CMPSC 293S
Internet of Things (IoT)

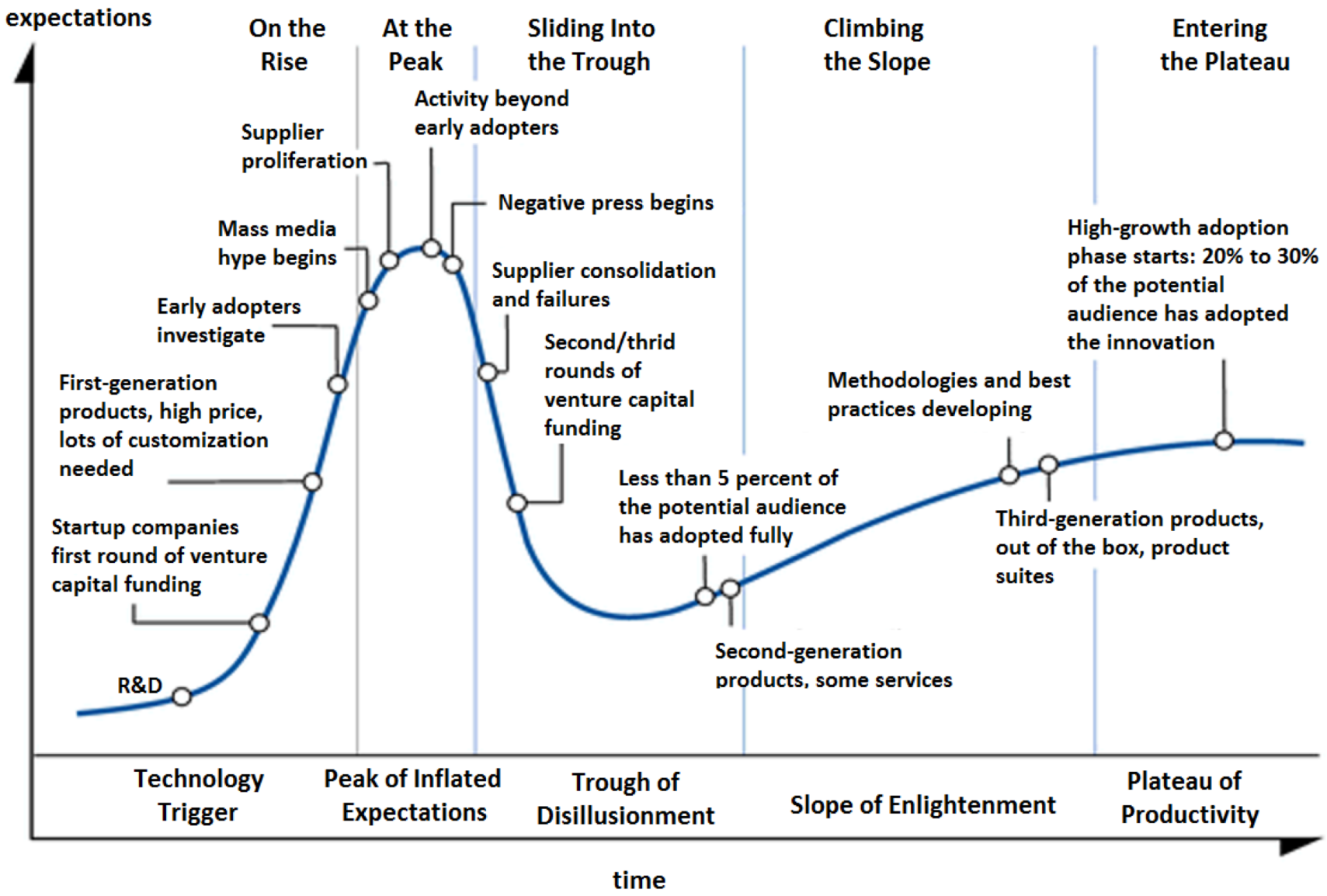
Winter Term 2019

UCSB

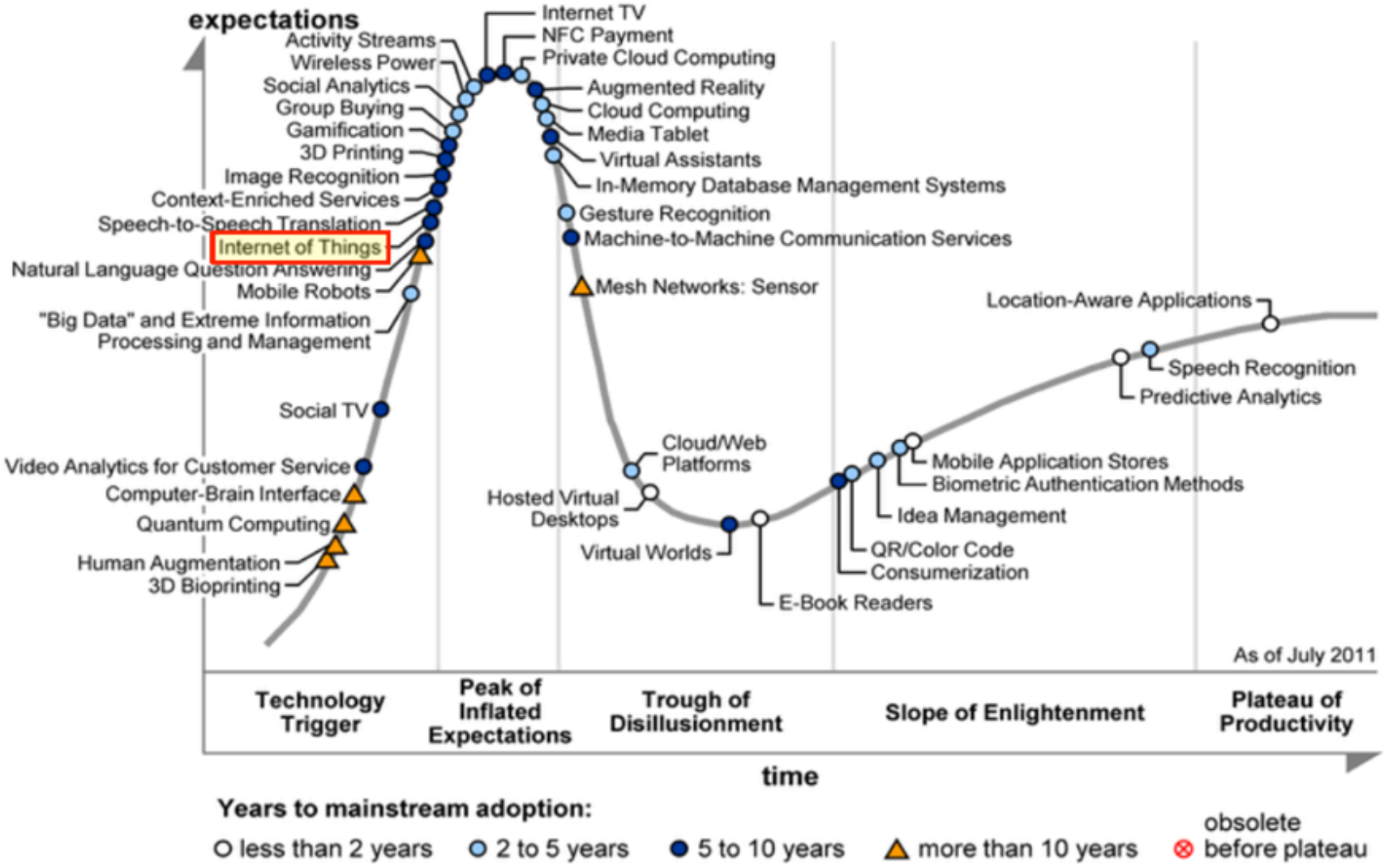
Prof. Dr. Markus U. Mock

Hype Cycle

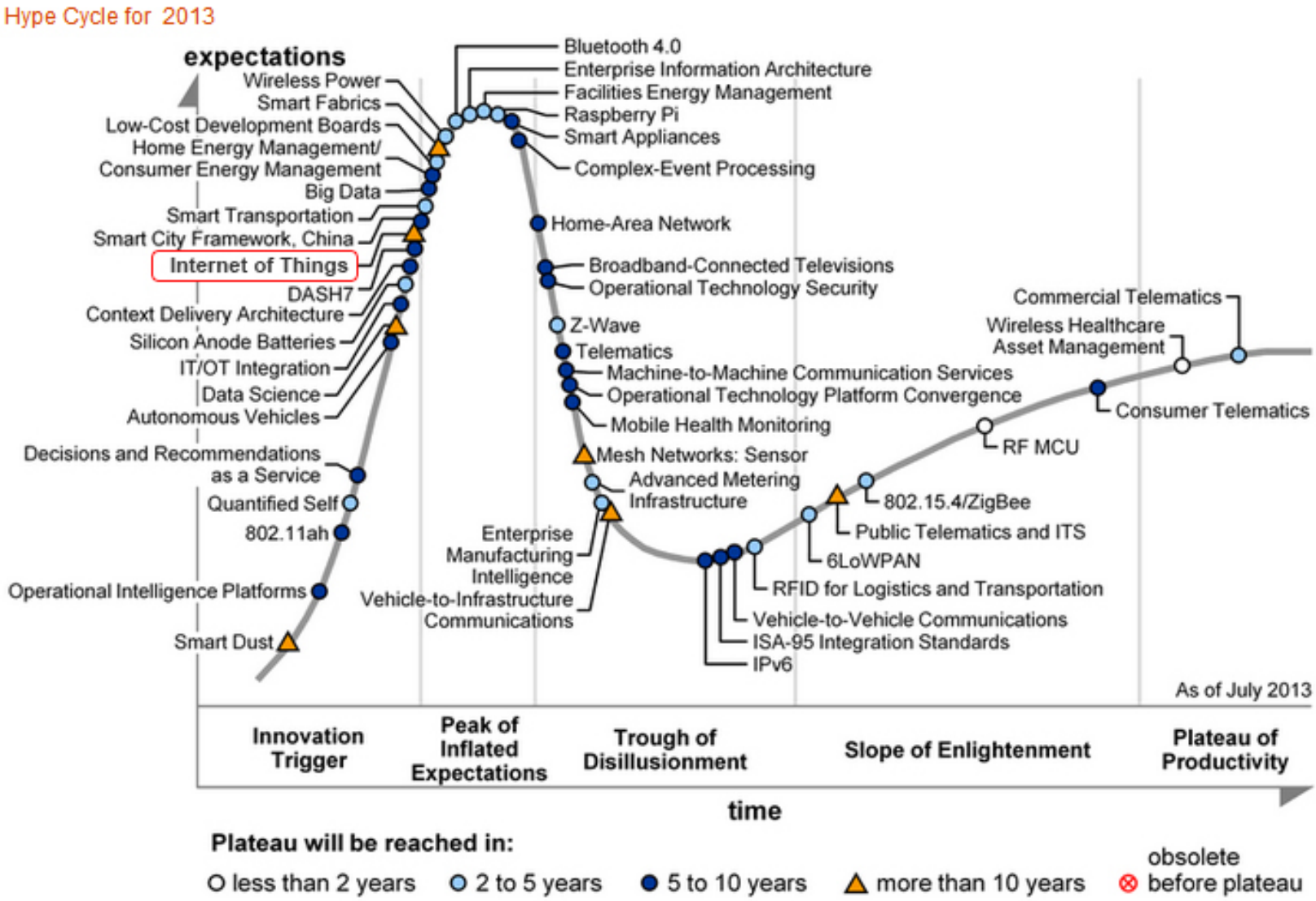
Hype Cycle is a chart that lays out where the hottest technologies are in terms of adoption. Developed by the research and advisory firm Gartner.



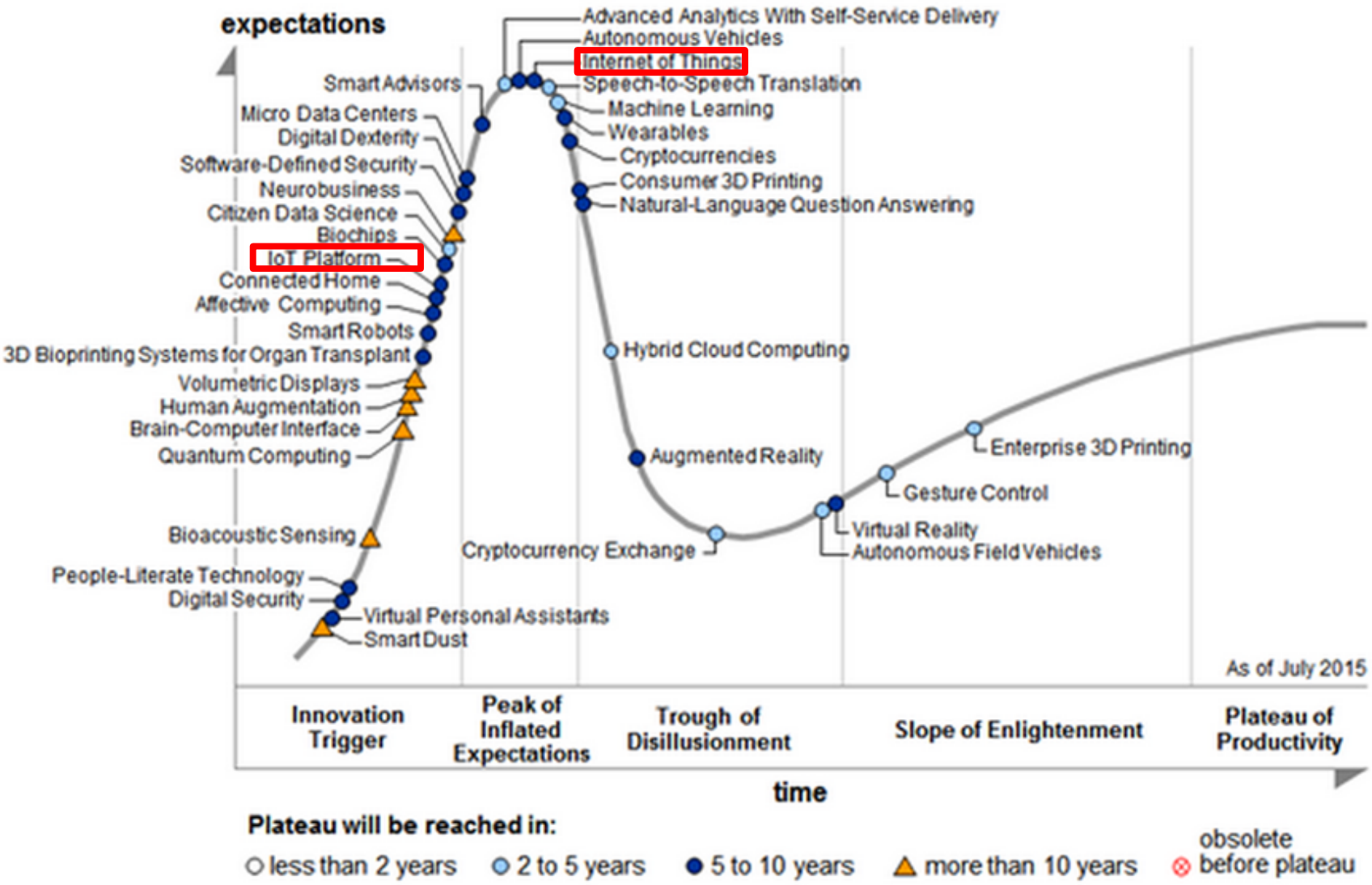
Hype Cycle 2011



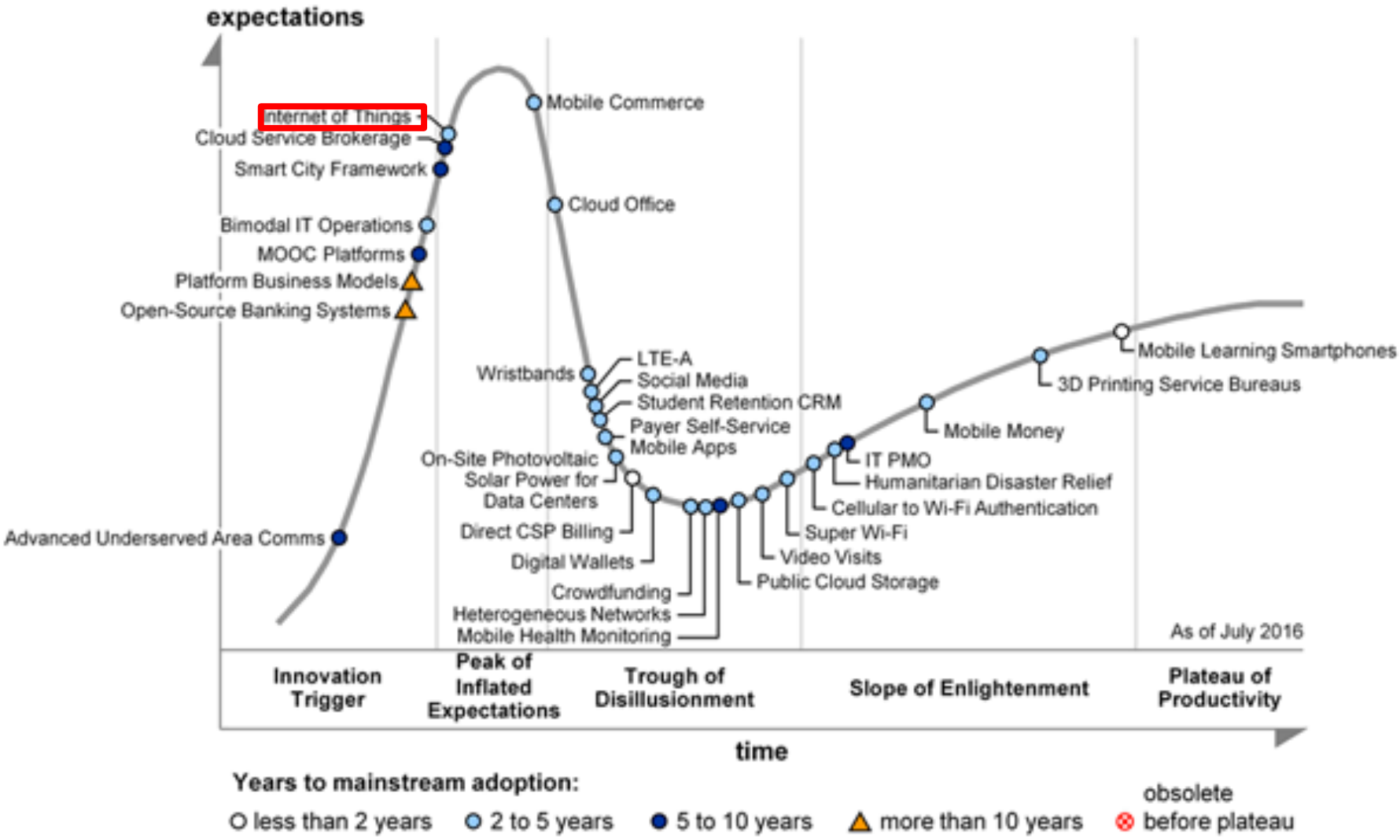
Hype Cycle 2013



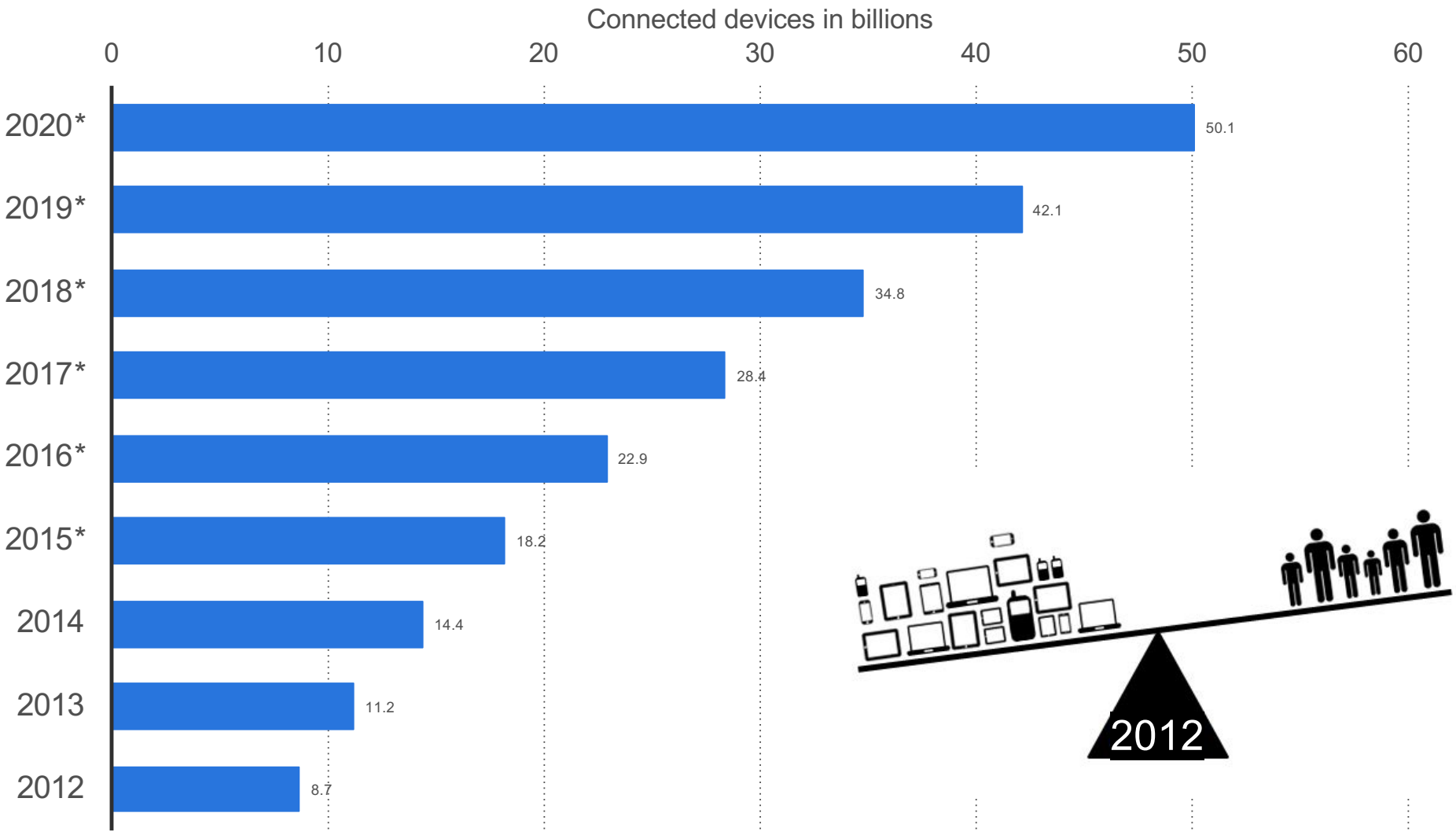
Hype Cycle 2015



Hype Cycle 2016

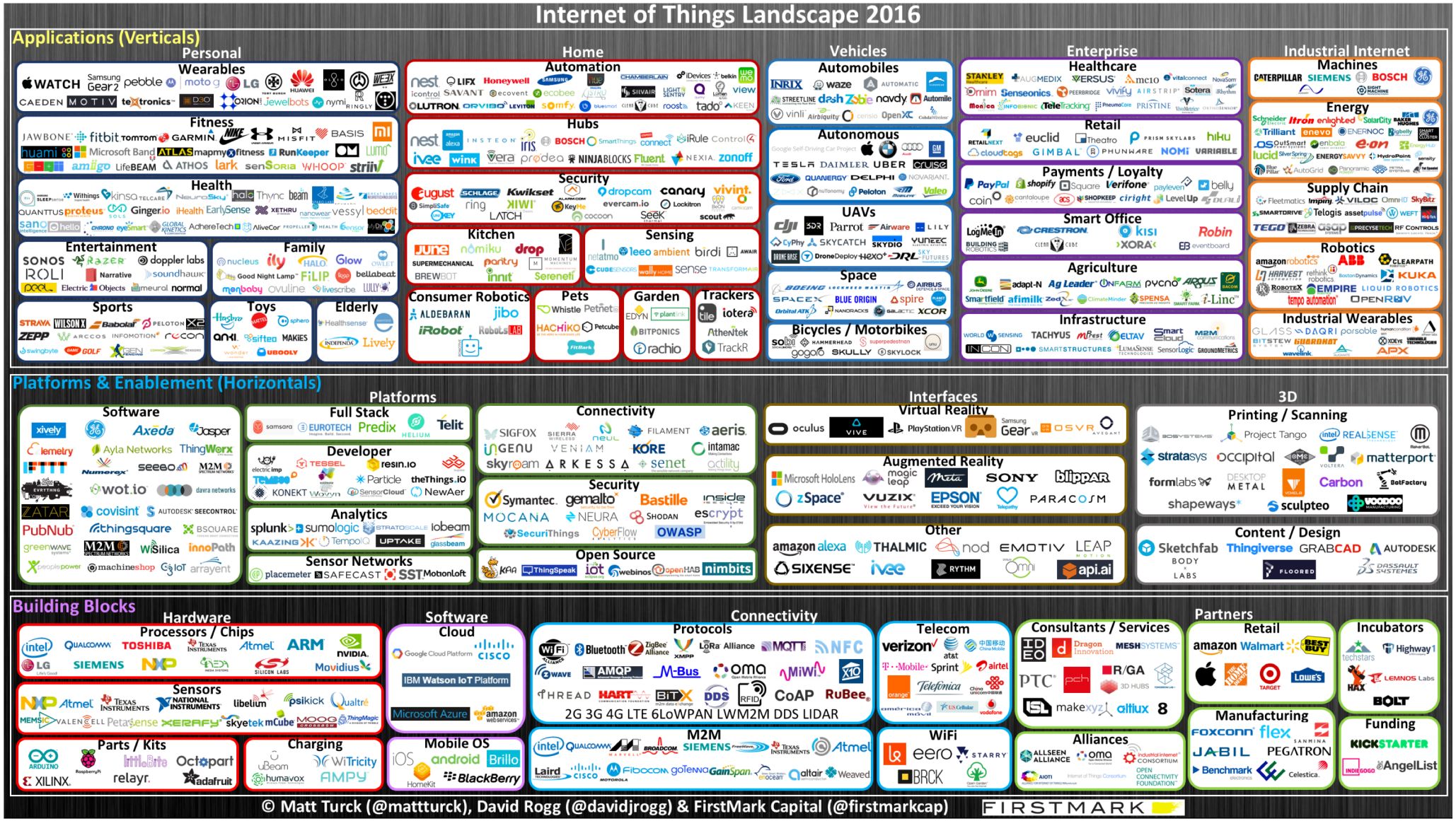


Number of connected devices worldwide



Source: Hotel News Resource; ID 471264

IoT Landscape: Huge Career Opportunities



A few words about myself

- Ph.D. in Computer Science, University of Washington, Seattle, 2002, on “Automating Selective Dynamic Compilation”, (Advisors, Susan Eggers and Craig Chambers)
- 2002 – 2005 Assistant Professor of Computer Science, University of Pittsburgh
  - Research and Teaching in Compilers, Programming Languages, Computer Architecture
- 2005-2010: Google, Mountain View, Advertising and Google Docs Backend
- 2010-2014: VMware, Consulting, Nutanix, Amazon Kindle Division
  - Worked on the Kindle Fire and Amazon Echo

