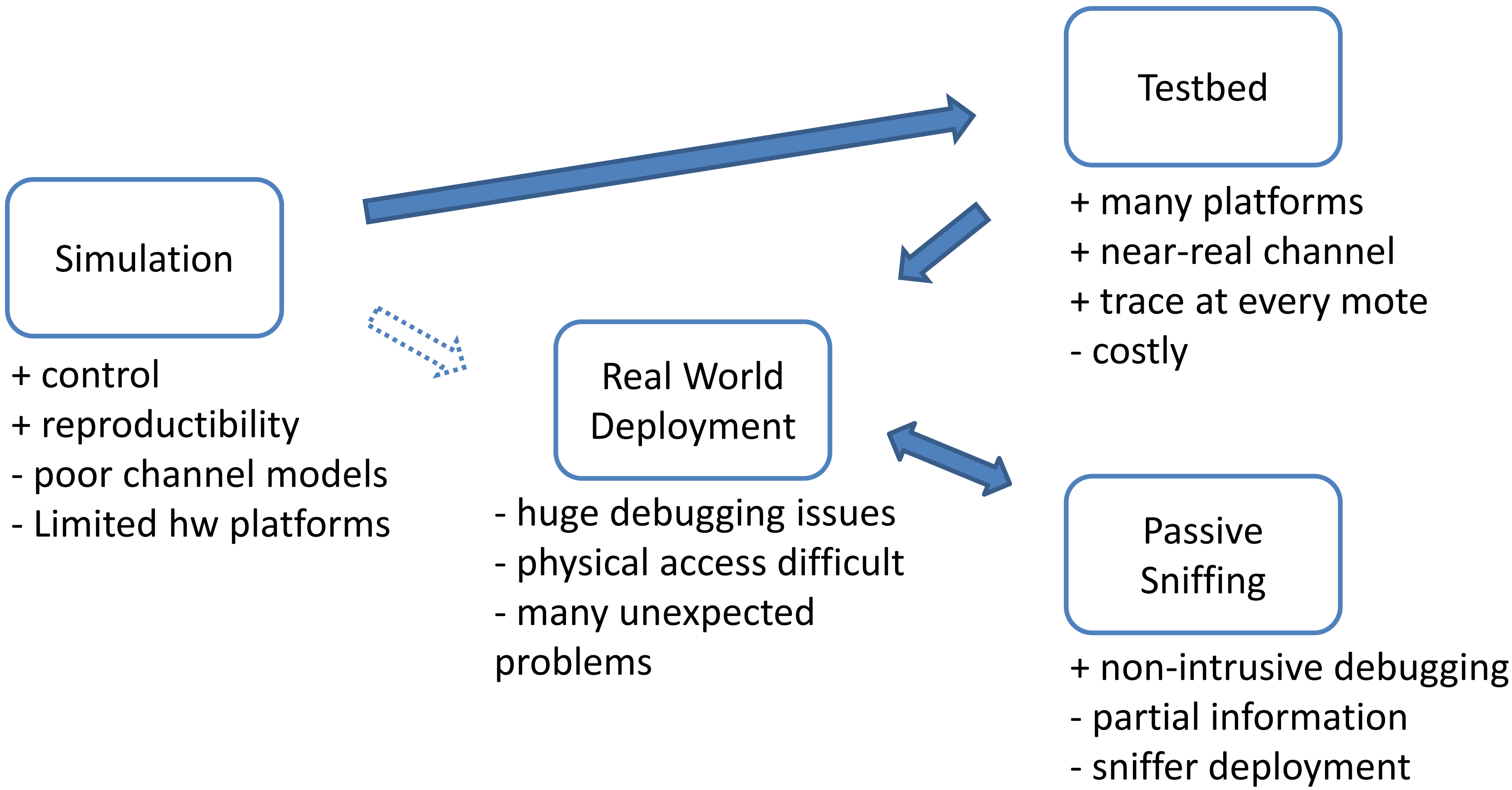
Testbed

+ many platforms
+ near-real channel
- costly
- harder to debug

Real World Deployment

- huge debugging issues
- physical access difficult
- many unexpected problems

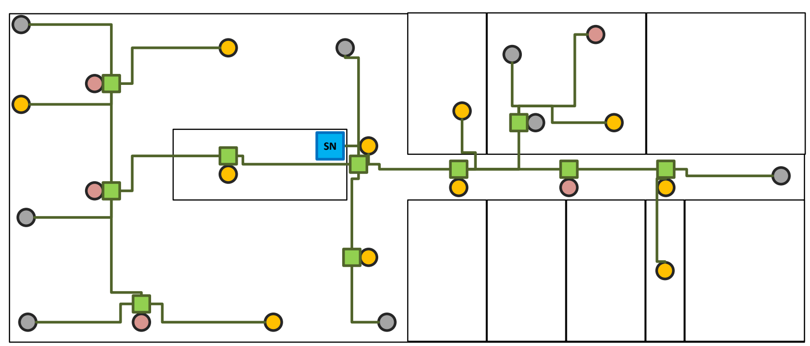
From Simulations to Reality



Ongoing Activities

@ cetic

6LBR



IoT Testbed

Foren6

Foren6: Passive IoT Diagnostics

~~Still Alive?~~
IoThanks!

CETIC

Aéropôle de Charleroi-Gosselies
Rue des Frères Wright, 29/3
6041 Gosselies
info@cetic.be

www.cetic.be

sebastien.dawans@cetic.be
laurent.deru@cetic.be