Organizational Issues (1)

- **Lecture**
  - Mondays & Wednesdays, 3pm - 4:50pm, Phelps 2510
- **Documents**
  - Syllabus online at http://cs.ucsb.edu/~mock/cs293S/index.html
  - Using Gouchospace for lecture notes etc.
  - Piazza sign up at: piazza.com/ucsb/winter2019/cmpsc283S
- **Exams**
  - Midterm planned for 2/4, no final
- **Lecturer**
  - Prof. Dr. Markus U. Mock
  - Office: HFH 5112
  - Contact: Via Piazza for questions etc.
  - Office hours: Wednesdays from 13:30 – 14:30 (subject to change) and after class

Organizational Issues (2)

- **Books & Articles**
  - There is no textbook for the class
  - Articles will be provided in Gouchospace as needed
- **Project**
  - Group Project 2-3 people in one group, form a group this week
  - Focus is Data Analysis, statistical and / or machine learning techniques
  - You will work with sensor data and do analysis for them
    - More details Wednesday

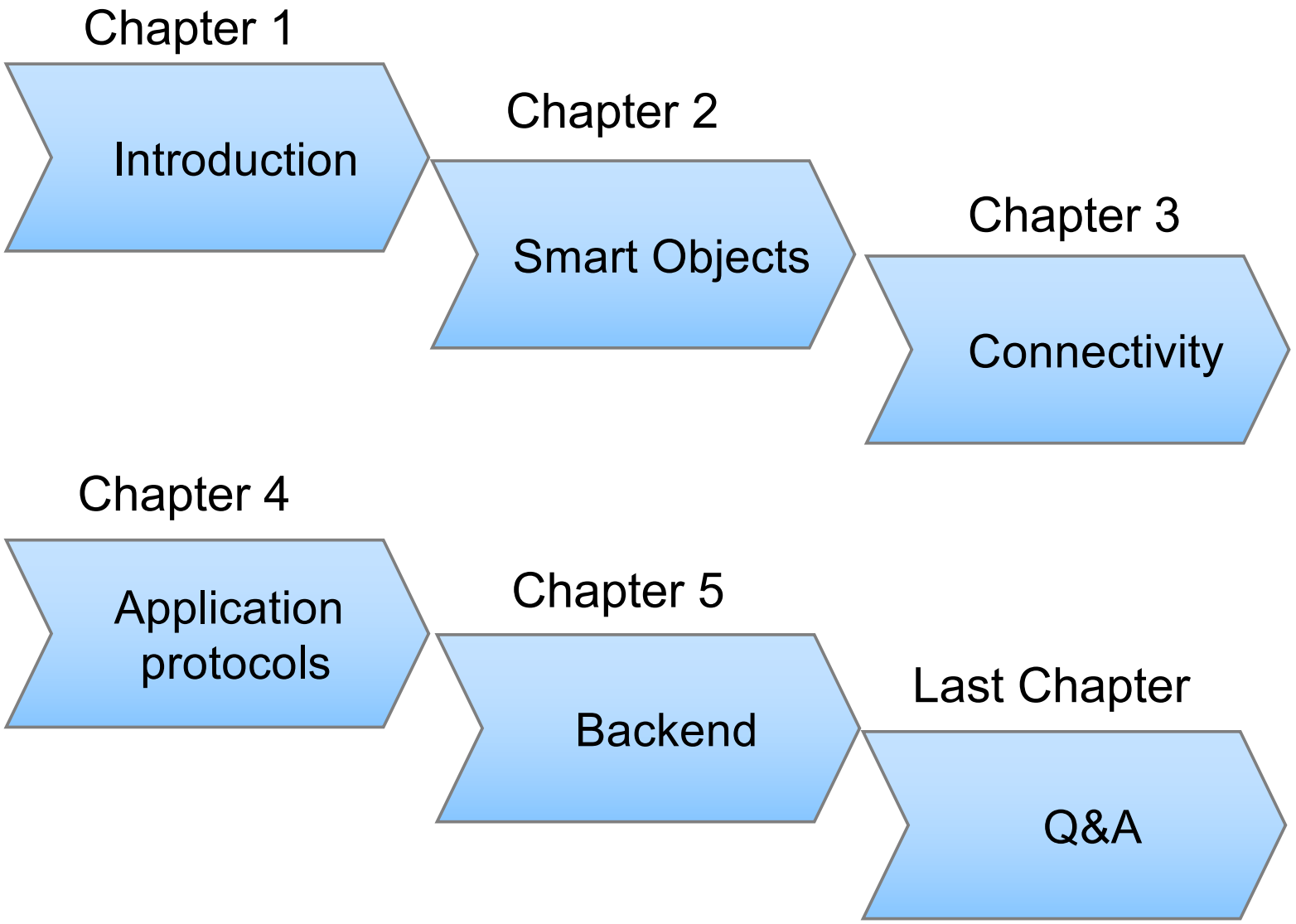
SYLLABUS

- Available on http://cs.ucsb.edu/~mock/cs283S/index.html

Will probably be updates as we move along

Content

Planning



Overview – Lecture Topics

- Introduction
- Smart Objects
- Raspberry-PI and Arduino Platforms
- Connectivity for IoT
- IoT App Protocols: MQTT & CoAP
- IoT Cloud Backends
- IoT Device Management: OMA LWM2M
- Application Development: NodeRED
- 11: IoT Misc.: Cloud / Energy-Efficiency / Security / OS
- Anomaly Detection methods
- Not necessarily all of these topics are covered

Group Exercise: Interviews

- Counting exercise
- Find an interviewee (listen for instructions)
  - Get to know your interviewee (1 minute)
    - Why are they taking the class?
    - What experience do they have in IoT?
    - Unique factoid: what do they think is something unique that no one else in the class has? (e.g.,10 siblings, married 15 times, born on an airplane etc.)
  - Take turns
    - Now the interviewer becomes the interviewee (1 minute)
- If you have a bad short-term memory: take notes
  - You will introduce your interviewee to the class (1 minute)

Introduction to IoT

Contents

- Illustration of Smart Objects
- IoT Definition
- IoT Applications/Verticals
- IoT Technology Roadmap
- IoT Market

What IoT Devices Do You Know?

Smart Things at Consumer Electronics Show (CES 2013)

- Parrot: “Flower Power” with humidity & light sensors [to monitor health of the plant]

- Withings: “Smart Body Analyzer”
  [to monitor weight, heart rate, temperature and air quality]

- Belkin: “Smart WeMo Light Switch”
  [to remote or automatic control light]

- Dacor: “Android Smart Oven”
  [to install apps and download recipes]

- .. And many others

Smart Things at CES'2015

- Sony Smartwatch
- Alcatel Smartwatch
- Withings Smartwatch with the HealthMate app

- Connected Pacifier

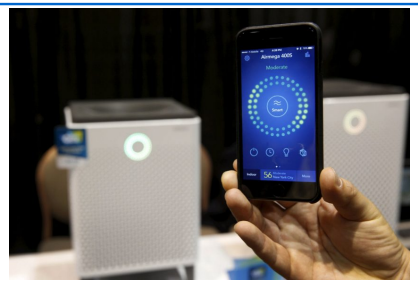
- Mini Drones

- .. And many others

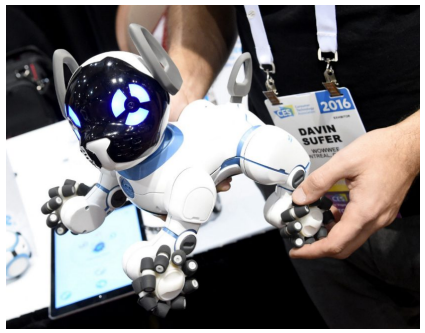
https://www.pastemagazine.com/blogs/lists/2015/01/the-10-best-gadgets-from-ces-2015.html

Smart Things at CES'2016

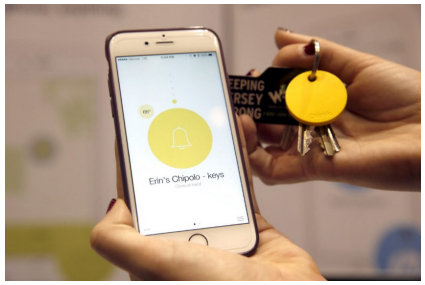
- The WiFi-enabled, Airmega smart air purifier



- WowWee's CHiP robot dog



- The Bluetooth-enabled Chipolo is a wireless tracker



- Digisole smart shoes are controlled by a smartphone app (runprofiler, heated, etc)

- .. And many others

https://www.thestar.com/business/2016/01/05/10-of-the-best-gadgets-at-ces-2016.html

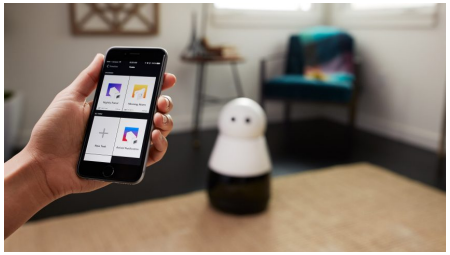
Smart Things at CES'2017

- Plume is a wearable device that tracks pollution around you

- The Griffin Connected Toaster

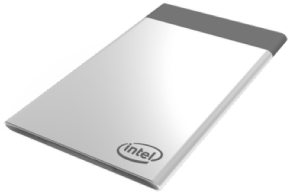
- Checking your blood alcohol content with a breathalyzer

- Kuri is an adorable little robot designed for the home



- Motiv's fitness/sleep tracking ring

- Intel's Compute Card, which is a mini-computer about the size of a credit card.



- .. And many others

https://techcrunch.com/2017/01/09/10-of-the-coolest-gadgets-we-saw-at-ces-2017/
http://time.com/4626654/ces-2017-best-gadgets/

Smart Cars

"The vehicle is actually the third-fastest growing connected device behind smart phones and tablets"
IHS Automotive

- FORD: EV charging app
- NISSAN: Kan-Kan-Kyo house

https://secure.myvolt.com/

Src: http://social.ford.com

- TOYOTA: Smart Center
- CHEVROLET Volt: OnStar app

Src: http://www.toyota-global.com/innovation/smart_grid/

https://secure.myvolt.com/

- DAIMLER BENZ: Smart Car2Go
- BMW-TENDRIL: BMW ActiveE

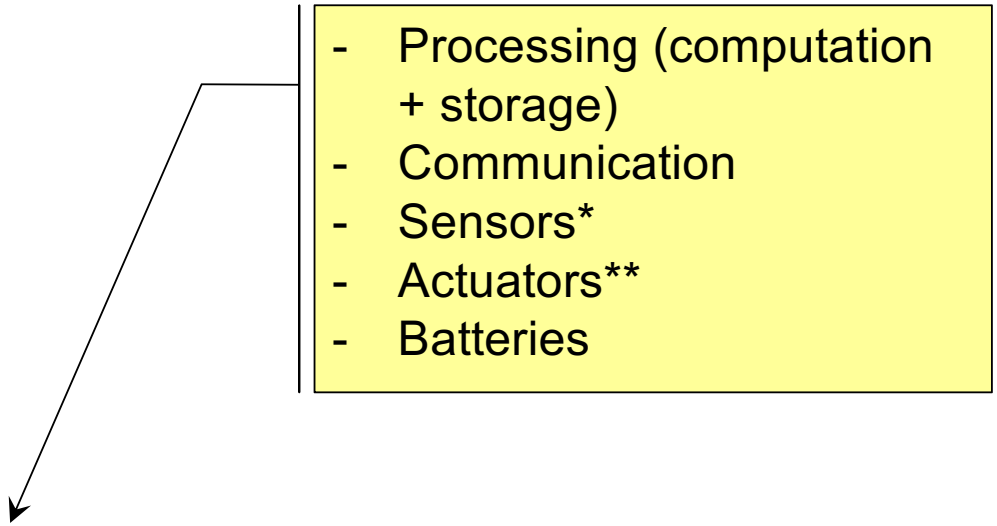
- ..

- Autonomous cars

Google

What is Common to All / Most IoT Devices?

Prof. Elgar Fleisch:



Thing + IT = Function +

Service

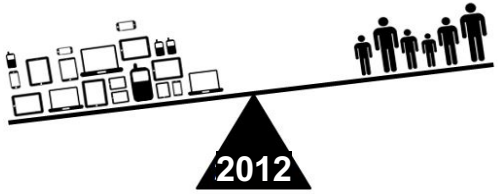
***Sensors** are active devices that **measure** some variable of the natural or man-made environment (e.g., a building, an assembly line, an industrial assemblage supporting a process).

An **actuator is a **mechanized device** of various sizes (from ultra-small to very large) that accomplishes a specified **physical action**, for example, controlling a mechanism or system, opening or closing a valve, starting some kind or rotary or linear motion, or initiating physical locomotion. An actuator is the mechanism by which an entity acts upon an environment.

Metcalfe's Law (1980)

- Robert M. Metcalfe is the inventor of Ethernet
- The value of a network grows **quadratically—proportionately** with the number of connections you can make.
- Metcalfe's Law: **value = $n^2 - n$**

# participants (n)	Value
3	6
10	90
100	9.900
$7x10^9$ (Internet of People)	$\sim 7^2x10^{18}$
$M*7x10^9$ (Internet of Things)	$\sim M^2*7^2x10^{18}$



In 2020: $50x10^9$ connected devices

"Everything that can be networked, should be networked!"

- But scaling is challenging!

Further important laws

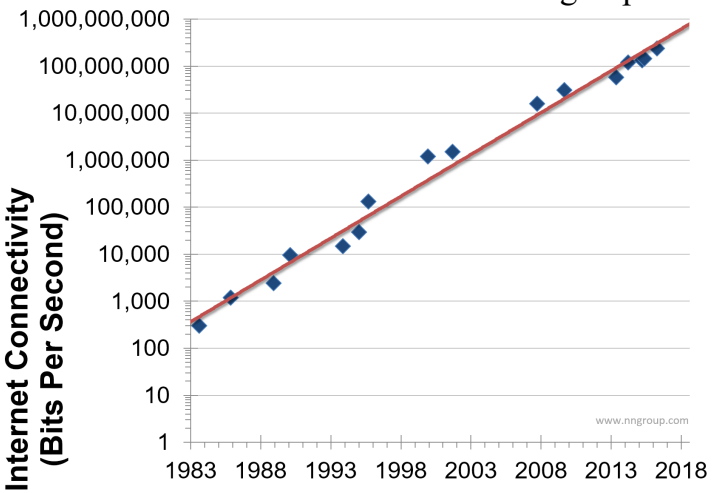
- Nielsen‘s law(1998): The bandwidth doubles every 24 months (updated 2018: 50% per year)

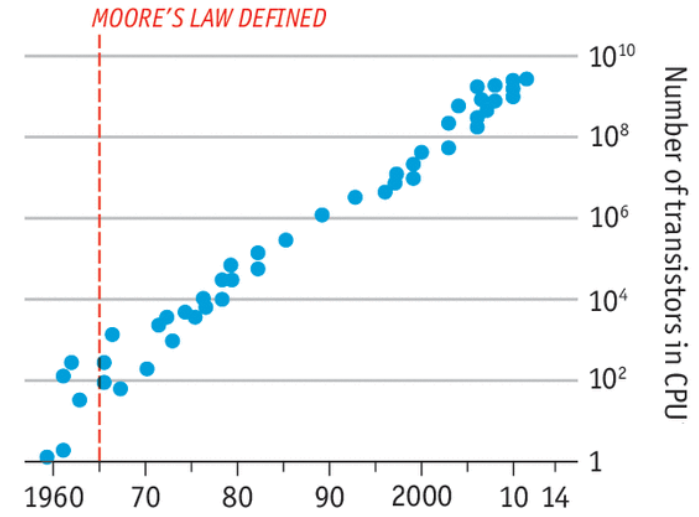
→ „Everything that can be networked is also networked!“

- Moore‘s law (1965): the integration density (or the computing power of computer chips) doubles approximately every 18 months .

→ „Everything that can be digitized is also digitized“ Karl-Heinz Land

Src: nngroup





Src: Intel

IoT Definition(s)

- "The **Internet of things** (**IoT**) is the inter-networking of physical devices, vehicles (also referred to as "connected devices" and "smart devices"), buildings, and other items—embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data" Wikipedia
- "The **Internet of Things (IoT)** is the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment" Gartner
- "The Internet of things links the objects of the real world with the virtual world, thus enabling anytime, anyplace connectivity for anything and not only for anyone. It refers to a world where physical objects and beings, as well as virtual data and environments, all interact with each other in the same space and time" Cluster of European Research Projects on the Internet of Things, "Vision and Challenges for Realizing the Internet of Things", March 2010

IoT Definition(s)

- "The IoT refers to as ubiquitous networking or pervasive computing environments, is a vision where all manufactured things can be network enabled, that is connected to each other via wireless or wired communication networks" European Network and Information Security Agency (ENISA)
- "The IoT is a world where physical objects are seamlessly integrated into the information network, and where the physical objects can become active participants in business processes. Services are available to interact with these "smart objects" over the Internet, query and change their state and any information associated with them, taking into account security and privacy issues. RFID, sensor networks, and so on are just enabling technologies " SAS
- "The **Internet of things** (**IoT**) is the infrastructure of the information society" Global Standards Initiative on Internet of Things: IoT-GSI, 2013

IoT: Espousing Cyber & Physical Worlds

Physical World - Continuous Digitalization:

- Pervasive sensing/embedded computing
- Low-cost wireless connectivity for things (RFID, ZigBee, NFC, dongles ..)

+

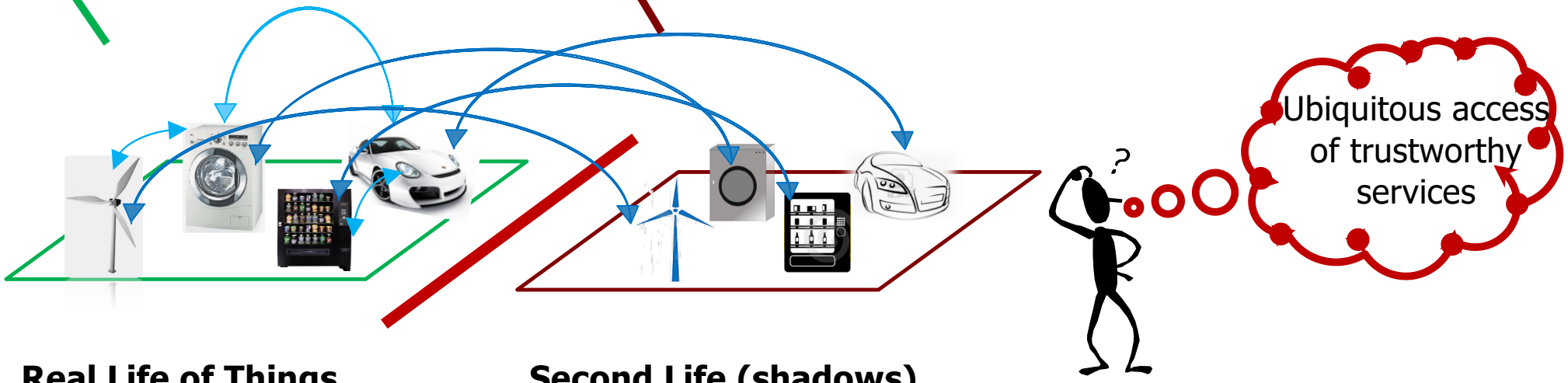
Cyber World - Web of Things:

- Powerful computation, sensing cloud (quality-aware, intelligent, predictive Big Data analytics)
- (Semantic) Web 3.0
- Information-centric networks

=

IoT - Cyber Physical World:

- Fully automated processes based on rules/predicates
- Social networking for things
- (Virtual) Things, augmented reality

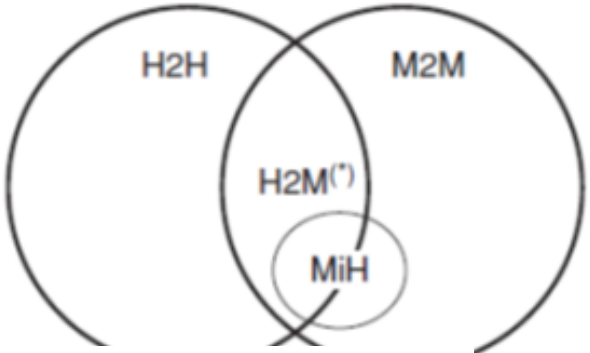


Real Life of Things (Physical World)

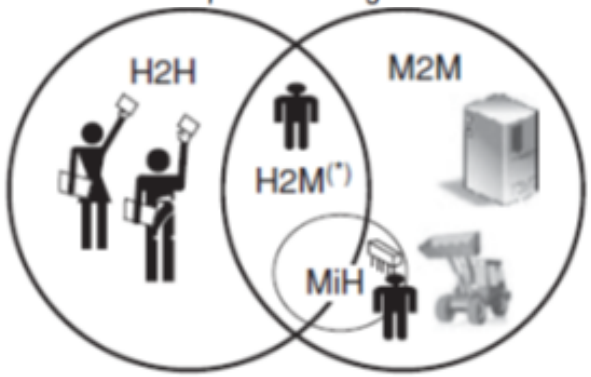
Second Life (shadows) of Things (Cyber World)

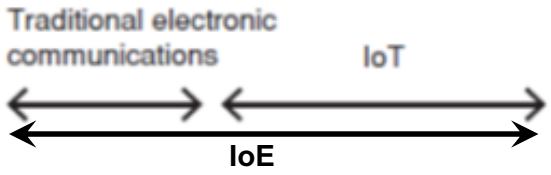
What is IoT, M2M, IoE?

Interaction space partitioning showing humans and machines



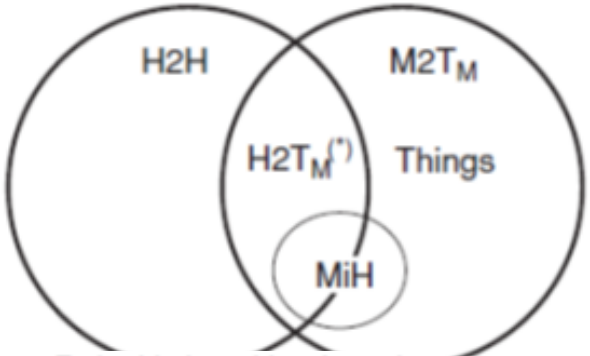
Interaction space showing icons



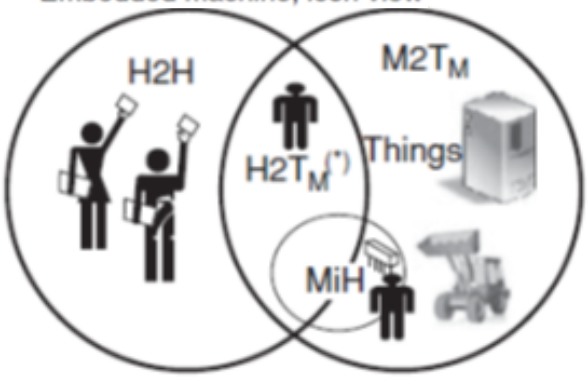


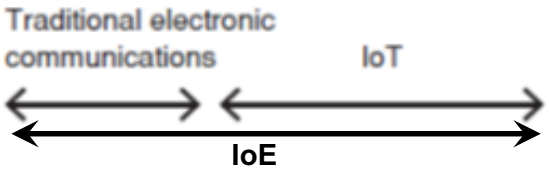
H2H: Human to Human
H2M: Human to Machine = $H2T_M$: Human to Thing with Microprocessor/Machine
IoE: Internet of Everything

The target machine is shown explicitly to be embedded in the "thing"



Embedded machine, icon view





M2M: Machine to Machine = $M2T_M$: Machine to Thing with Microprocessor/Machine
MiH: Machine in Humans
(e.g., medical sensors)
(also includes chips in animals/pets)

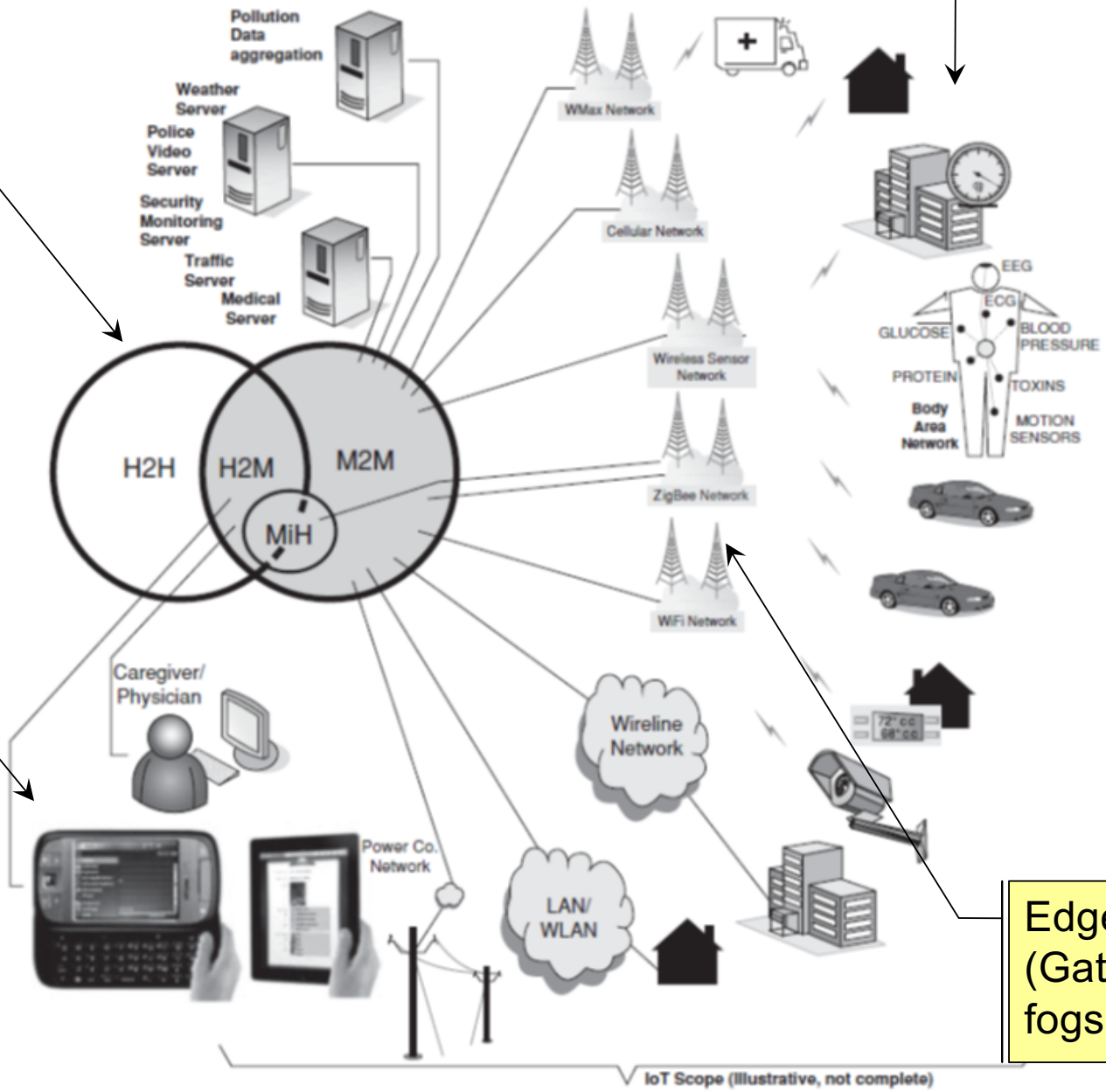
SRC: Minoli „BUILDING THE INTERNET OF THINGS WITH IPv6 AND MIPv6", 2013

Scope of IoT by way of Illustration

Smart devices

Backend (server, cloud,…)

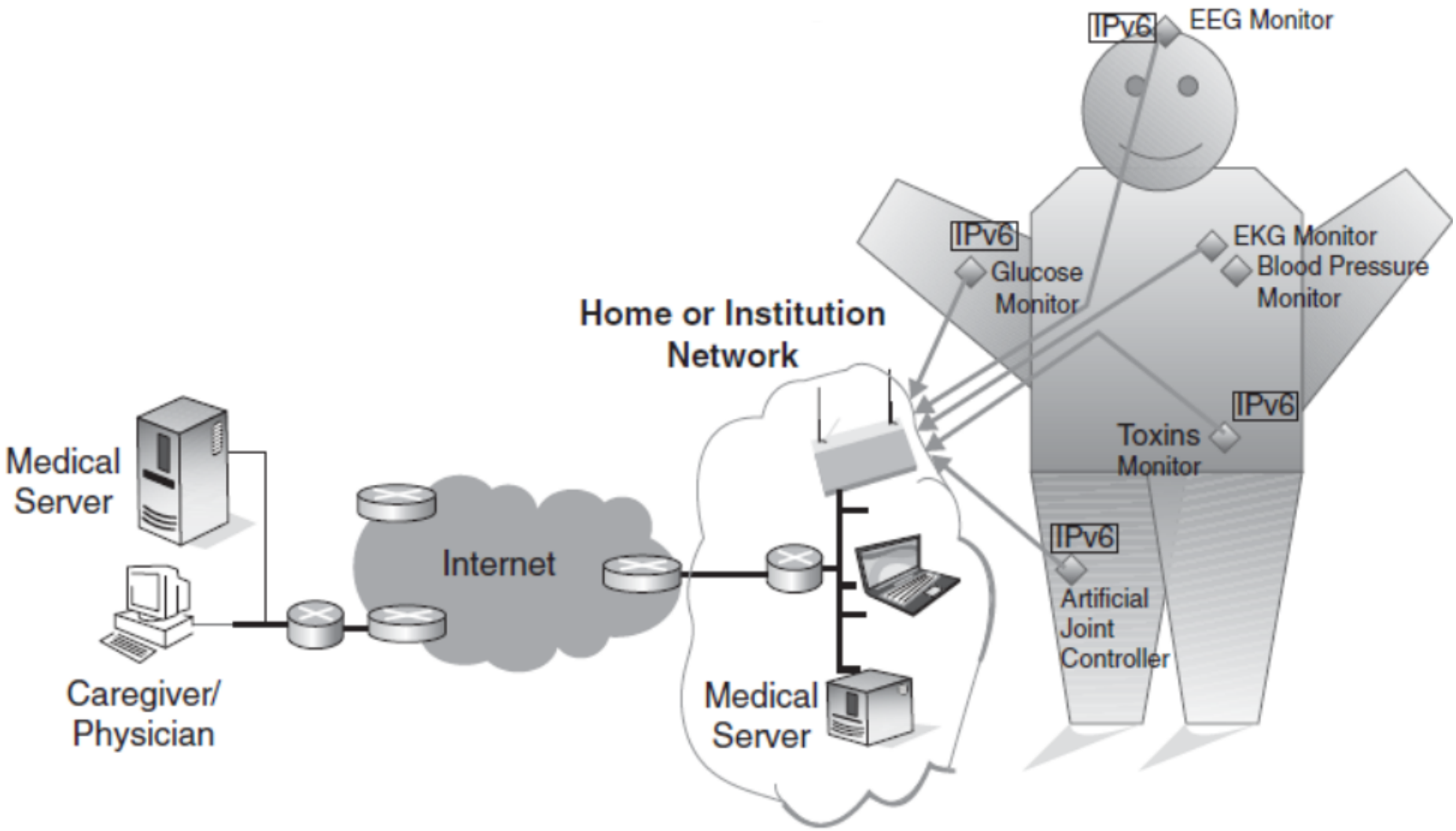
Endusers



Edge devices (Gateways, fogs)

SRC: Minoli „BUILDING THE INTERNET OF THINGS WITH IPv6 AND MIPv6“, 2013

Yet another illustrative example of the IoT (Body Area Network – BAN)



SRC: Minoli „BUILDING THE INTERNET OF THINGS WITH IPv6 AND MIPv6“, 2013

Application Domains (or Verticals)

Automotive

Smart Cities

Asset Mgmt

Industrial manufacturing

Fleet Mgmt & Logistics

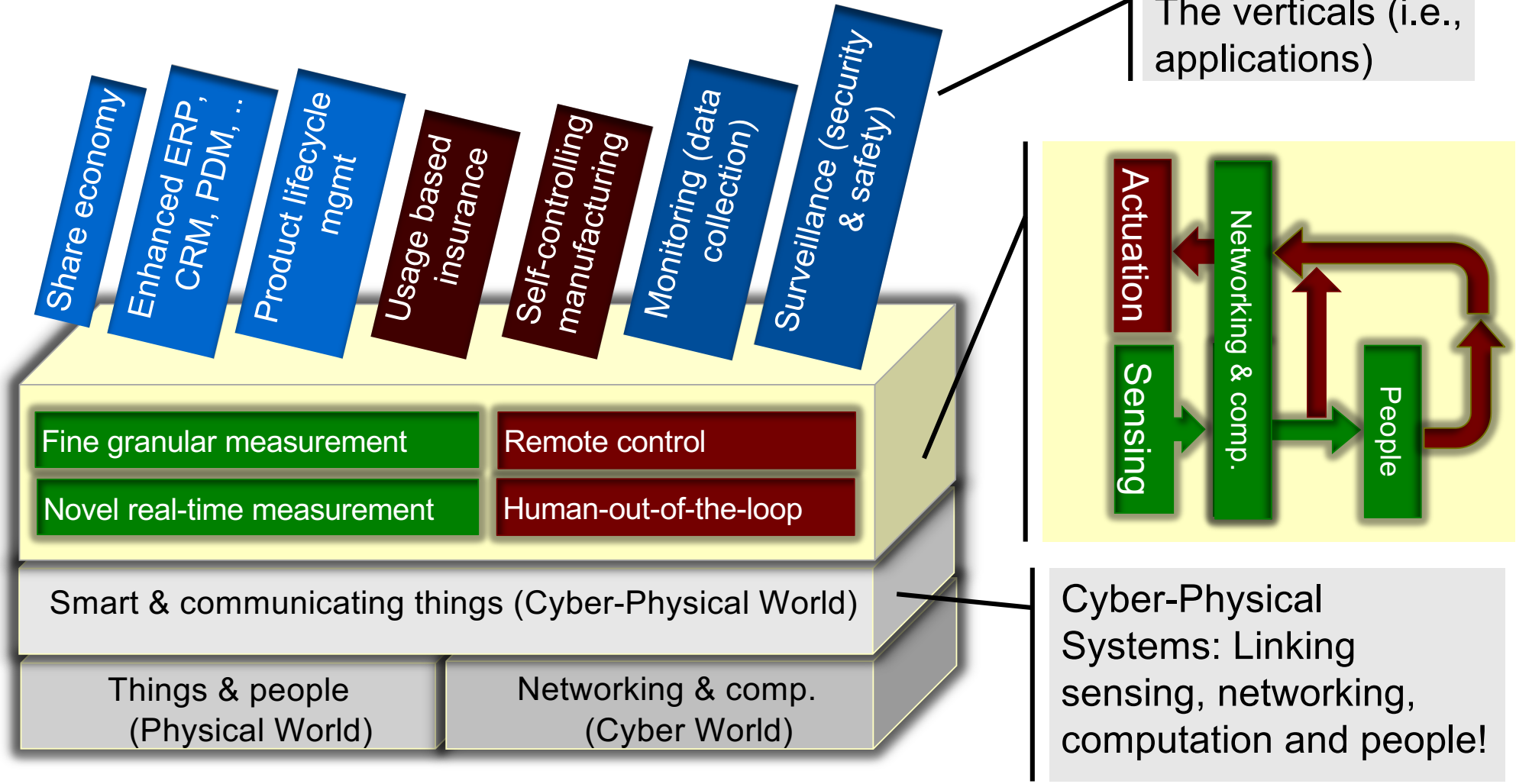
Wearables

Smart meters

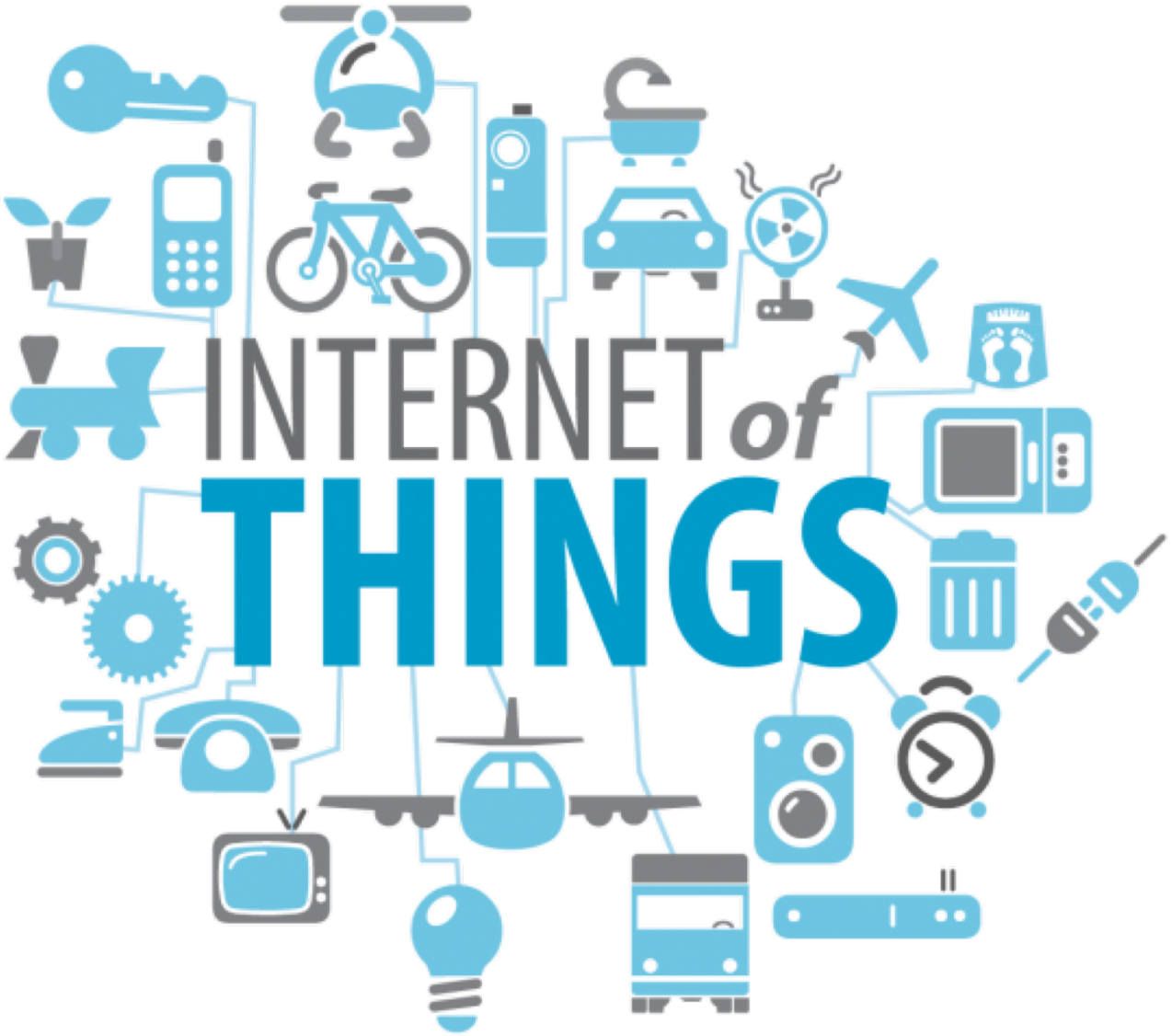
Safety & security

Agriculture & environment

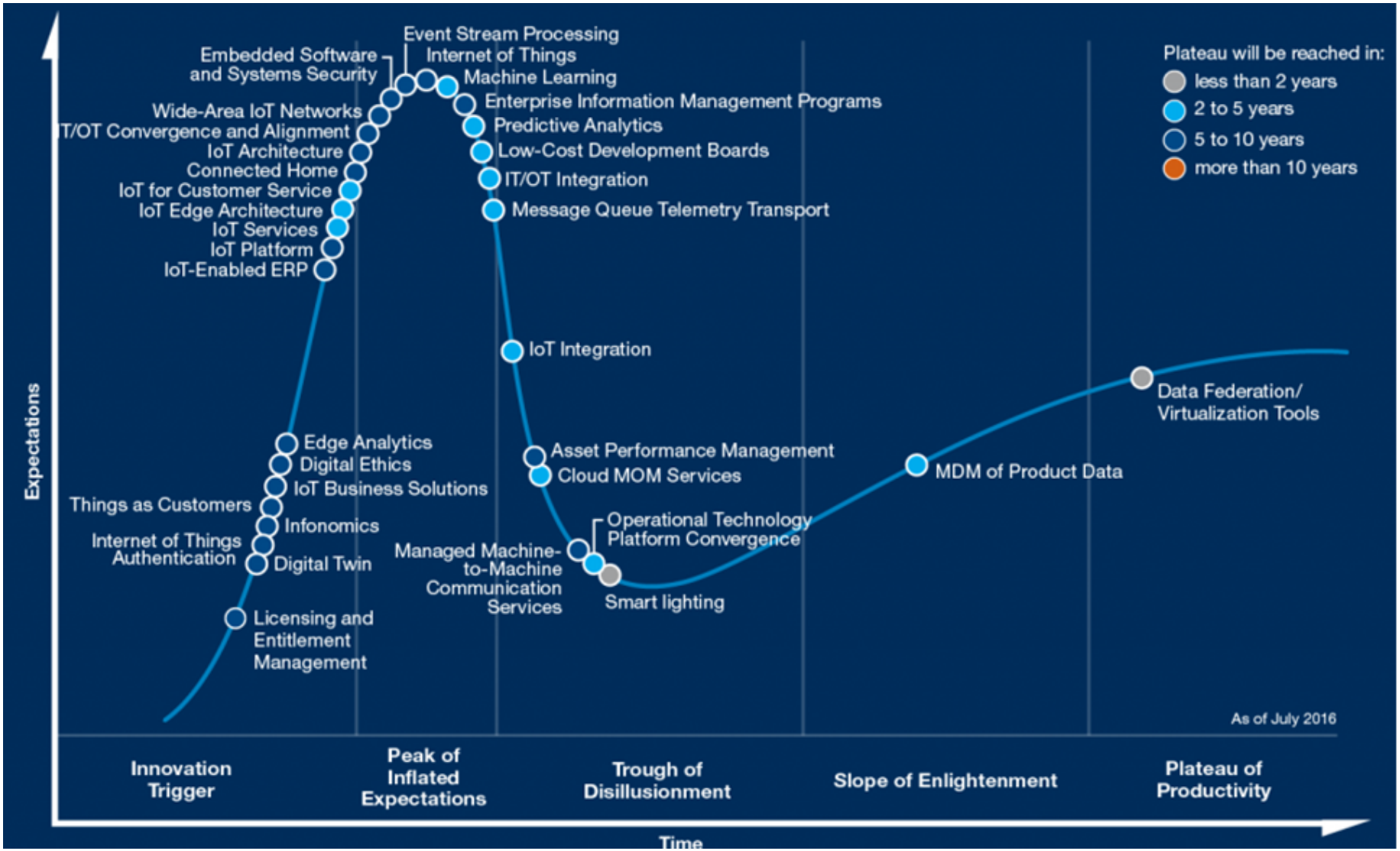
IoT Applications



What isn’t IoT?



Hype Cycle for the IoT, 2016



IoT Standardization

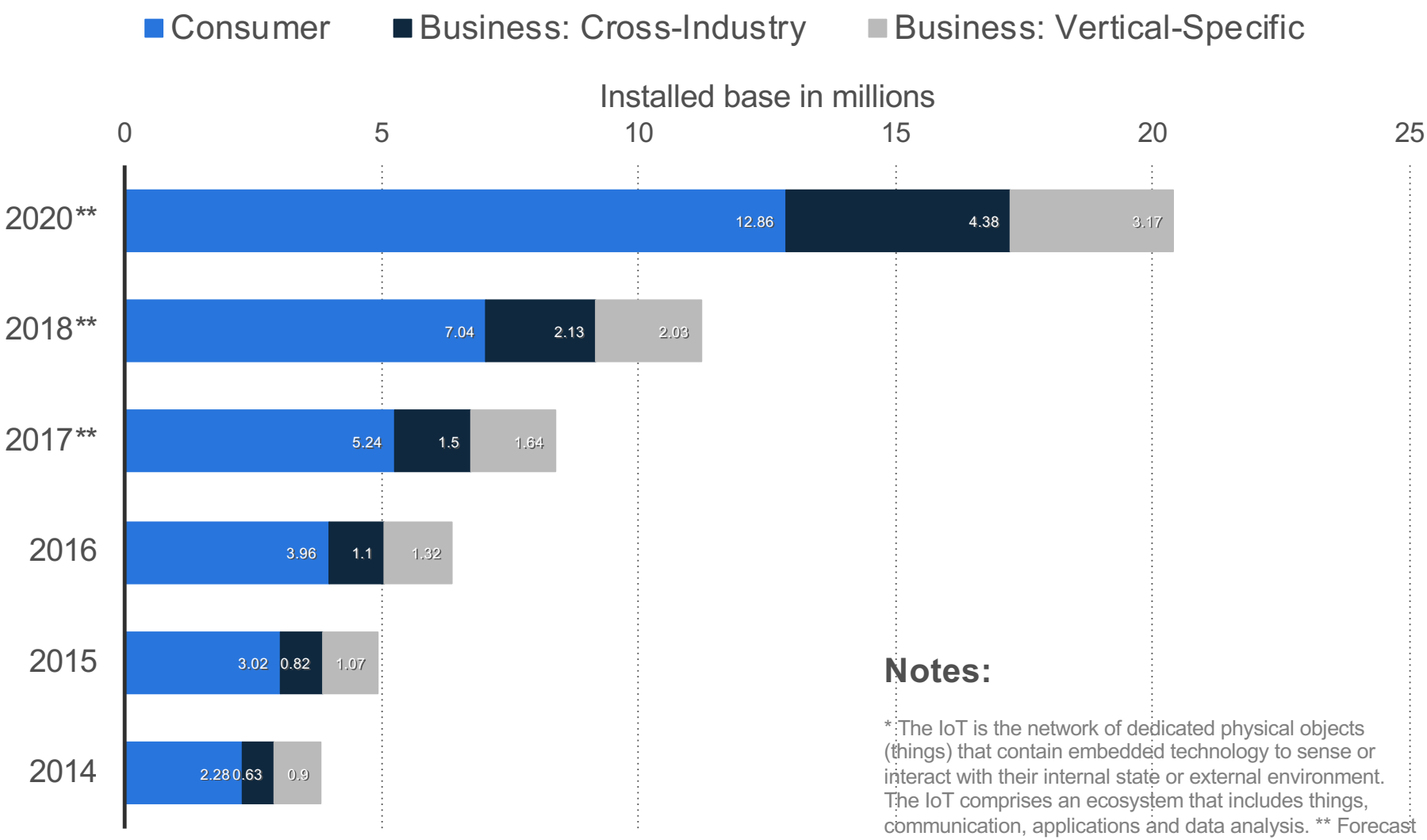
- 7 SDO (ETSI, ATIS, TIA, CCSA, TTA, ARIB, TTC): **OneM2M** (since 2012)
- ETSI: **M2M** service layer standard (published Jan 2012)
- IETF: **CORE** (Constrained RESTful Environments), ROLL, RPL, 6LoWPAN, CoAP
- **3GPP Machine Type Comm. (MTC)**

- OpenADR (Open Auto-Demand-Response) for smart grids

- ITU-T: USN (Ubiquitous Sensor Networks)
- ISO/IEC: WGSN (Working Group on Sensor Networks)

- IEEE 802.14.5, WirelessHART, ZigBee, DASH7, Bluetooth, UWB,
- Sigfox UNB (10 - 1000 bps)
- The LoRa Alliance

The IoT* units installed base by category from 2014 to 2020

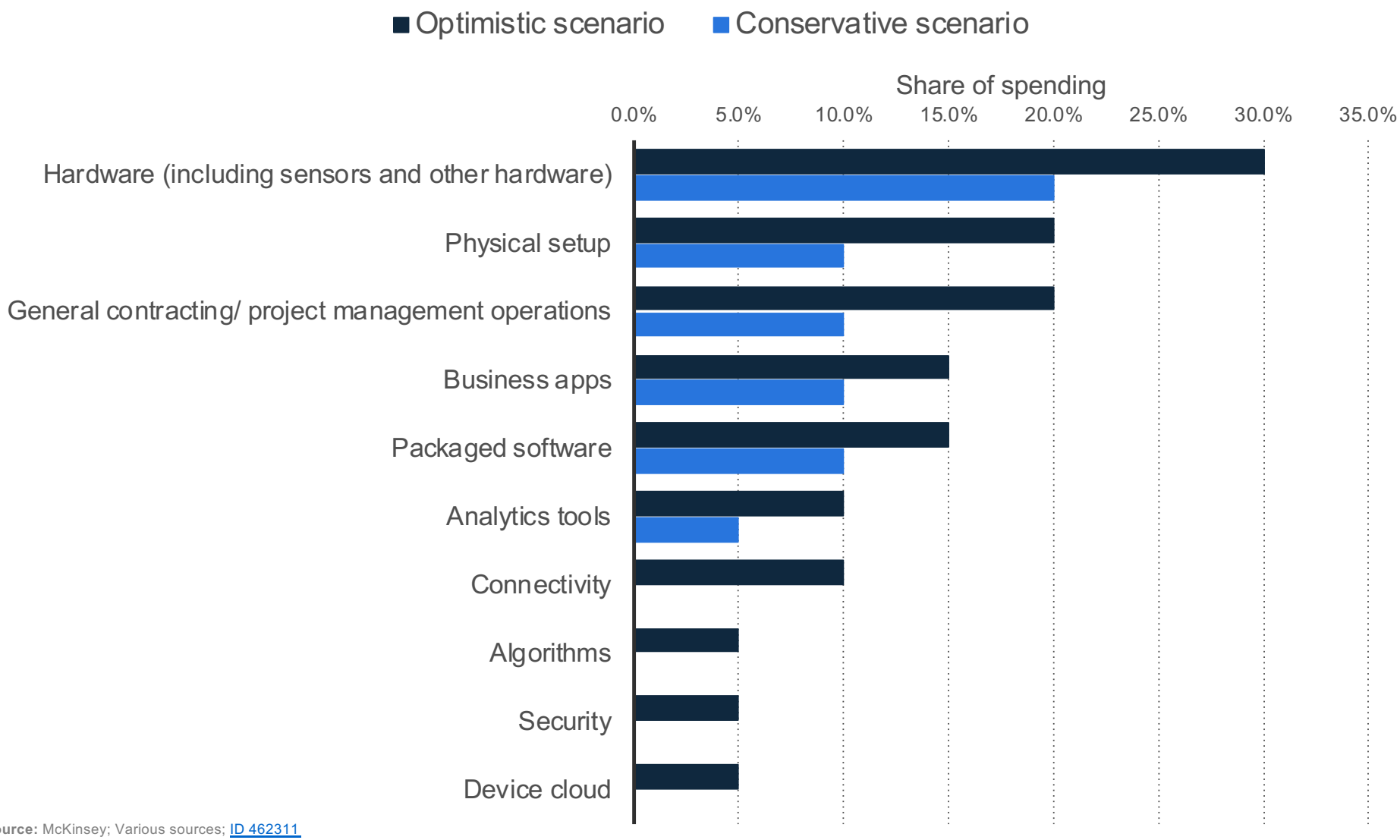


Notes:

* The IoT is the network of dedicated physical objects (things) that contain embedded technology to sense or interact with their internal state or external environment. The IoT comprises an ecosystem that includes things, communication, applications and data analysis. ** Forecast

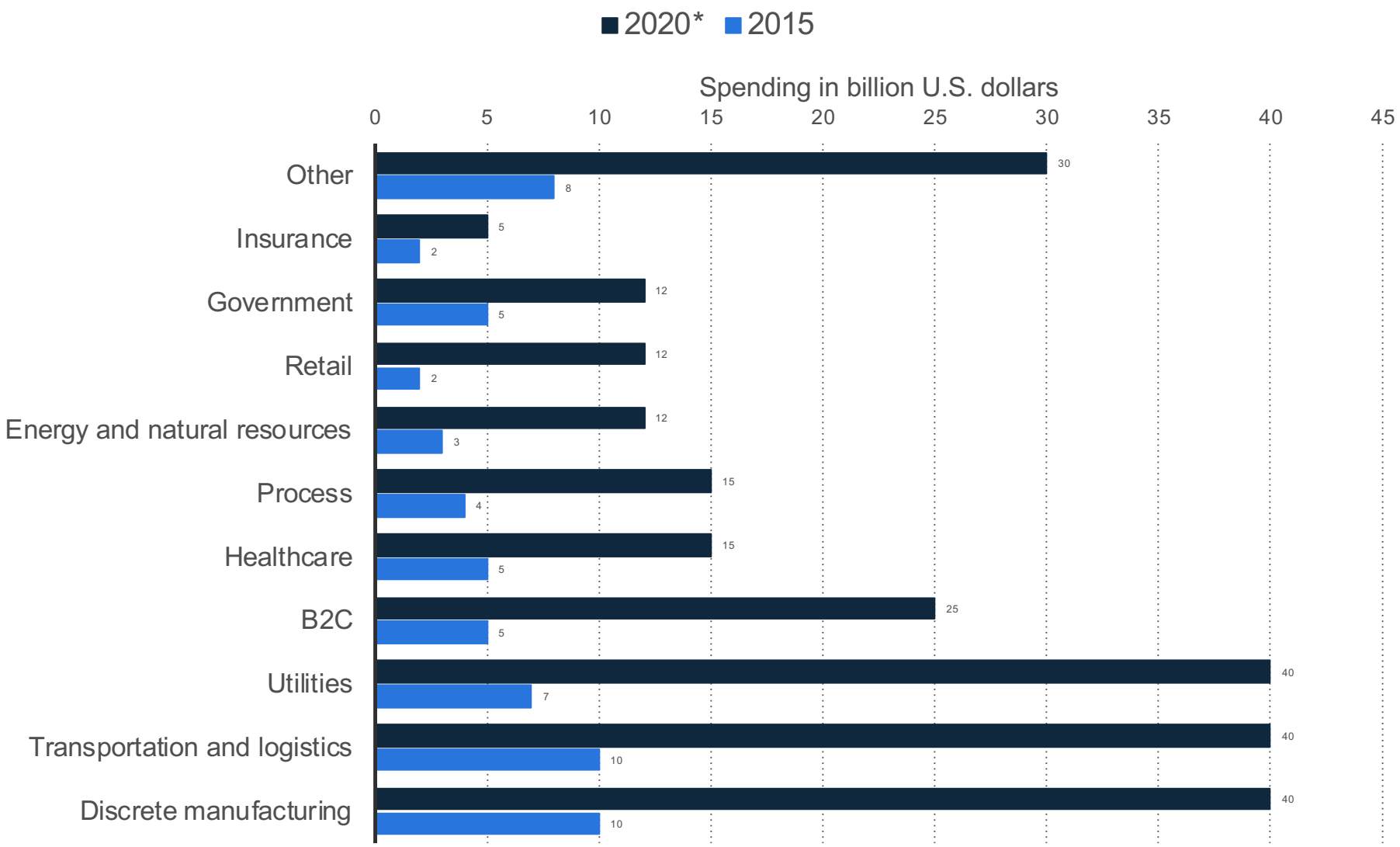
Source: Gartner; ID 370350

Estimated IoT technology spending breakdown in 2015, by scenario



Source: McKinsey; Various sources; ID 462311

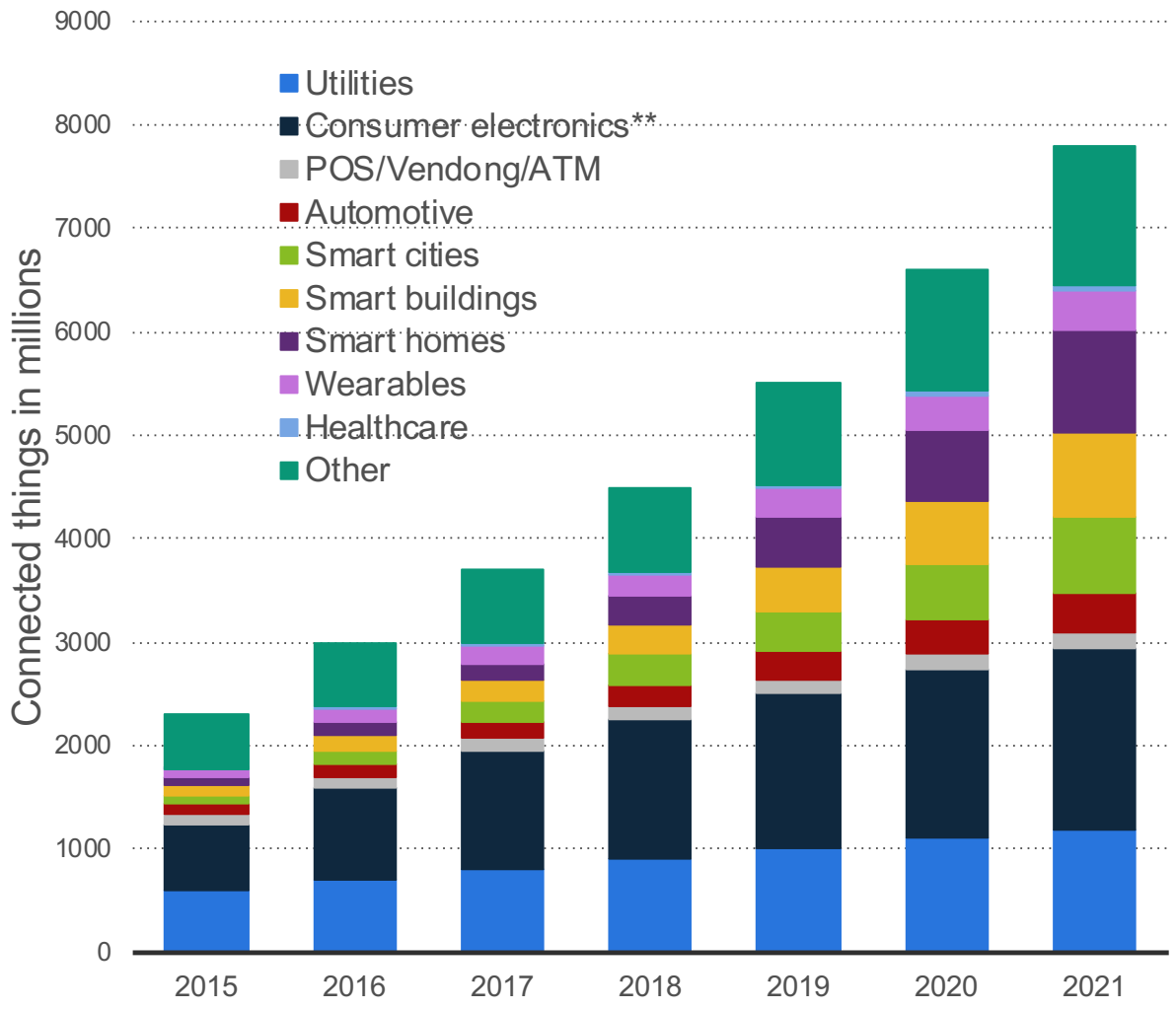
Spending on IoT worldwide by vertical in 2015 and 2020 (in billion U.S. dollars)



Source: BCG; ID 666864

IoT Market

- Forrester Research states that the M2M market will be "the biggest growth market of the next 5 to 20 years."
- Berg Insight: Shipments of cellular M2M devices are forecasted to grow 19.2% p.a.



IoT@ HAW-LA


Waspmote


iWatch

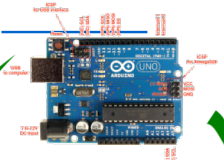

Google glasses


Quadrocopter


Smart phones

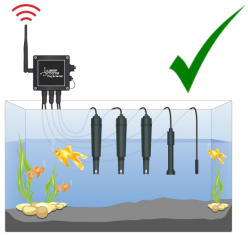

Lego

Libelium


Arduino

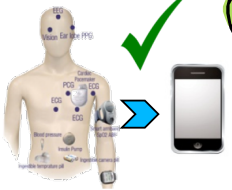

Cooking Hacks

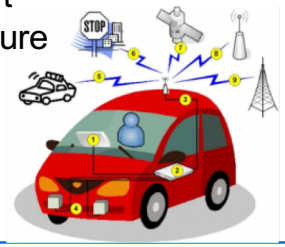

Raspberry

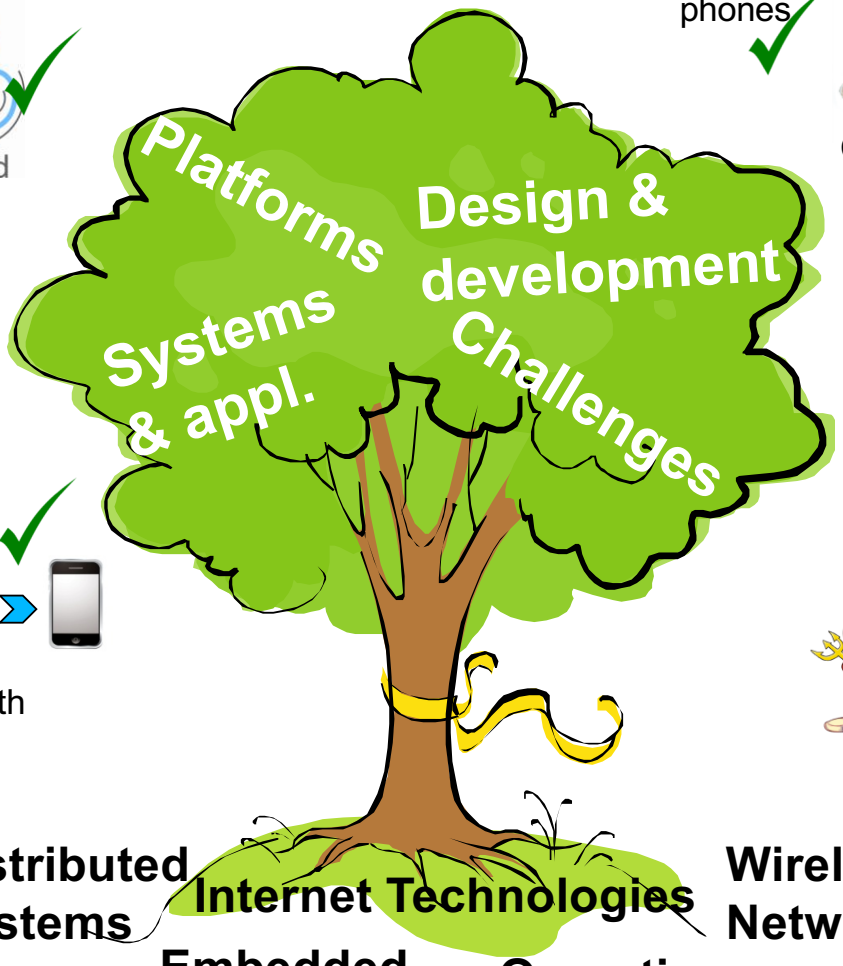

Smart Aquarium


Smart Agriculture


Smart city


eHealth


Mobility



Smart Objects/Things

Outline – Lecture

- LEC1: Introduction
- **LEC2: Smart Objects**
- LEC3: Raspberry-PI and Arduino
- LEC4: Information Models (Eclipse Vorto)
- LEC5: Connectivity for IoT
- LEC6: 6LoWPAN
- LEC7: IoT App Protocols: MQTT
- LEC8: Scaling MQTT (HiveMQ)
- LEC9: IoT App Protocols: IETF CoAP
- LEC10: IoT App Protocols: OMA LWM2M
- LEC11: IoT Application Development
- LEC12: IoT Cloud
- LEC13: Security / OS / Energy-Efficiency
- LEC14: Exam preparation

Chapter Outline

- Smart Objects
  - Selected Examples
  - Definition
  - Requirements
  - Technolgies
  - Applications and Business Models
- Smart Home
  - Definition and Key Applications
  - Solution Architecture
  - Interoperability Gap
- Smart Thermostats
  - The 1885 Thermostat
  - Programmable Thermostats
  - Smart Thermostats

Repetition!

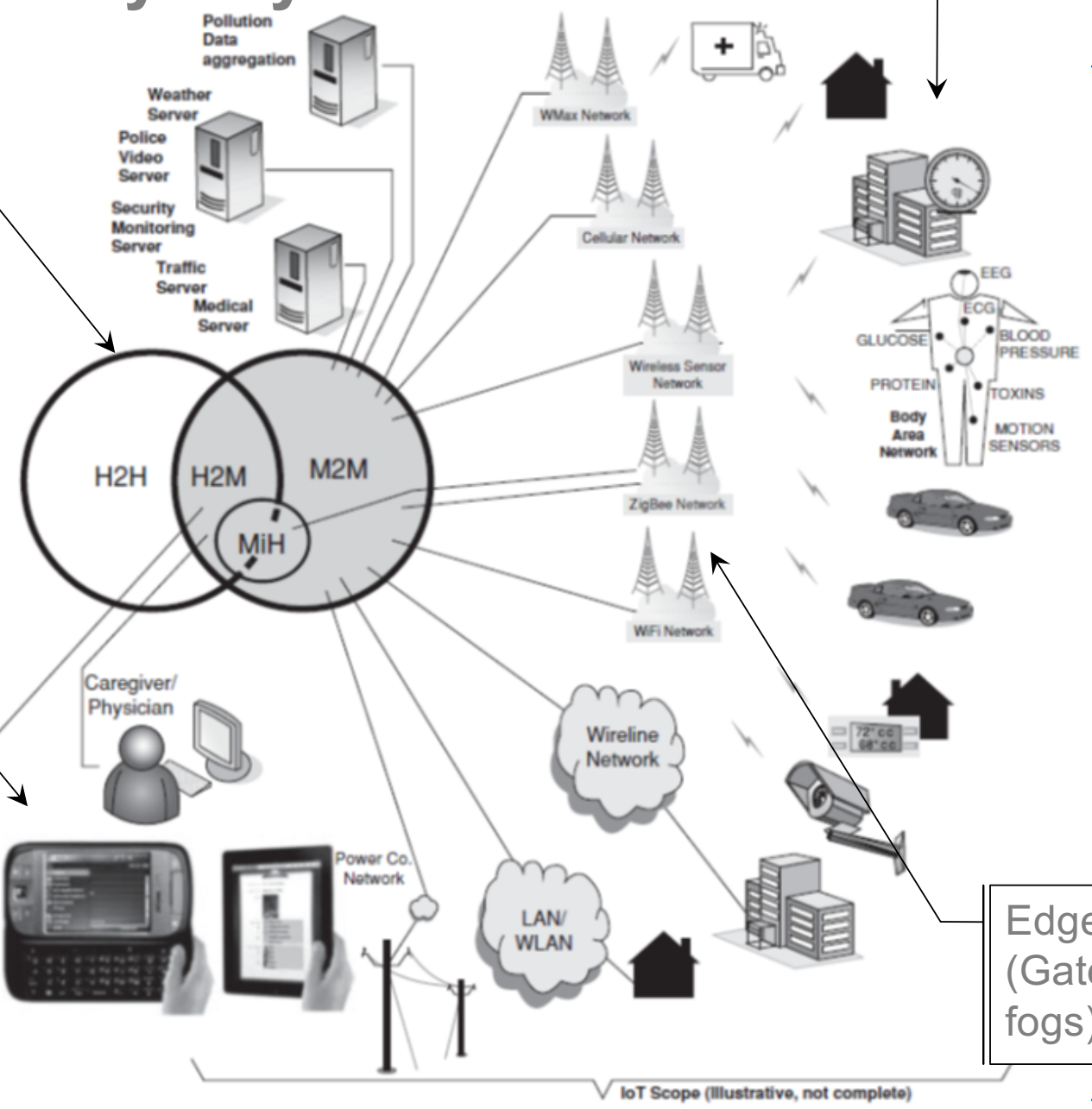
Scope of IoT by way of Illustration

Smart devices

Backend (server, cloud,…)

Endusers

Edge devices (Gateways, fogs)



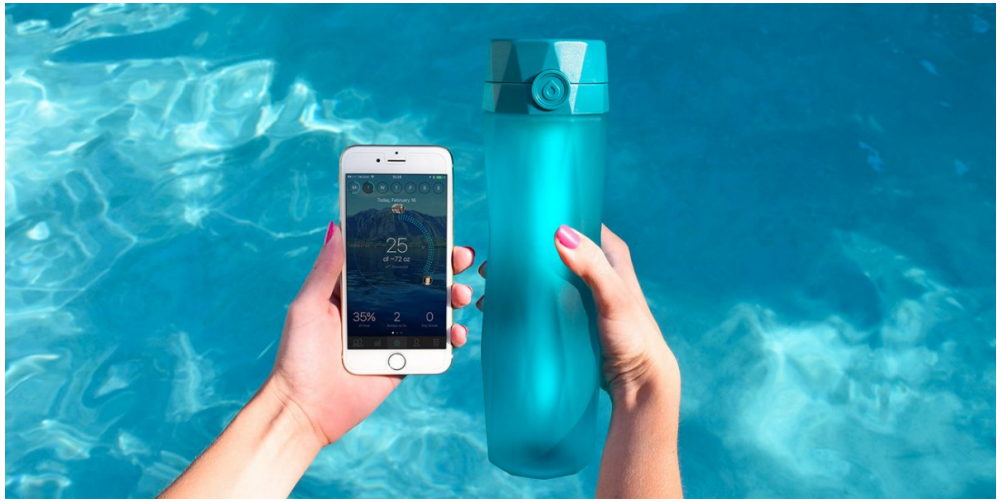
SRC: Minoli „BUILDING THE INTERNET OF THINGS WITH IPv6 AND MIPv6“, 2013

Smart Gadgets

- The Quirky Egg Minder

$ 12

- The Hidrate Spark $ 55



Smart Gadgets

- The Onvi Prophix isn't the only "smart toothbrush"

  $ 400

- The Flosstime, a smart floss dispenser

  $ 30

Smart Gadgets

- The June is a smart countertop oven

$ 1500

- The HapiFork is a Bluetooth-enabled “smart fork”

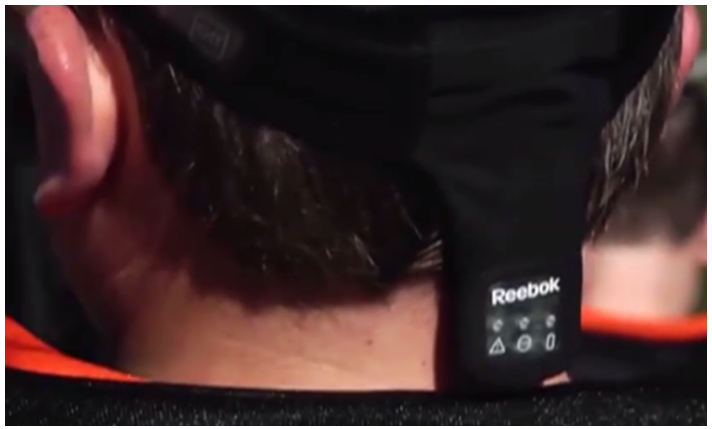
$ 65

Smart Gadgets

- The Belty Good Vibes

$149

- The Verizon Smart Football Helmet (CES 2013)



Smart Gadgets

- The Oombrella

$ 80

- The Porkfolio Piggy bank

$ 19

Smart Gadgets

- iTouchless Automatic Trash Can

- The Bruno, the smart trash gadget

- Kinect-powered trash can

- ..

Niels‘ law
Moore‘s law

Minutarized + low cost

- Processing (comp. + storage)
- Communication
- Sensors
- Actuators
- Batteries

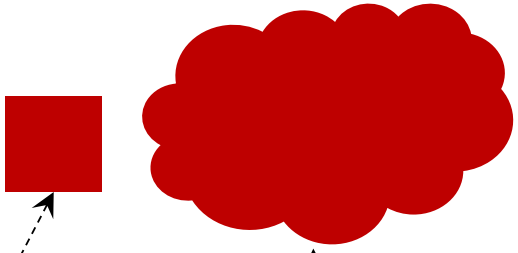
Metcalfe‘s law

Prof. Elgar Fleisch:

Thing + IT = Function + Services

„Internet“

User (IntranetoT)



User (InternetoT)

Gateway/ Edge device

Backend/ Cloud

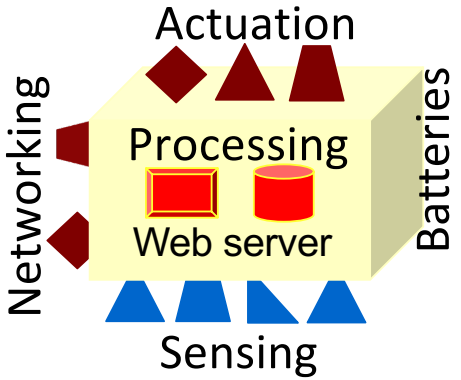
User Interface (UI) / Smartphone app

Definition and Classification

- Definition
  - Thing that is simply connected to internet … A thing that plays an active role in the network… A thing that has a counterpart in the cyber world.
- Classification criteria: Awareness, representation, interaction
- Common classification (Awareness)
  - Activity-aware objects: Maintain logs info about work activities of their own
  - Policy-aware objects: Understand events and activities w.r.t pre-defined policies
  - Process-aware objects: Understand inbuilt processes and provide context aware guidelines
- Recent smart objects also exhibit pseudo social behavior!

Requirements on the “IT-ilization” of Things

- Easy to embed to physical objects
  - Compact & easy to add-on even for small sized objects
  - Affordable even for low-cost objects (Cost should not exceed 3%)
  - Energy efficient in particular for battery-powered objects



- Minimally intrusive, no disturbance of main functionality

- Easily interchangeable

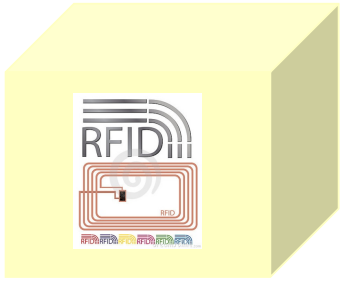
The Early Smart Things

Bar/QR

(Quick Response) Code

RFID (Radio-Frequency IDentification)

Tag/Transponder



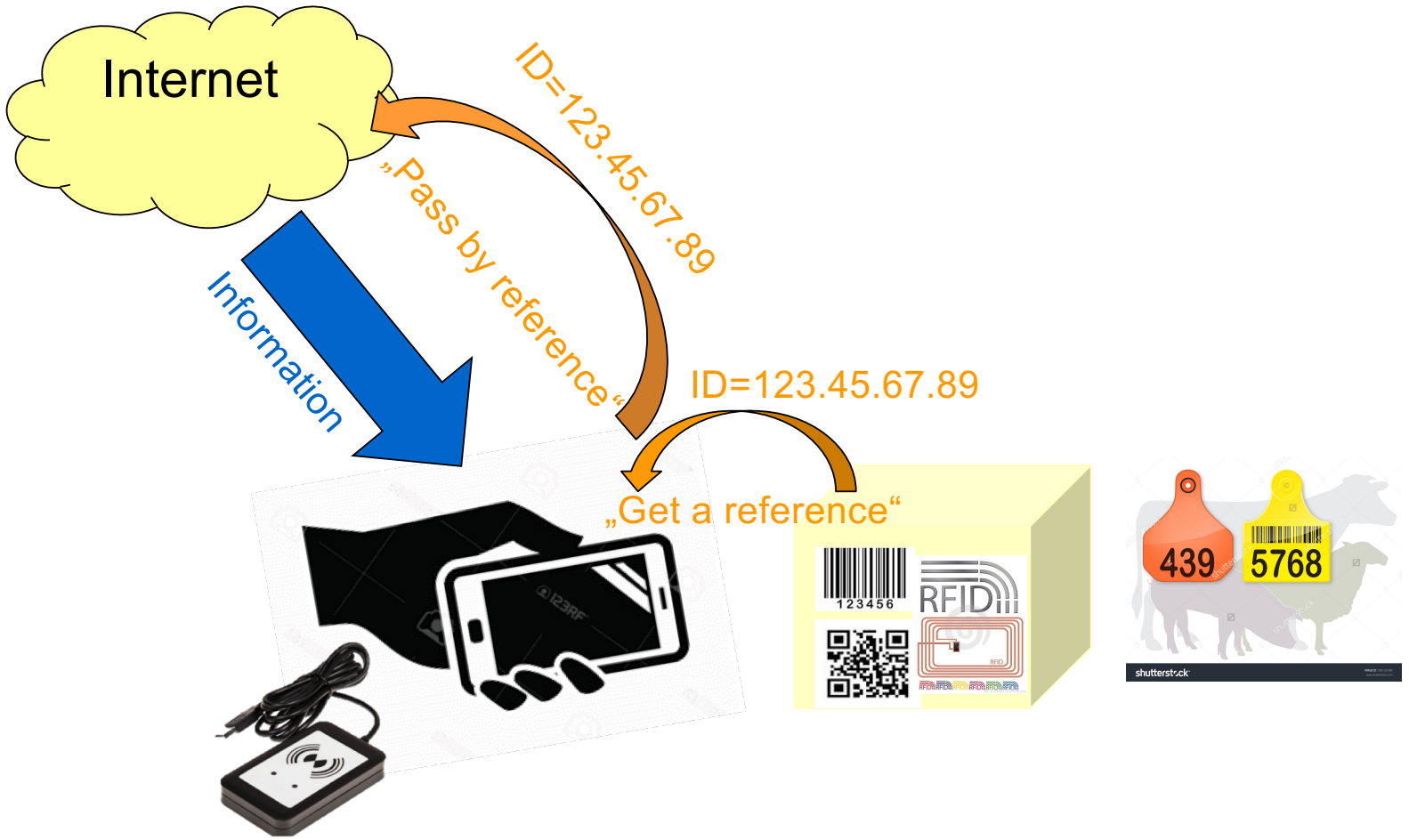
BLE (Bluetooth Low Energy) Tags / NFC (Near Field Communication) (Chapter 5)

Linking a thing to a web service (homepage, cloud etc) for:

- Product rating
- Self-checkout
- Stock-taking
- Dynamic pricing
- Proof of origin
- Replenishment alert
- Fair trade check
- Political shopping
- Counterfeit check

„Pass by Reference“

Intuitive Interaction: Access information through pointing / „touch“

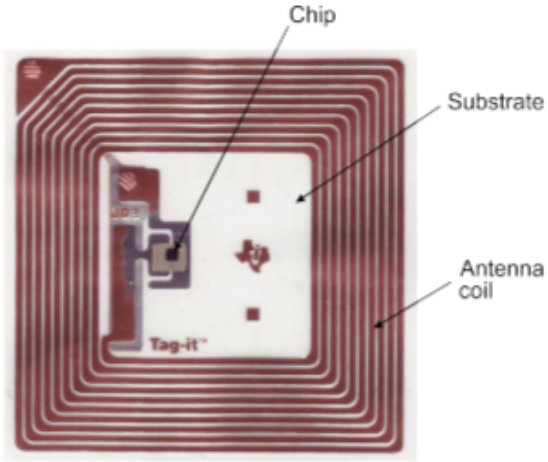


QR-Codes

- Free tools to generate QR codes:
  - http://www.qrstuff.com/
  - http://goqr.me/
  - www.unitag.io/qrcode
- Commercial tools
  - http://www.visualead.com/
- Scanner apps (Android, iOS, win10)

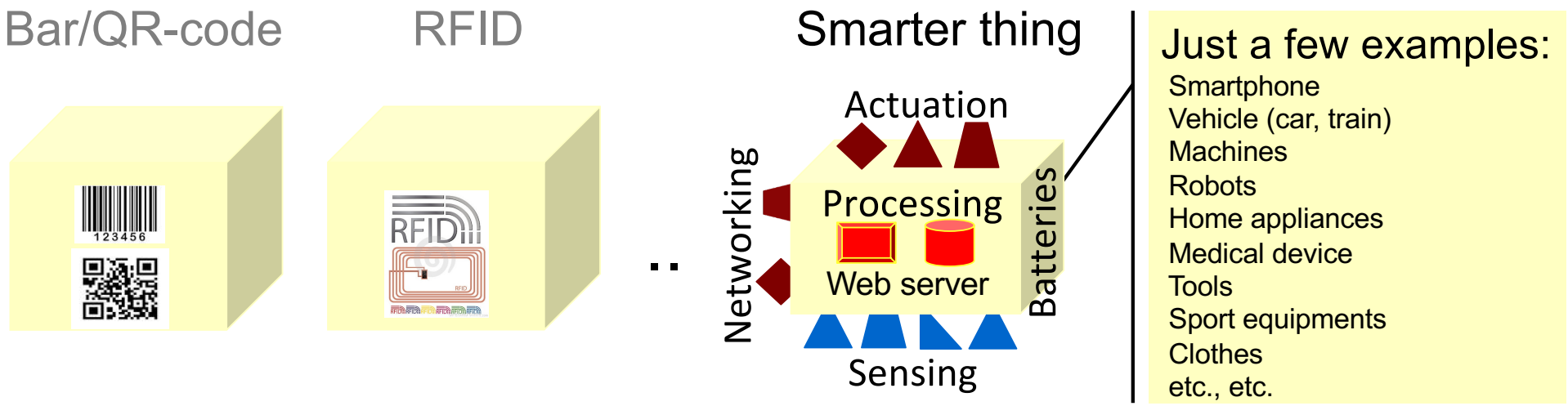
RFID Systems

- Reduced size, weight, energy consumption, and cost of radio
- RFID system: Reader(s) + unique tags as identifier
  - Monitor objects in real time without the need to be in Line-Of-Sight
    - Logistics, e-health, security
    - Mapping real world -> virtual world
- An RFID tag is a small chip with antenna
  - Receiving signals, and transmitting the tag ID
    - Induction, current
    - Signal power recv. divided by power transm. = ID
  - Passive (low-cost things), semi-passive (with battery) and active (with battery, for higher cost things)
  - UHF: 300 MHz- 3GHz, 860 – 960 MHz



RFID Tag

Smarter Things (Software Defined Things!)

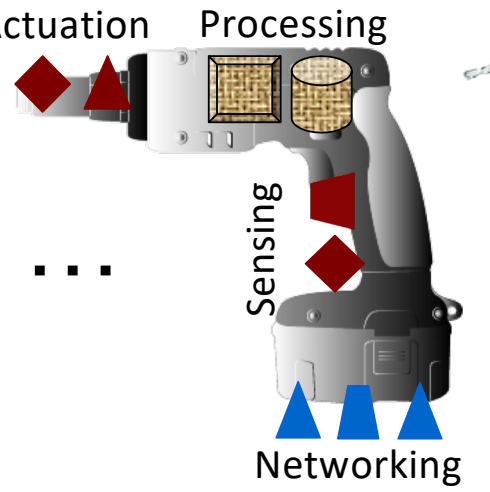


Enabling things with sensing/actuation, computation & networking capabilities:

- Advanced product lifecycle mgmt: More individualized products, estimated residual lifetime, predictive maintenance, usage behavior, etc.
- Context-aware behavior: Tailor functionality to environment/lifetime, ..

Smartening Tools (Software Defined Tools!)

Code (Bar, QR..) Tag (RFID, ThinFilm..) Smarter Tools Product Examples

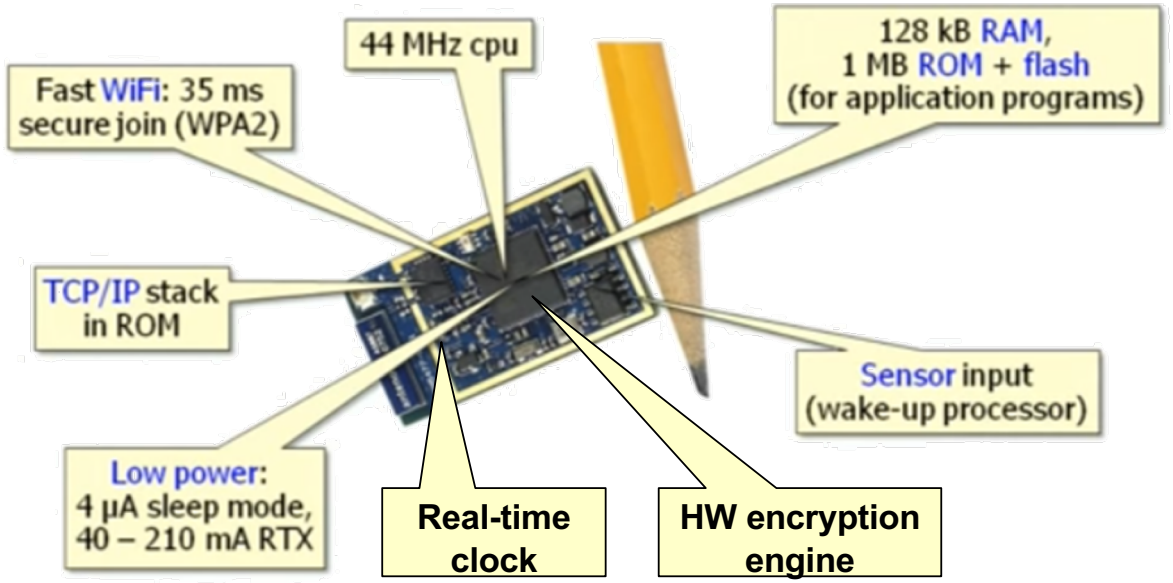


- Enabling assets/tools with sensing/actuation, computation & networking capabilities:
  - Sensing: Cameras, GPS, light intensity, micro, etc.
  - Actuation: Light, sound, reconfiguration of standard functions, etc.
  - Networking (Chapter 5)

→ Provide Services: Tracking, proactive maintenance, usage monitoring, rent/sharing, payment per use, etc

Bosch Rexroth Nexo
- On-board controller for Sensing of torque and angle of rotation
- WiFi
- Barcode scanner

Low-Power, Low-Cost WiFi Modules



G2M 5477
(About 10$)

.. others

Products in the IoT World

- Are much smarter
- Know their life cycle
- Are customizable for customer requests
- Can understand and interpret user behavior
- Can be maintained remotely and preventively (networked products change the service model from reactive to proactive)

Manufacturer in the IoT World

- The manufacturer receives useful information about the long-term use behavior of his product under real operating conditions
- Away from the product seller to the service provider
- Stay in constant contact with your customers throughout the product lifecycle
- Dependence on trading partners is decreasing, customer loyalty is being improved

Customers in the IoT World

- IoT starts with the customer
- Customers have moved from the end of the production chain into the production and logistics process. As a customer I want:
  - Know the status of my order in real time
  - Flexible, situational and individual solutions
  - An individual, fast customer service and an uncomplicated service experience
- Age of the Customer: service orientation instead of product orientation

New Business Models

- Individualisation of products
  - „We produce what we sell“ instead of what „we sell what we produce“
- Services instead of products
  - Pay as your Drive (PAYD)
  - Pay-per-use
    - → Extensive sharing
      - Dynamic tolling on roads depending on pollution zones, traffic, law enforcement rules,
      - Etc.
    - → Renting instead of selling
      - Pay-per-volume
      - Etc.
- Fine-granular product lifecycle management
  - Traceability
  - Predictability
  - Etc.

Shareconomy: A Driver for IoT

Shareconomy already successful in the cyber world: Cloud computing

Resource sharing in the cyber physical world

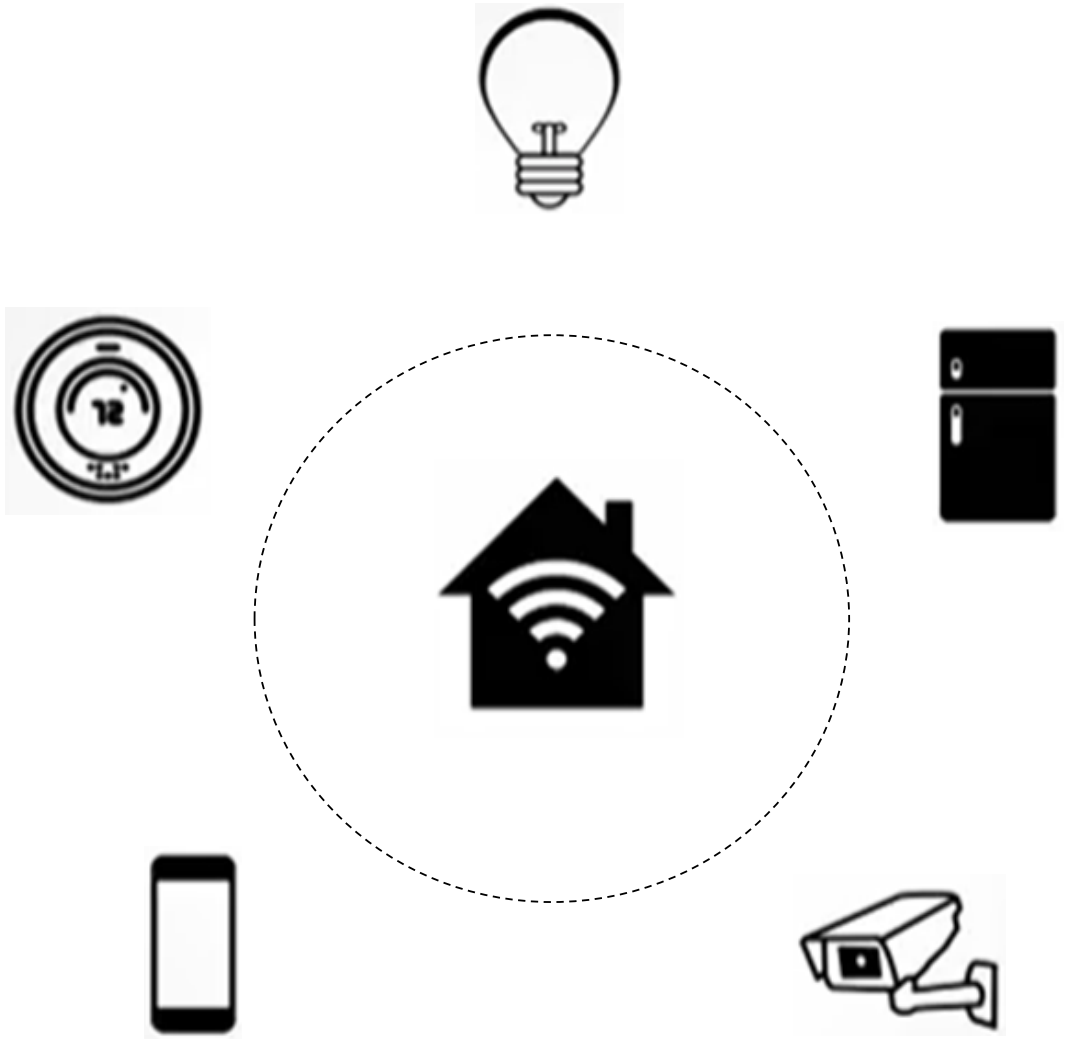
- Car sharing
- Bike sharing
- ..

Why not sharing lower cost objects?

- Machines
- Tools
- Toys
- ..

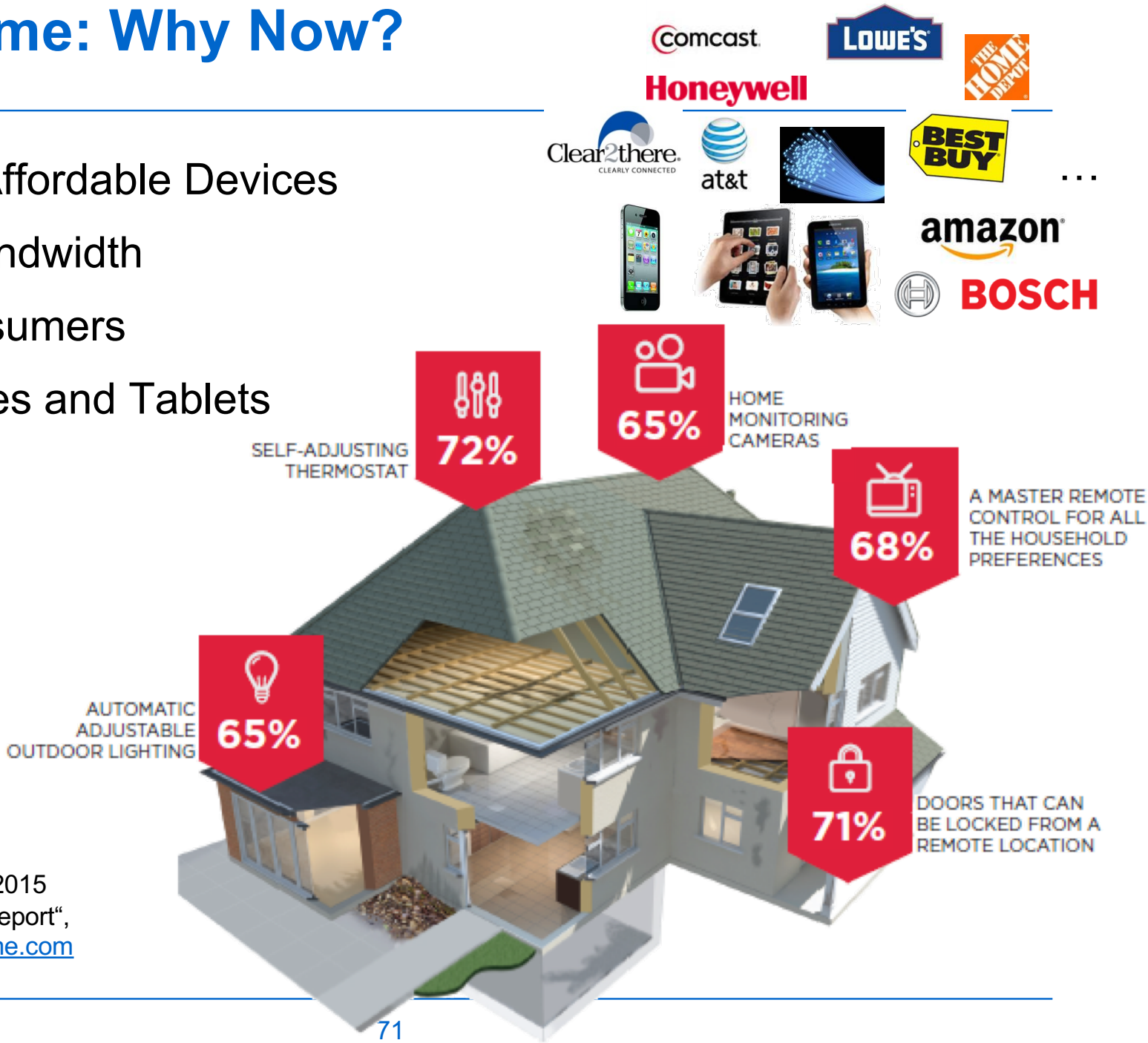
„Smart Home: Thermostat as an Illustration of a Smart Object“

Smart Home



Smart Home: Why Now?

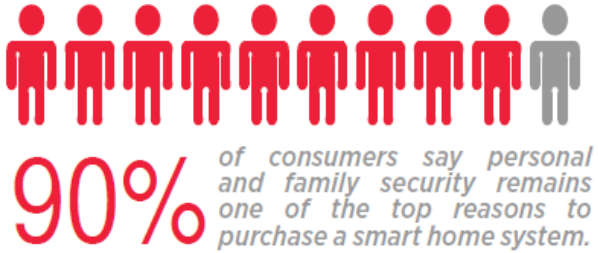
- Access to Affordable Devices
- Plentiful Bandwidth
- Savvy Consumers
- Smartphones and Tablets



SRC: iControl Networks: „ 2015 State of the Smart Home Report“, www.StateOfTheSmartHome.com

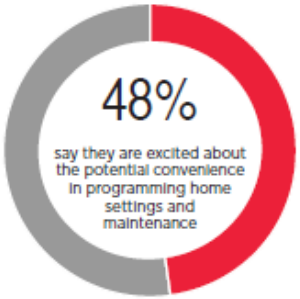
Smart Home Applications

- Security

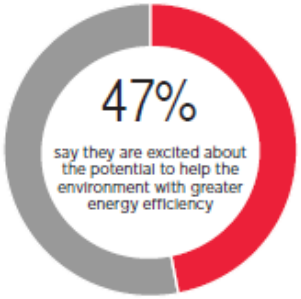


- Energy efficiency (example later)

- Home maintenance

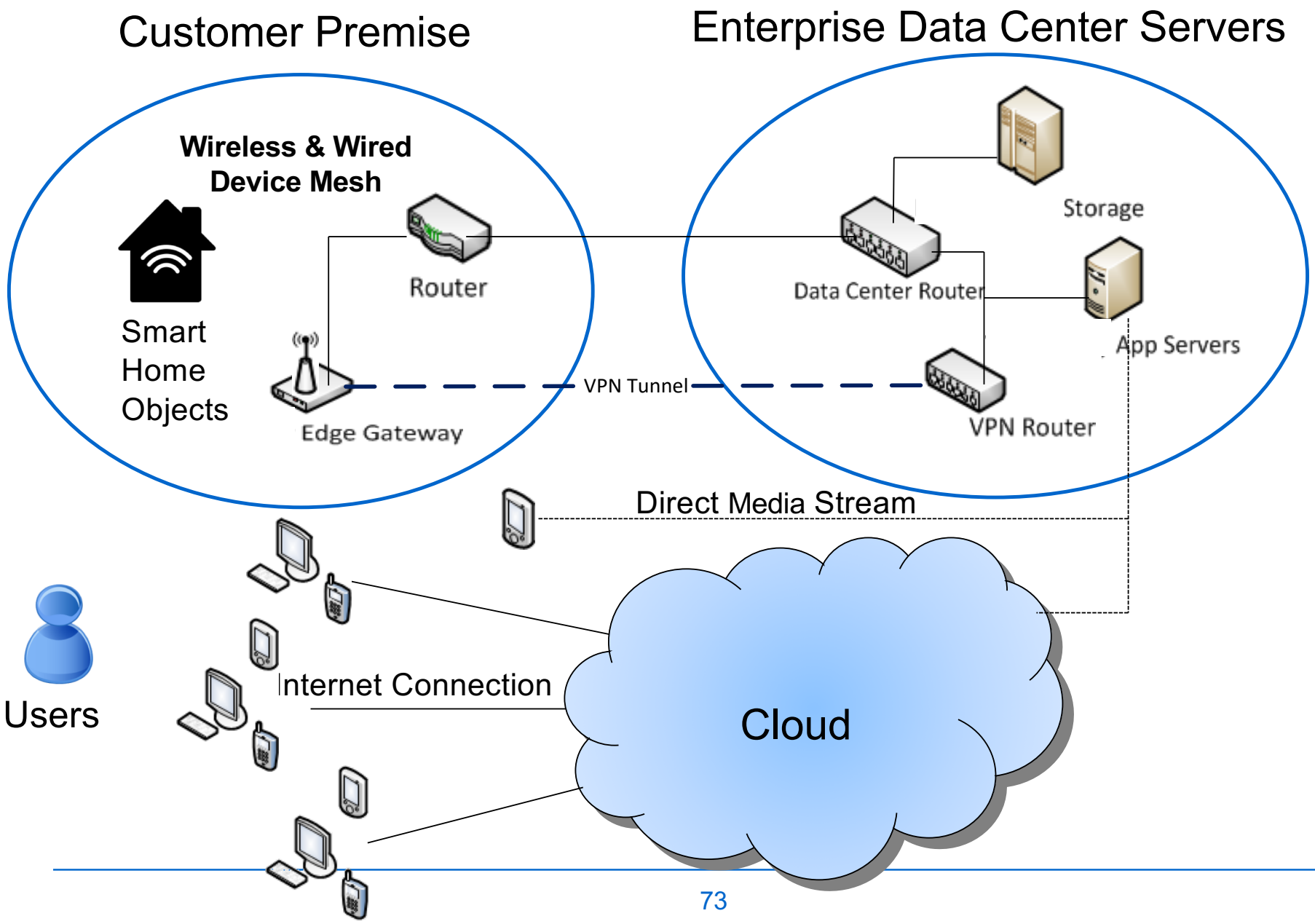


- Environment protection



SRC: iControl Networks: „ 2015 State of the Smart Home Report“, www.StateOfTheSmartHome.com

Technology Design



Major Commercial Gateways & Plattforms

- Apple Homekit
  - http://www.electronichouse.com/daily/smart-home/homekit-coming-will-apple-home-automation-different/
- Deutsche Telekom Smart-Home-Box
  - Telekom Smart Home Base Qivicon
- Mosaic Gateway (Bosch, ABB, Cisco)
- HUE Philips
- Belkon WeMo switch
- Dropcam

Interoperability:
Accept the Diversity, Break the Silos

- Poor interoperability is the main barrier for a sustainable ecosystem
  - HW-agnostic platforms

- Eclipse SmartHome™ project
  - Addresses a vast variety of comm. mechanisms
  - Serves as an abstraction and translation framework that makes interaction possible across system and protocol boundaries

- Other initiatives
  - openHAB (open source) Run on Linux, OS X, Windows, Java 8, Raspberry Pi
  - Mosaic (Bosch, ABB, Cisco)

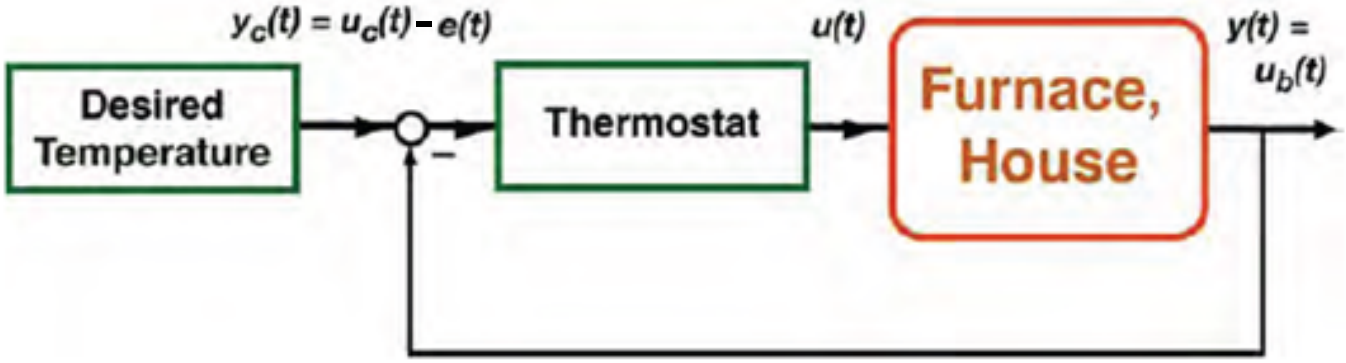
Smart Thermostat

1885 Thermostat

- Albert Butz (started a company that became Honeywell in 1927)
  - Bimetal plate (sensor/control)
  - Motor to move the furnace damper
- On-off control based on threshold (room predetermined temperature)
- Thermostat switching on -> main motor shaft turns one-half revolution opening furnace's air damper **to let in air (fire burn hotter)**.
- Thermostat switching off -> motor shaft turns another half revolution closing furnace's air damper.

http://www.travelfilmarchive.com/item.php?id=13013

Thermostat Control Logic



Control Law *[i.e., logic that drives the control variable, u(t)]*

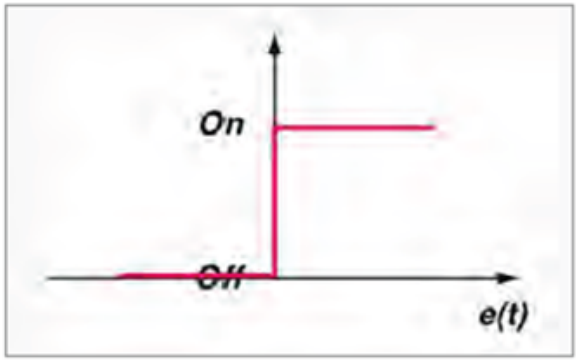
$$e(t) = y_c(t) - y(t) = u_c(t) - u_b(t)$$

$$< \text{Thermostat} >$$

$$u(t) = \begin{cases} 1\,(on), & e(t) > 0 \\ 0\,(off), & e(t) \le 0 \end{cases}$$

- y_c: Desired output variable (command)
- y: Actual output
- u: Control variable (forcing function)
- e: Control error

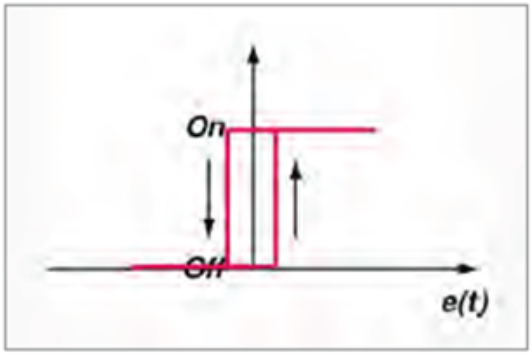
Thermostat Control Logic (Cont.)



$$u(t) = \begin{cases} 1\,(on), & e(t) > 0 \\ 0\,(off), & e(t) \le 0 \end{cases}$$

- ...but control signal would "chatter" with slightest change of temperature
- Solution: Introduce *lag* to slow the switching cycle, e.g., *hysteresis*

$$u(t) = \begin{cases} 1\,(on), & e(t) - T > 0 \\ 0\,(off), & e(t) + T \le 0 \end{cases}$$



.. Programmable Thermostats:
Clock thermostats, Digital thermostats, Digital thermostats with PID controller..

PID Controller

- PID stands for
  - Proportional
  - Integral
  - Derivative

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(t')\,dt' + K_\mathrm{d} \frac{de(t)}{dt}$$

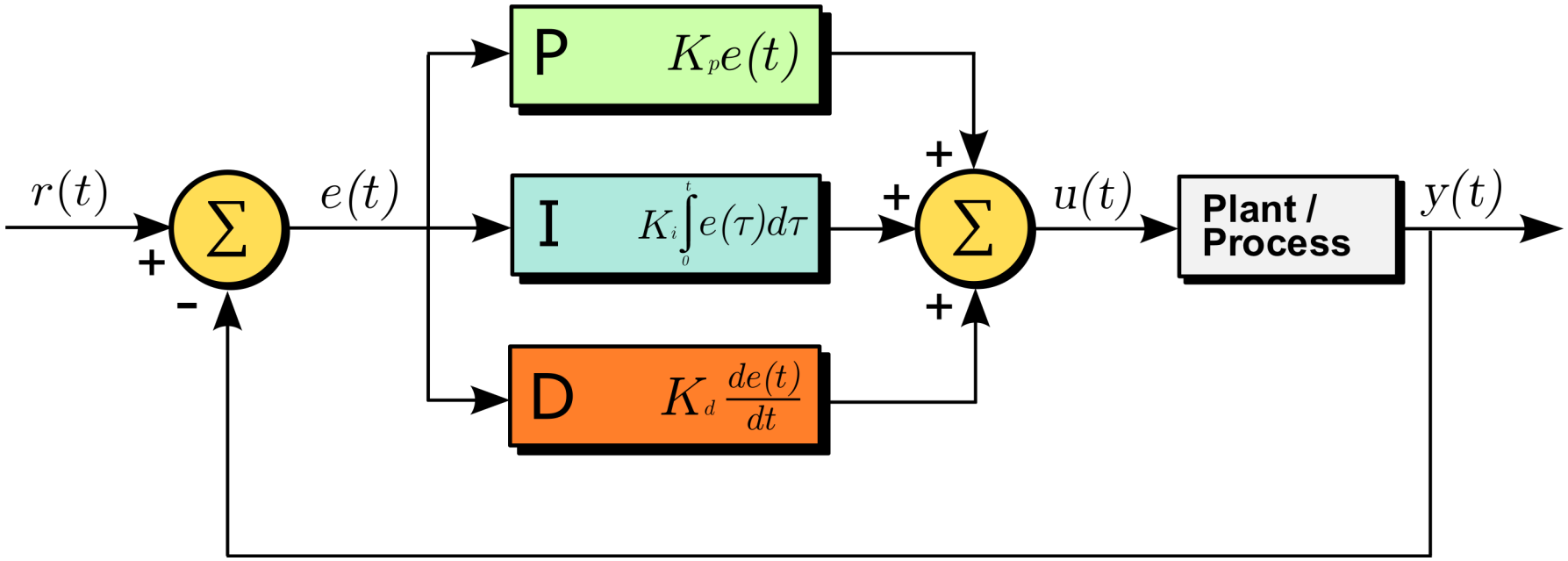


Image credits: CC Wikipedia

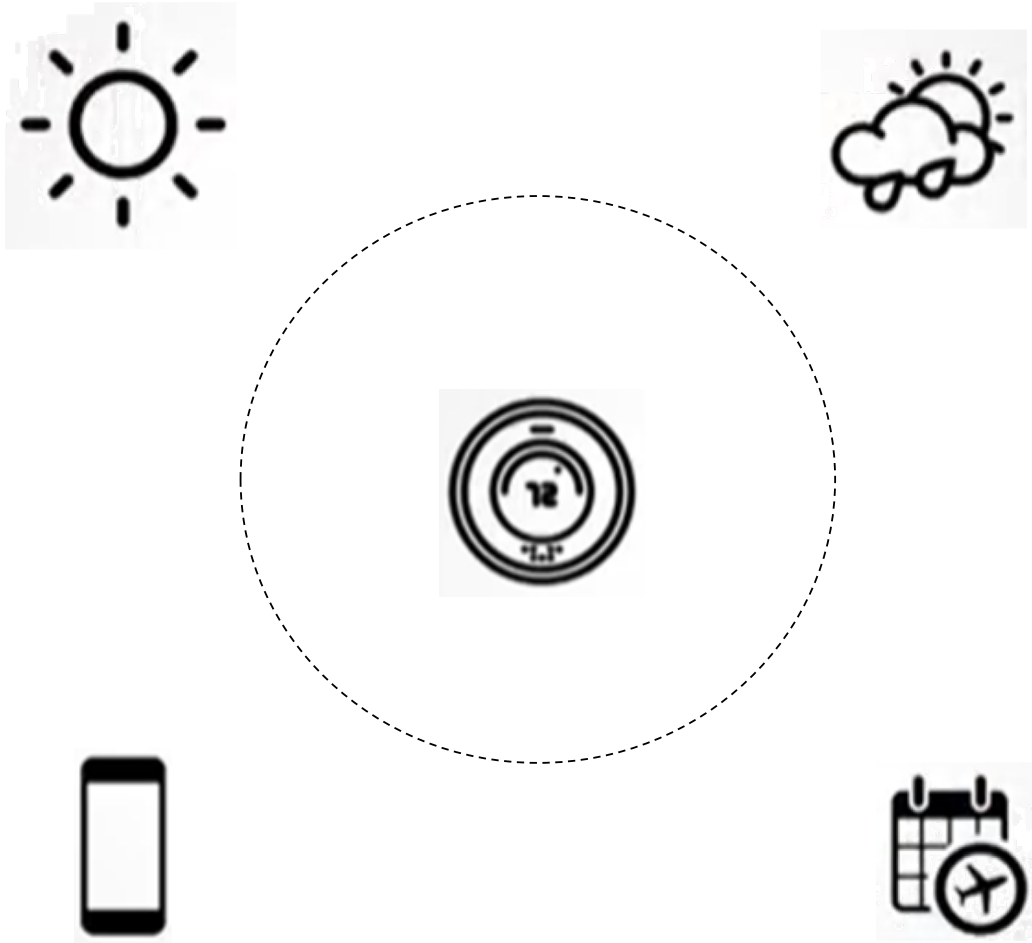
Programmable Thermostats

- The U.S. Department of Energy has estimated that the average homeowner can save between 5 and 20% of their heating and cooling costs by using a programmable thermostat.

- Some field studies have shown **no significant savings** in households using programmable versus non-programmable thermostats. These studies point out that programmable thermostats are only used successfully by about 50 percent of home occupants, although estimates vary among the different studies.

- The programmable thermostat itself does not guarantee energy savings; **savings depend on how the device is programmed and used in each household** versus how a manual thermostat would be used in that household.

Smart Thermostat: “do the thinking for their owners”

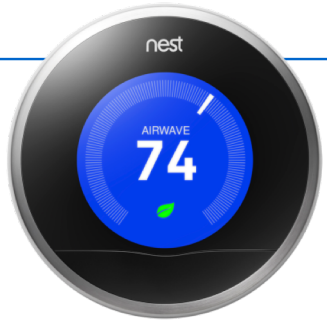


Smart Thermostat

- High degree of automation
  - Information gathered by sensors and other data acquisition devices, monitoring weather forecasts
  - Communications incl. Internet connectivity
  - Self-programming and adaptive learning (interview-based programming), **“set and forget” approach**
- User interface
  - Home Energy Displays (HEDs) (feedback and recommendations)
  - Dashboard or portal
  - Smartphone App
- 20 - 40% energy savings

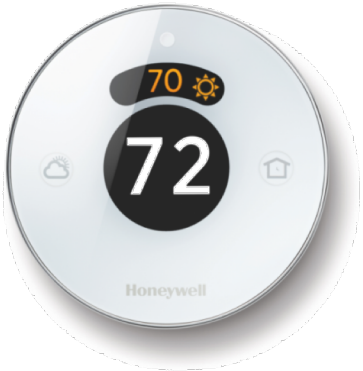
Smart Thermostat – Current products

- Nest ($200), 2011+



- Ecobee ($170, $70/sensor), 2007+
  - Temperature & presence sensors

- Honeywell ($250)
  - Interview-based programming



- Others (Filtrete/Homewerks, Emerson, Venstar, Proliphix, Aprilaire, Lux, Robert Shaw, Lennox, Carrier, Bay Controls, Evolve, Hunter, Jackson Systems, ICM Controls, Net/X, Intwine Energy, Schlage Nexia, Enphase Energy, Energate, Trane and LockState)

Conclusions

- Smartening Objects is the main driver of IoT
  - Several techniques
  - New business models: Pay-as-*, pay-per-*
- Smart Home is
  - Profiting from affordable HW and plethora of applications
  - But still struggling with the interoperability gap

Raspberry PI & Arduino Uno

Chapter Outline

- Raspberry PI
  - Specs
  - Setup & Configuration
  - Python
  - Hello World
- Arduino Uno
  - Specs
  - Programming
  - Hello World
- Comparison

RASPBERRY PI

Raspberry PI 3 & the Rest of the World

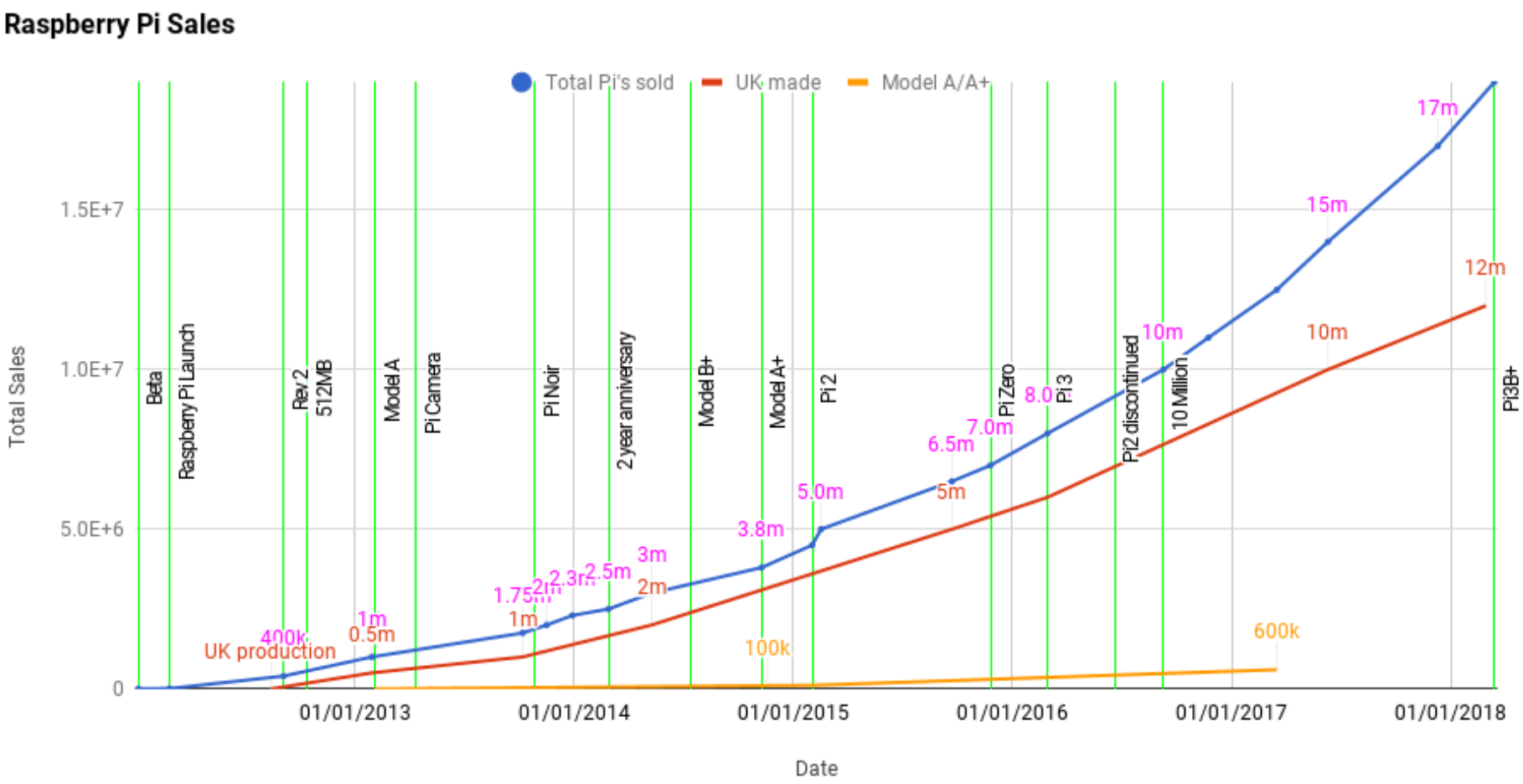
	Raspberry PI, B	Beagle Bone Black	Odroid C1+	Banana Pi	Pine 64	pcDuino3	C.H.I.P
	Raspberry Pi 3 Model B	Beagle Bone Black	Odroid C1+	Banana Pi	Pine 64	Linksprite pcDuino3 Nano Lite	The C.H.I.P
							
CPU	Quad-Core ARM Cortex-A53	AM335x ARM Cortex-A8	Amlogic ARM Cortex-A5	ARM Cortex-A7 dual-core	Quad-Core ARM Cortex A53 64-Bit	ARM Cortex A7 Dual Core	ARMv7 Processor rev 2
CPU Speed	1.2GHz	1GHz	1.5GHz	1GHz	1.2GHz	1 GHz	1GHz
GPU	Dual Core VideoCore IV	3D graphics accelerator	Mali-450 MP2	Mali-400 MP2	Mali-400 MP2 Dual core	Mali-400 Dual Core	No
RAM	1GB	512MB	1GB	1GB	2GB DDR3	1GB	512MB
Onboard Storage	No	4GB	eMMC storage	No	No	No	4GB
Expandable Storage	MicroSD	MicroSD	MicroSD	MicroSD	MicroSD	MicroSD	No
Network Connectivity	10/100 Ethernet	2 X 10/100 Ethernet	10/100 Ethernet	10/100 Ethernet	10/100/1000 Ethernet	10/100 Ethernet	10/100 Ethernet
Wireless Connectivity	WiFi, Bluetooth 4.1	No	Infrared	Infrared	WiFi, Bluetooth 4.1	No	WiFi, Bluetooth 4.0
USB2.0	4 X	4 X	4 X	2 X	2 X	2 X	1 X
HDMI	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Display Connector	Composite RCA, LCD	LCD	No	LCD	LCD	No	LCD
Camera Connector	Yes	No	No	Yes	Yes	Yes	Yes
Audio	HDMI/3.5mm	HDMI/3.5mm	HMDI/I2S Interface	HDMI/3.5mm	HDMI/3.5mm	HDMI/3.5mm	HDMI/3.5mm
GPIO	40 pin	69 pin	40 pin	26 pin	68 pin	14 pin	80 pin
Real Time Clock	No	Optional	On-board RTC function w/ a backup battery connector	No	Yes	No	No
OS Support	Linux/DebianNondroid/BSD/RISC/ webOS	Linux/Ubuntu/Debian/Gentoo/ ArchLinux/LinuxCNC/Minix/XNU/ FreeBSD/Nintendo/Symbian/QNX/ Windows CE	Ubuntu/Android	Linux/Ubuntu/DebianNondroid	Android/Ubuntu	Ubuntu/Android	Linux/Debian
Size L X W (mm)	96 x 71	87 X 53	85 X 56	92 X 60	127 X 79	92 X 54	38.1 X 58.4
Power	5V1/2.5A	5V/460mA	5V 2A	5V	5V	5V	5V
Cost	$35	$45	$35	$35	$29	$35	$9

OS

COST

Src: www.pi-top.com

Raspberry PI is the Most Popular

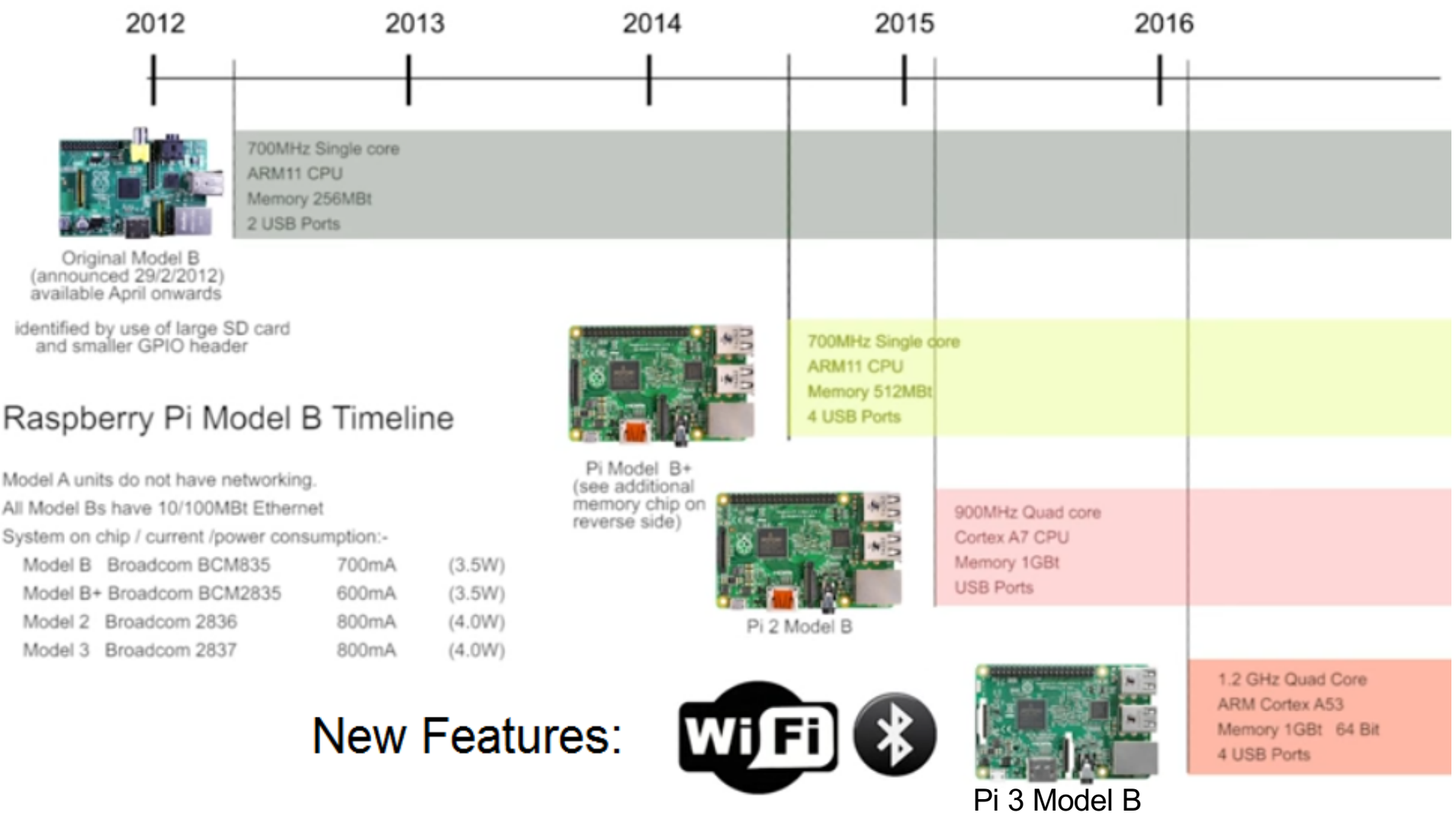


SRC: https://docs.google.com/spreadsheets/d/1zWwpcckDEEVAhNH3y7JQGxxbjP42nUywPOzDWr1fH28/edit#gid=0

Two Models

- Model A
  - Lower-spec variant of the Raspberry Pi (256 MB RAM, 1x USB port, no Ethernet)
  - Lighter and consumes less power
  - Suitable for embedded projects
    - Robotics
    - Projects where weight and low power are paramount
- Model B

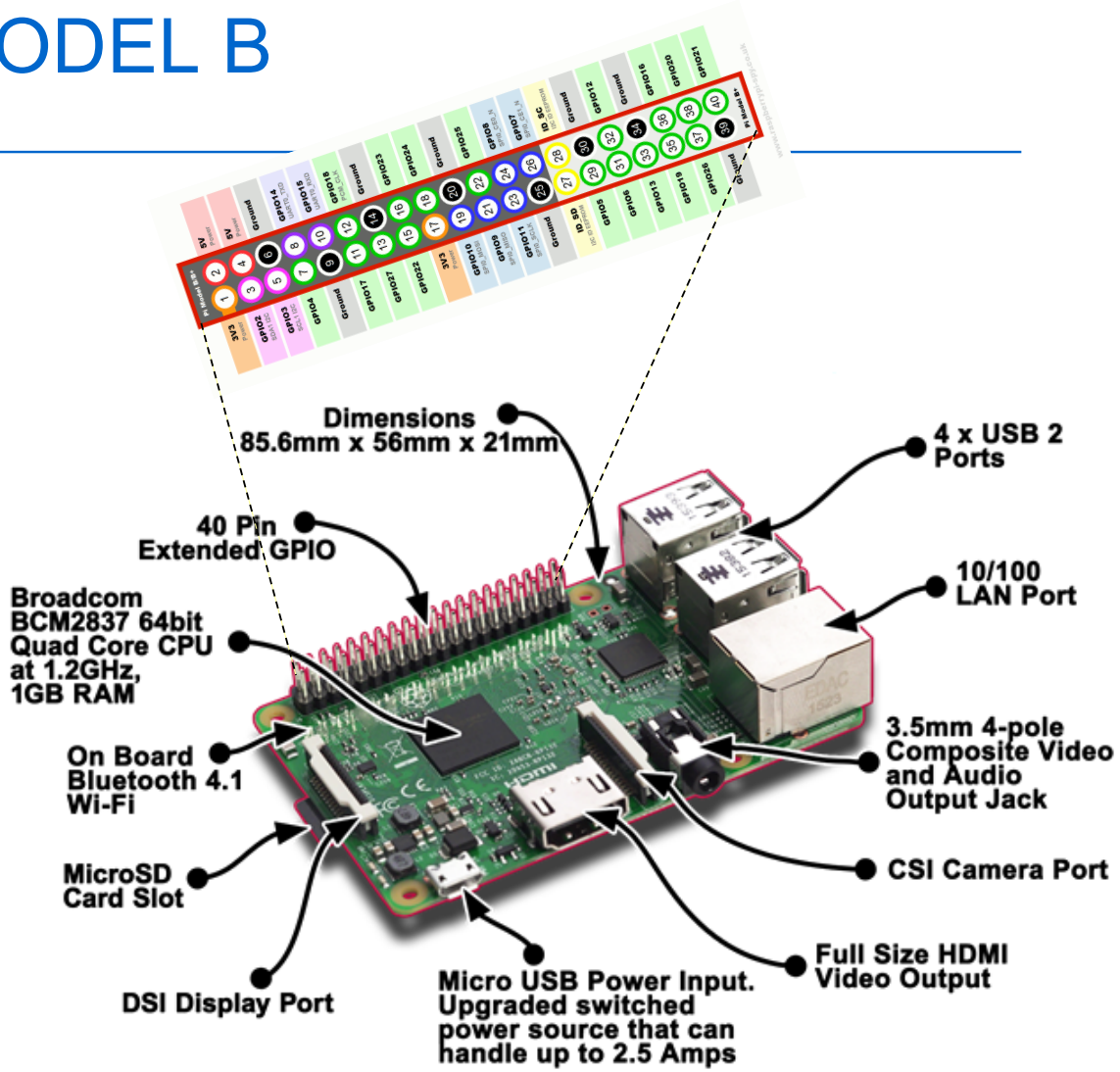
Model B Timeline



RASPBERRY PI 3 MODEL B

Specifications (Specs):

SoC (System on Chip): Broadcom BCM2837
CPU: 4xcore, ARM Cortex-A53, 1.2GHz, 64-bit
GPU: Broadcom VideoCore IV
RAM: 1GB LPDDR2 (900 MHz)
Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
Storage: microSD
GPIO (General-Purpose Input/Output)**:** 40-pin header (only 26 for Pi A)
Ports: HDMI, 3.5mm analog audio-video jack, 4x USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)



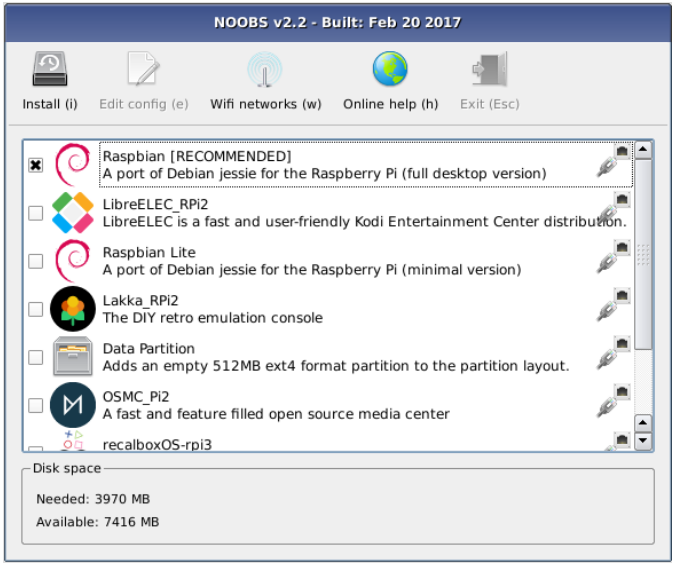
https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks

Setup of the Raspberry Pi

- Step 1: Setup an interface to the device
  - Plug in a monitor (via HDMI)
  - Keyboard/mouse via USB
- Step 2: Get an Operating System (OS)
  - Raspberry Pi needs an OS
  - OS image must be present on the micro SD card
- Step 3: Power supply
  - Micro USB power supply (at least 2A at 5V)

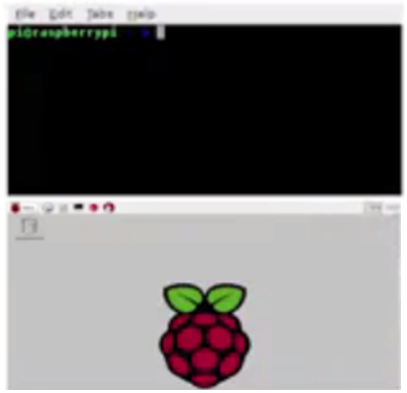
Installing an OS

- Use NOOBS (New Out-Of-Box-Software)
  - Comes pre-installed on Micro SD bundled with Raspberry Pi boards
  - Otherwise use a “good quality” 8GB+ micro SD and do:
    - Format the micro SD (need an SD reader)
    - Download NOOBS for free from www.raspberrypi.org/downloads
    - Extract NOOBS download
    - Put it on the micro SD
- NOOBS will install an OS on the SD card
  - You get a choice of OS
    - Longer list if you are connected to Internet
  - Choose RASPBIAN (distribution of Linux/Debian)



Configuration of Raspberry Pi

- **Raspi-Config**
  - is a tool, which provides various setup/boot options for Raspberry Pi
  - will run automatically when you boot with a new SD card for the first time
- Raspi-Config key Options
  - Expand Filesystem: reformats your micro SD card filesystem to allow access to all the memory
  - Change User Password (highly important!)
    - Raspberry Pi starts with one default user account
      - Username: pi
      - Password: raspberry
  - Change Boot options
    - Console (text-based interface, default)
    - Desktop graphic interface



Programming Raspberry Pi

- Many programming languages can be used
  - Need a compiler (C, C++, Java, etc) or an interpreter (Python, Perl, etc)
  - Python is most convenient
    - Good programming environment built-in
    - Good APIs available to access Raspberry Pi hardware
- Python language
  - High-level language, easy to use
    - No need to explicitly declare data types
    - No pointers
    - Object-oriented programming
  - Slow compared to C/C++ (interpreted not compiled)
  - Two versions: 2.x and 3.x (3.x recommended)

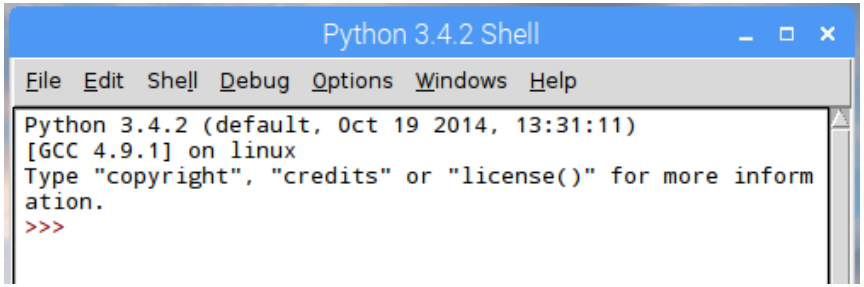
Python Programming Environmment

- Two possible environments
  - Integrated Development Environment (IDE)
    - IDLE is the best option
    - Invoke via ***Menu > Programming > Python***
    - Select Python 3
  - Text editor and interpreter separately
    - Use Raspberry Pi text editor (e.g., Pico or Nano) to write a program „test.py“
    - Execute program by typing „python3 test.py“

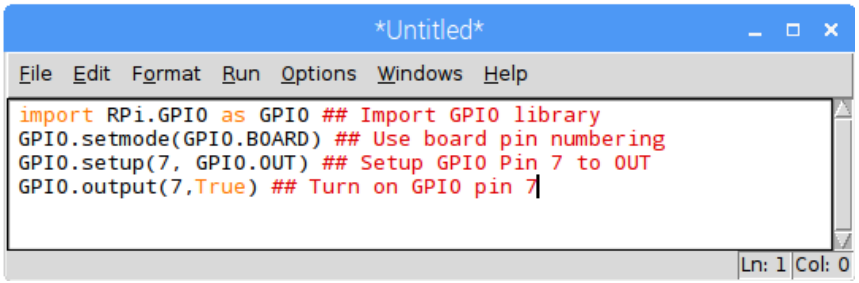
Executing Python Code

Two ways to do it:

- Interactive
  - Execute lines typed interactively in a Python console/shell
  - Start IDLE, shell is default
  - In terminal type „python3“

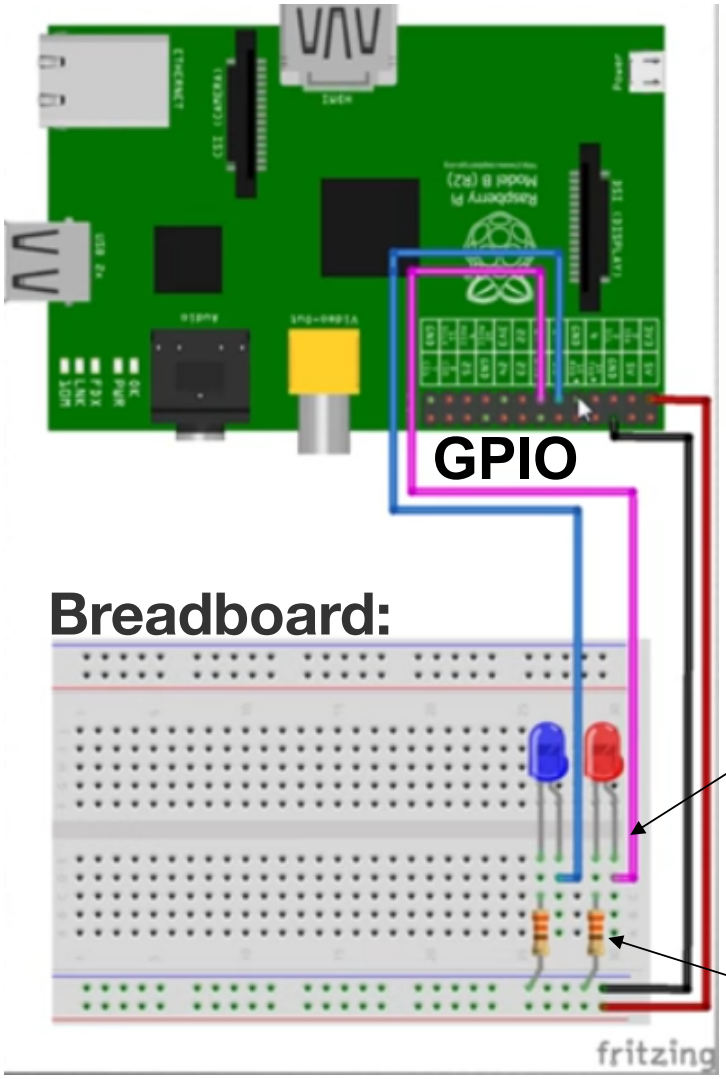


- Batch
  - Execute an entire Python program
  - Start IDLE
  - ***File > New File*** to create a new text editor window
  - Type in code
  - ***Run > Run Module***
  - Python shell will open and code will execute

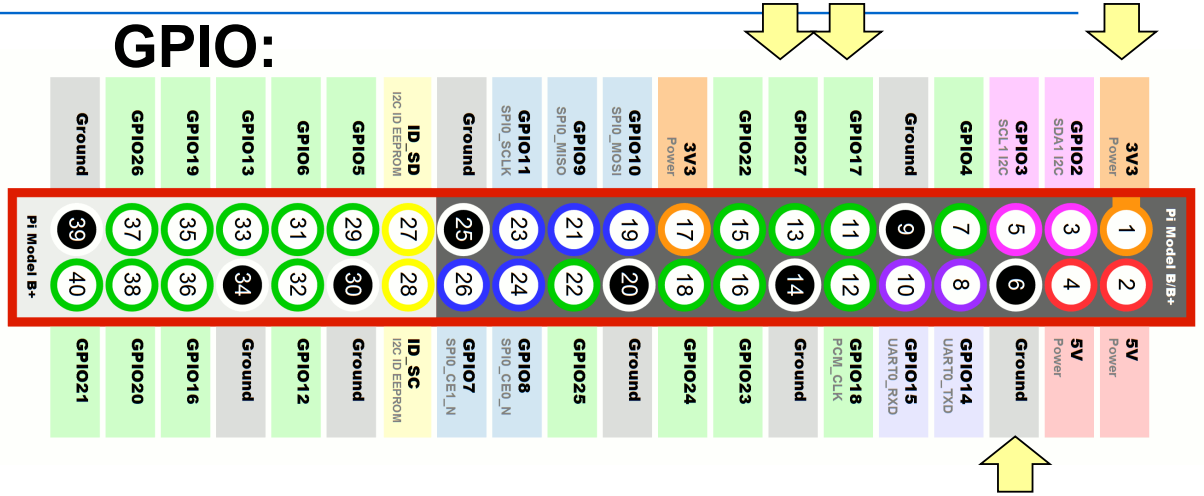


Setup for Optional LED Lab (1)

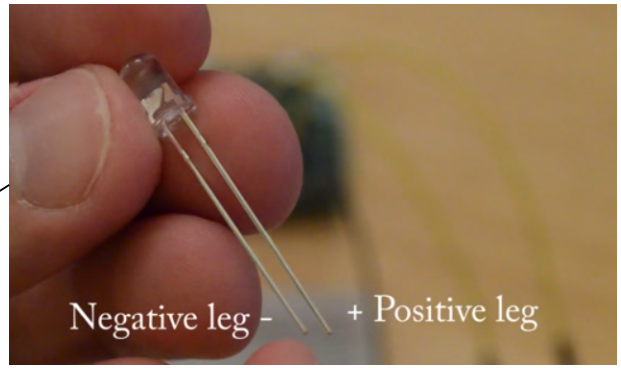
Command: gpio readall



GPIO:

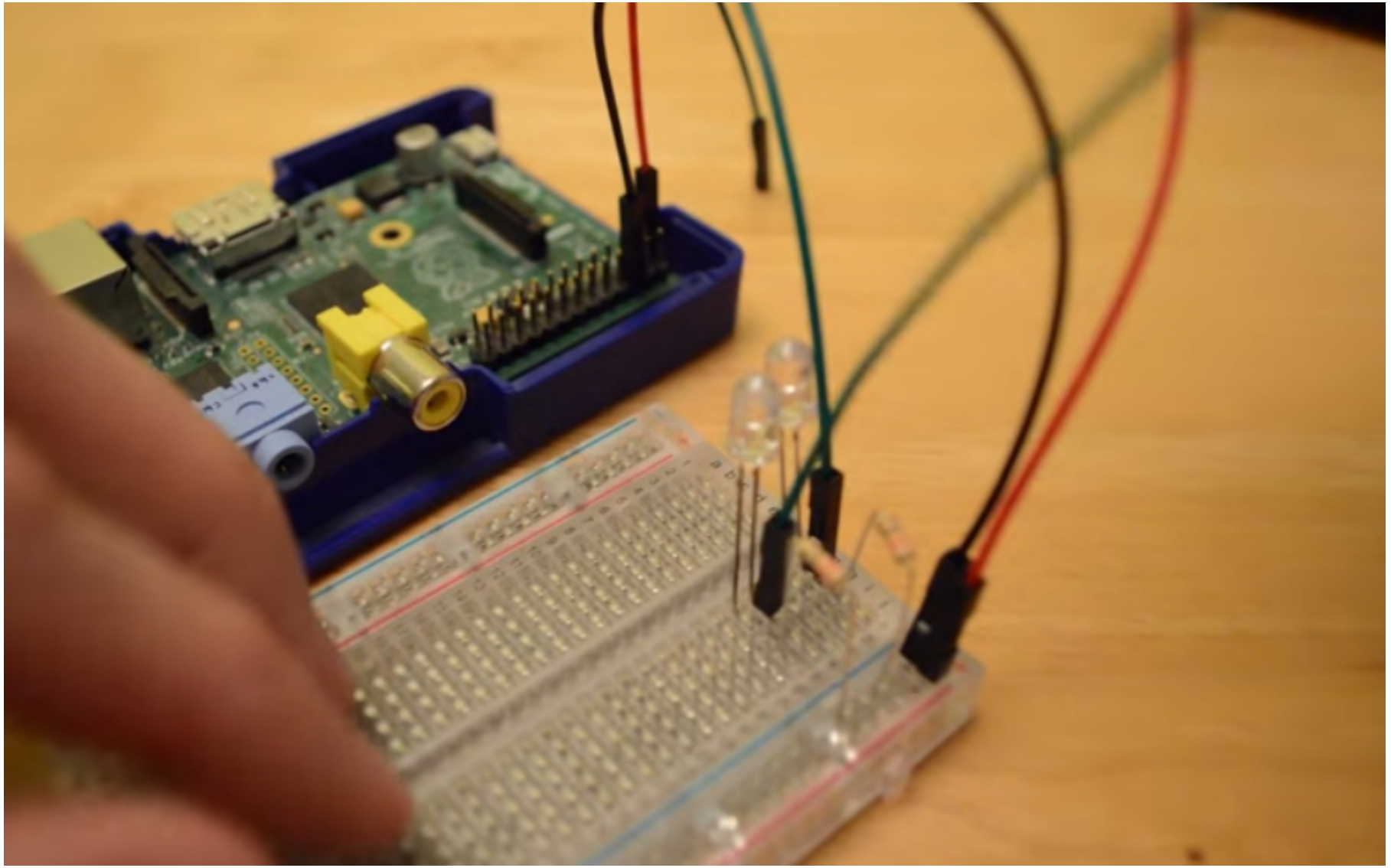


LED:



Resistor: Place the LED with a 270 ohm resistor in series with the GND and GPIO pin of Raspberry Pi (Always use a resistor with LEDs to limit the current, else you might end up with a burnt LED!)

Lab Setup (2)



LED on/off

File in Moodle: blink.py

LED-on-off.py - /home/pi/Desktop/LED-on-off.py (3.4.2)

File Edit Format Run Options Windows Help

```
#!/usr/bin/python
# import libraries
import time # import time to use sleep()
import RPi.GPIO as GPIO # import our GPIO library
# setup GPIO options
GPIO.setmode(GPIO.BCM) # set the board numbering system to BCM
GPIO.setwarning(False) # turn off warnings as they are annoying
# setup our output pins
GPIO.setup(17,GPIO.OUT)
GPIO.setup(27,GPIO.OUT)
# Turn LEDs on
print "lights on"
GPIO.output(17,GPIO.HIGH)
GPIO.output(27,GPIO.HIGH)
time.sleep(1) # sleep for 1 second
# Turn LEDs off
print "lights off"
GPIO.output(17,GPIO.LOW)
GPIO.output(27,GPIO.LOW)
# reset all the configurations made in this script.
GPIO.cleanup()
```

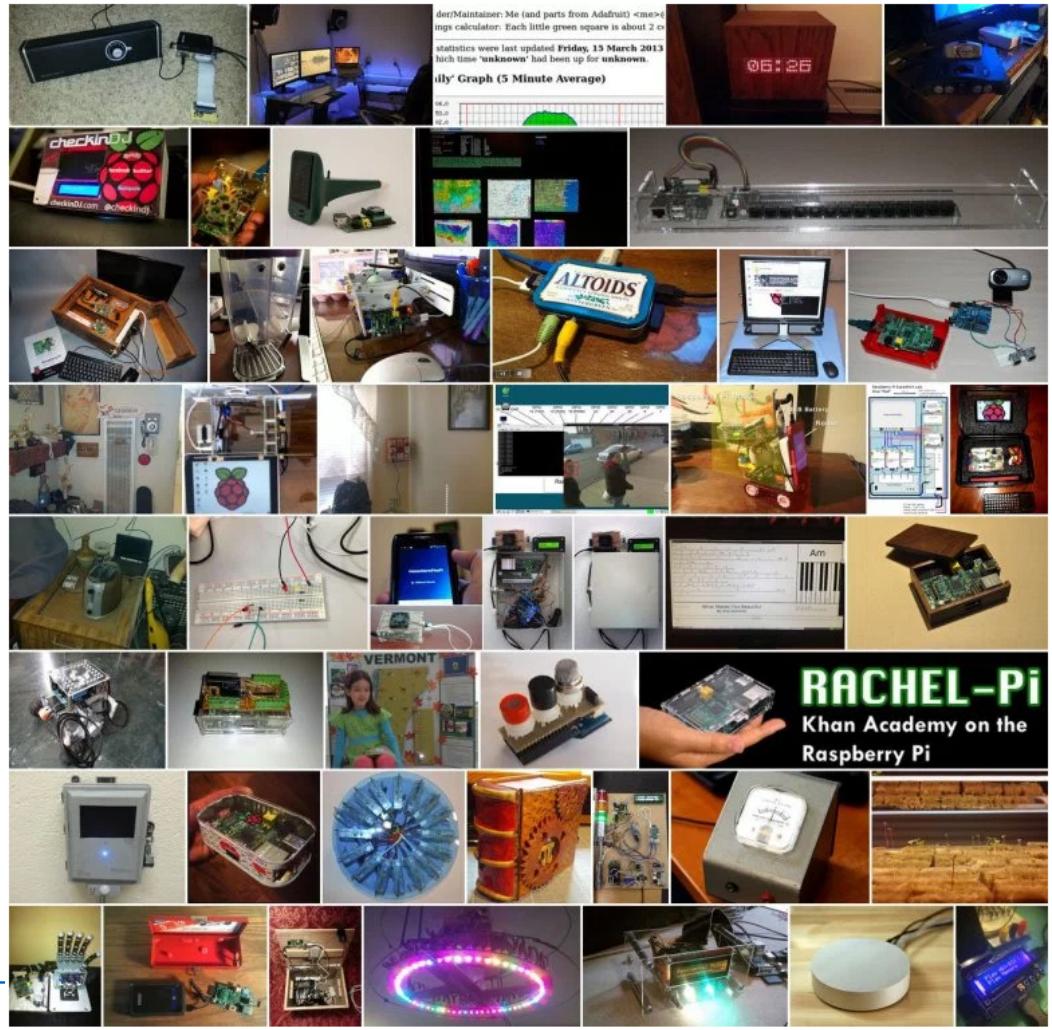
Ln: 16 Col: 15

Is Raspberry Pi an IoT Device?

- Maybe – Depends on how it is used!
- Similarities
  - Network connectivity and computational intelligence
  - Small and cheap (relative to a PC)
  - Can interface directly with sensors/actuators via pins
- Differences
  - Interface can be exactly the same as a PC running Linux
    - Complexities of the system can be visible

Raspberry PI - Samples of IoT Projects

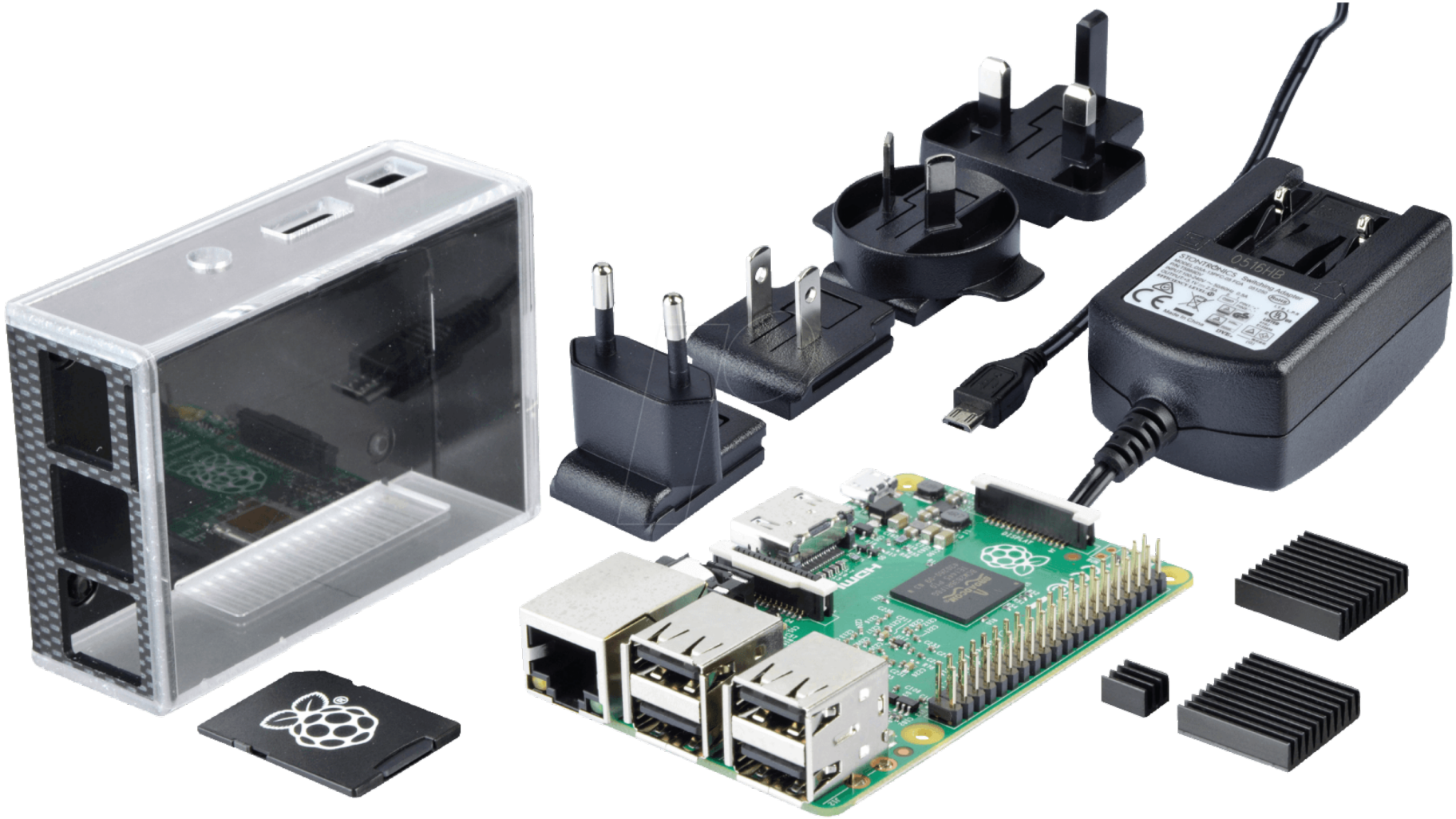
http://makezine.com/2013/04/14/47-raspberry-pi-projects-to-inspire-your-next-build/



http://drstrangelove.net/2013/12/raspberry-pi-power-cat-feeder-updates/

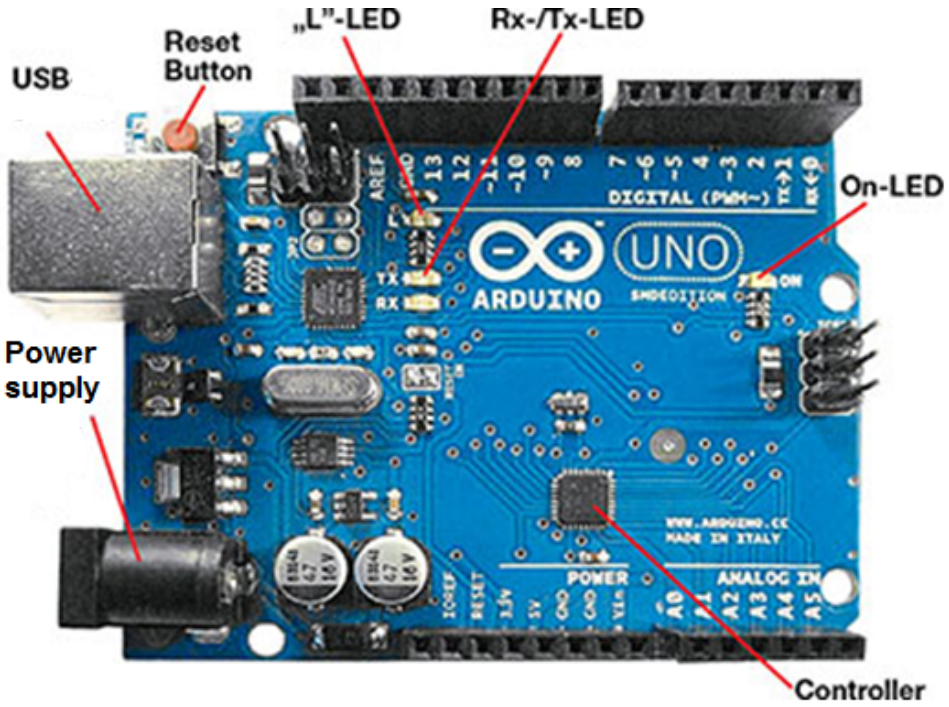
Lab Hardware: Raspberry Pi

Raspberry PI 3 B All-In-Bundle: **58,78 €**



ARDUINO

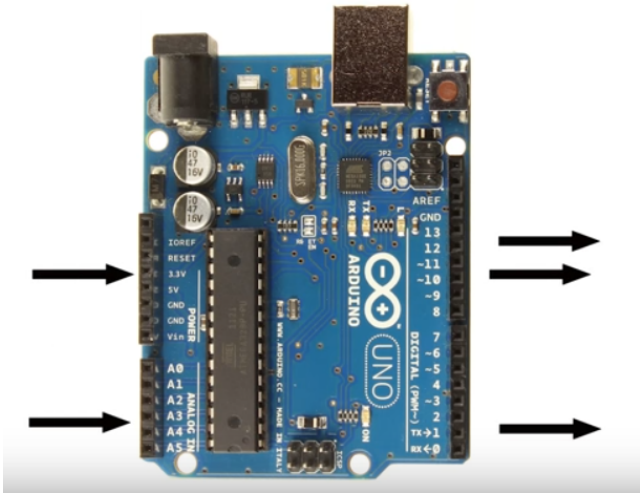
Arduino UNO



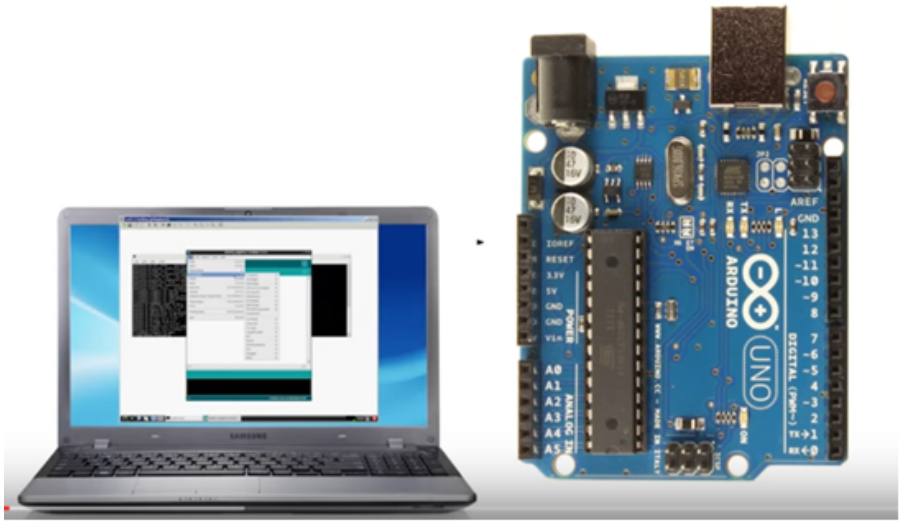
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14
PWM Digital I/O Pins	6 (out of 14)
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Programming Arduino (1)

- Is designed for turning electronic inputs to outputs
  - Rapidly & Cheaply!



- Writing programs (called *sketches* is done on a separate machine
  - Uploaded to the Arduino for execution

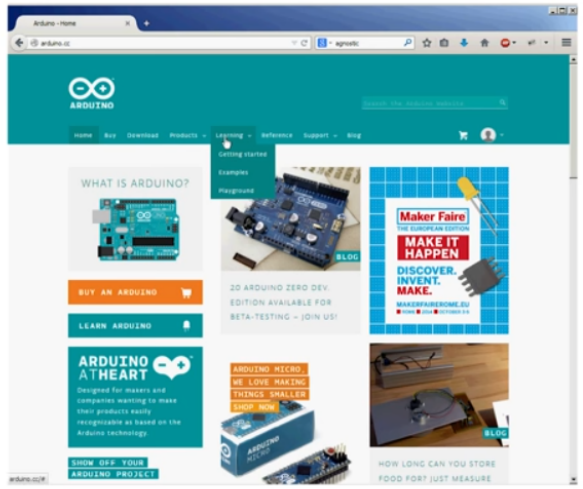


Arduino is based on a microcontroller

- Microcontroller vs microprocessor
  - Microprocessor = CPU
  - Microcontroller = CPU, RAM, ROM + some peripherals on 1 chip
- How do you run code without an operating system

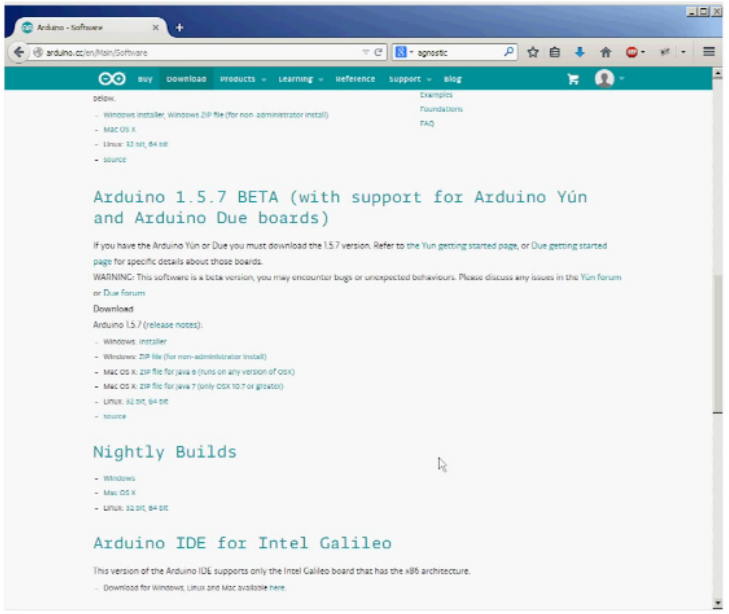
Programming Arduino (2)

- www.arduino.cc

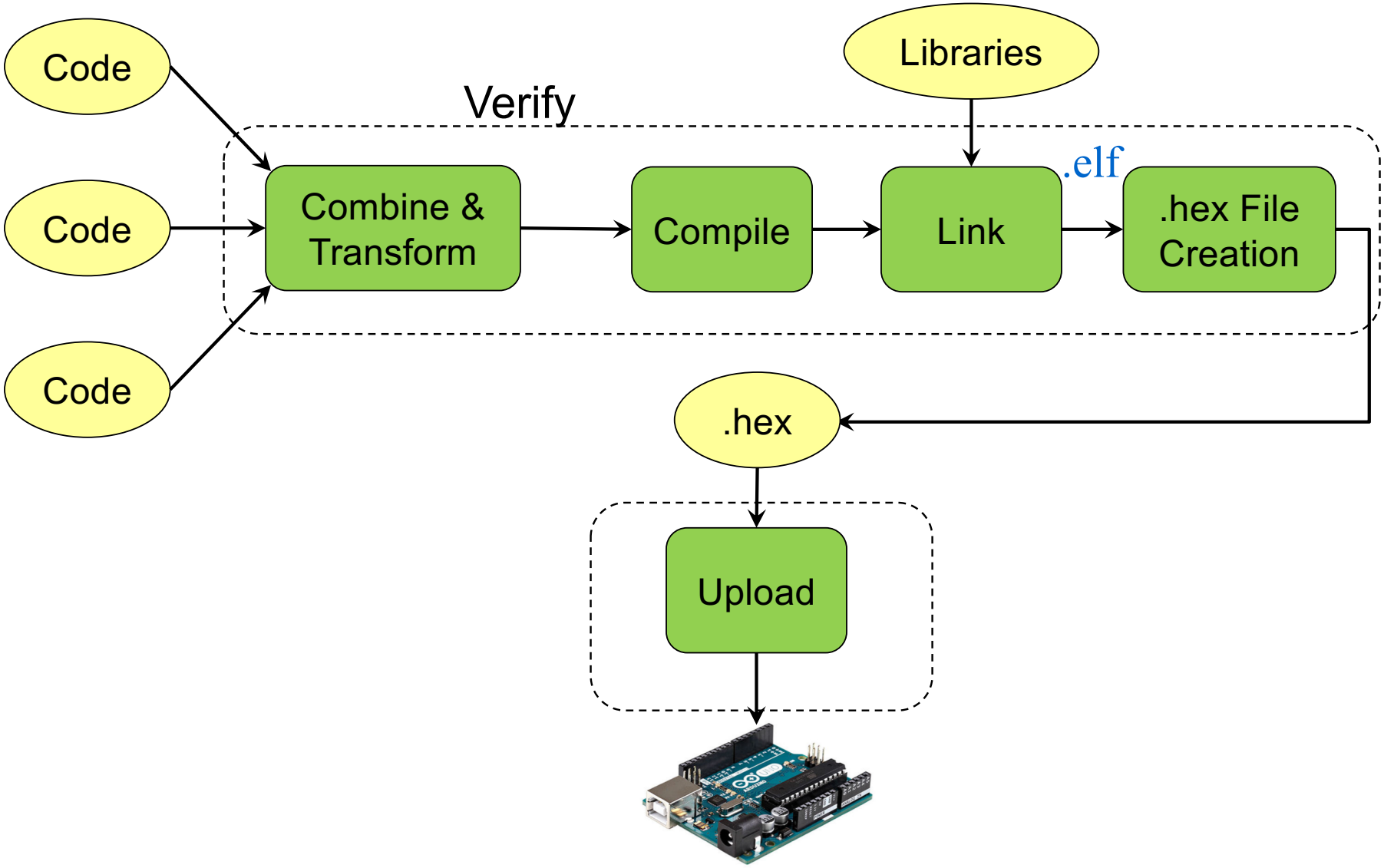


www.arduino.cc
IDE Integrated Development Environment
Home of the Arduino project
open source

- IDE
  - Platform agnostic – works on
    - Desktop
    - Laptop
    - Tablets
    - Mobile
  - OS agnostic



Verify and Upload



Verify (1)

Combine & Transform

- All program files are combined into one
- An #include is added to reference basic Arduino libraries
- Function prototypes are added
- A main() function is created

Compile & Link

- avr-gcc is invoked to cross-compile the code
  - Resulting code executes on AVR
- Generates an object file (.o)
- Object file is linked to Arduino library functions
- Result is an .elf file

Verify (2)

Hex File Creation & Programming

- avr-objcopy is invoked to change the format of the executable file
- A .hex file is generated from the .elf file

Arduino Programs

- A program is called sketch
- C++ program using Arduino library functions (C++ is a superset of C)
- Classes defined in libraries
  - Ethernet.begin(mac);
  - Serial.begin(speed);
  - Client.print(“Hello“);
  - Serial.print(“Hello“);

Sketch Structure

Setup() Function

- A sketch does not have a main() function
- Every sketch has a setup() function
  - Executed once Arduino is powered up
  - Used for initialization operations
  - No argument / no return value

```
Void setup(){
        …
}
```

Loop() Function

- Every sketch has a loop() function
  - Executed iteratively as long as Arduino is powered up
  - Loop() starts executing after setup() has finished
  - Loop() is the main program control flow
  - No argument / no return value

```
Void loop(){
        …
}
```

Input/Output (I/O): Functions to Access Pins

Pin Mode

Void pinMode(pin, mode)

- Sets a pin to act as either I/O
- pin is the pin number
  - 0-13 for digital pins
  - A0-A5 for analog pins
- mode is the I/O mode the pin is set to
  - INPUT
  - OUTPUT
  - INPUT_PULLUP: acts as INPUT with reversed polarity

Digital I/O

Int digitalRead(pin)

- Returns state of an input pin
- Returns either LOW (0 volt) or HIGH (5 volts)

Void digitalWrite(pin, value)

- Assigns the state of an output pin
- Assigns either LOW or HIGH

Analog Input

Int analogRead(pin)

- Returns state of an analog input pin
- Returns an integer 0 .. 1023
- 0 (o volt), 1023 (5 volts)

Blink Sketch

sketch_iot_1 | Arduino 2:1.0.5+dfsg2-4

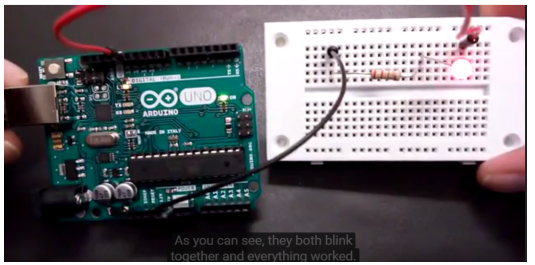
File Edit Sketch Tools Help

sketch_iot_1

```
// SIMPLE PROGRAM

int ledPin = 13;                  // LED connected to digital pin 13

void setup()
{
  pinMode(ledPin, OUTPUT);        // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH);     // sets the LED on
  delay(1000);                    // waits for a second
  digitalWrite(ledPin, LOW);      // sets the LED off
  delay(1000);                    // waits for a second
}
```

1 Arduino Uno on /dev/ttyACM0

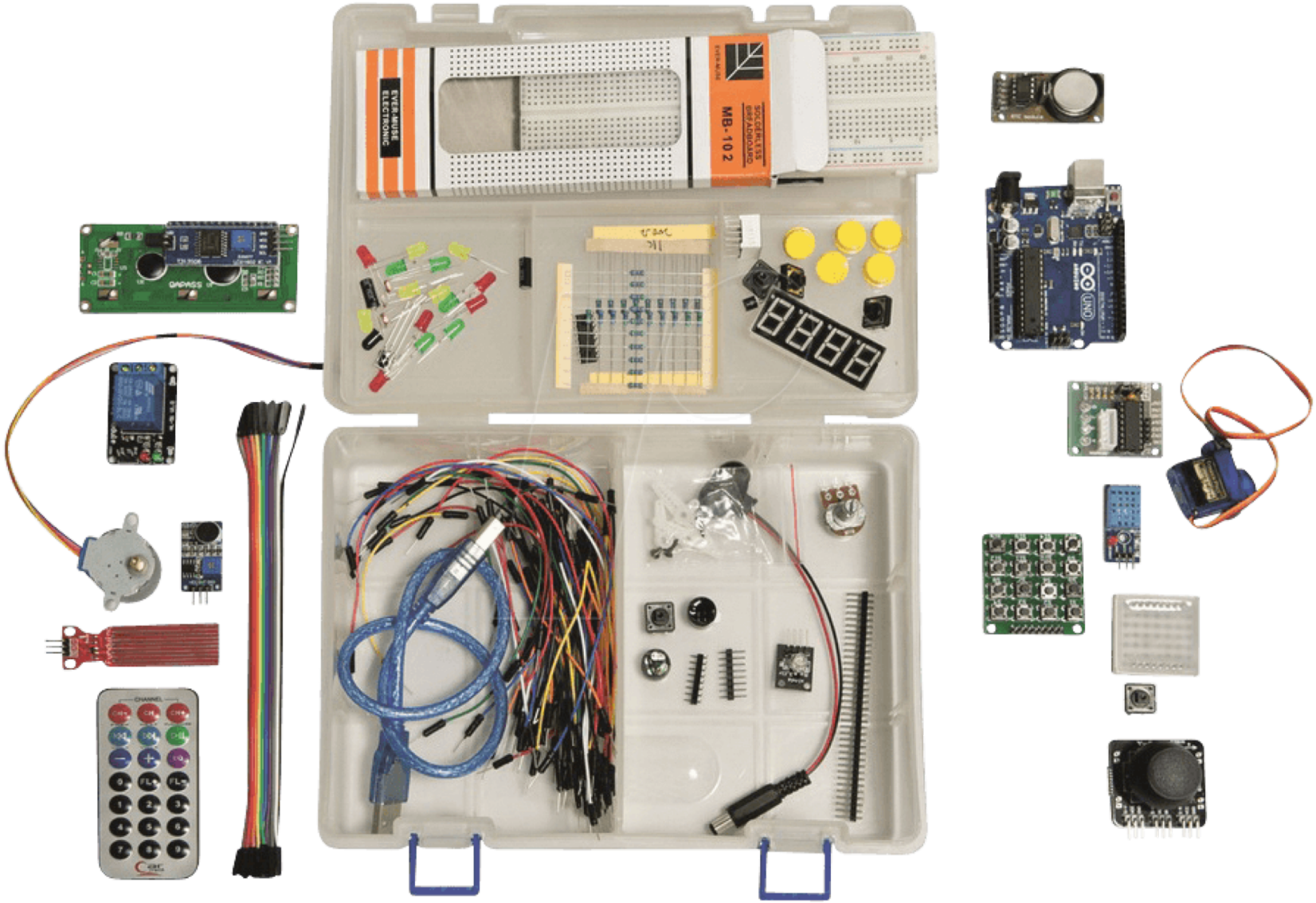
Wired LED



Built-in LED

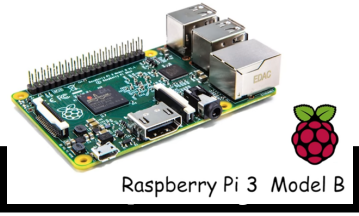
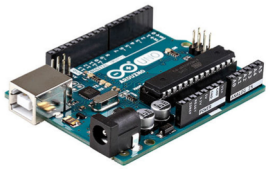
Lab Hardware: Arduino

Allnet 4duino Starter-Kit **41,98 €**



Raspberry Pi vs. ARDUINO

Raspberry Pi vs. Arduino

			
Processor	1200 MHz	16 MHz	PI is faster Larger address space
	64 Bit	8 Bit	
Memory	1024 MB SRAM	2 KB SRAM	Pi has more memory
	4 GB Flash	32 KB Flash	
	-	1 EEPROM	
OS	Full fledged OS	-	
	Processes	-	
IO	3.3 V voltage level	5 V	PI higher energy efficiency
	Ethernet	-	
	SD Card	-	
	-	Analog input	
	Accessing to pins may be time-consuming	Accurate time for writing to pins	Arduino better supports time-sensitive applications

Chapter 6:
Connectivity for IoT

Enabling Communication Technologies for IoT

Learning Objectives

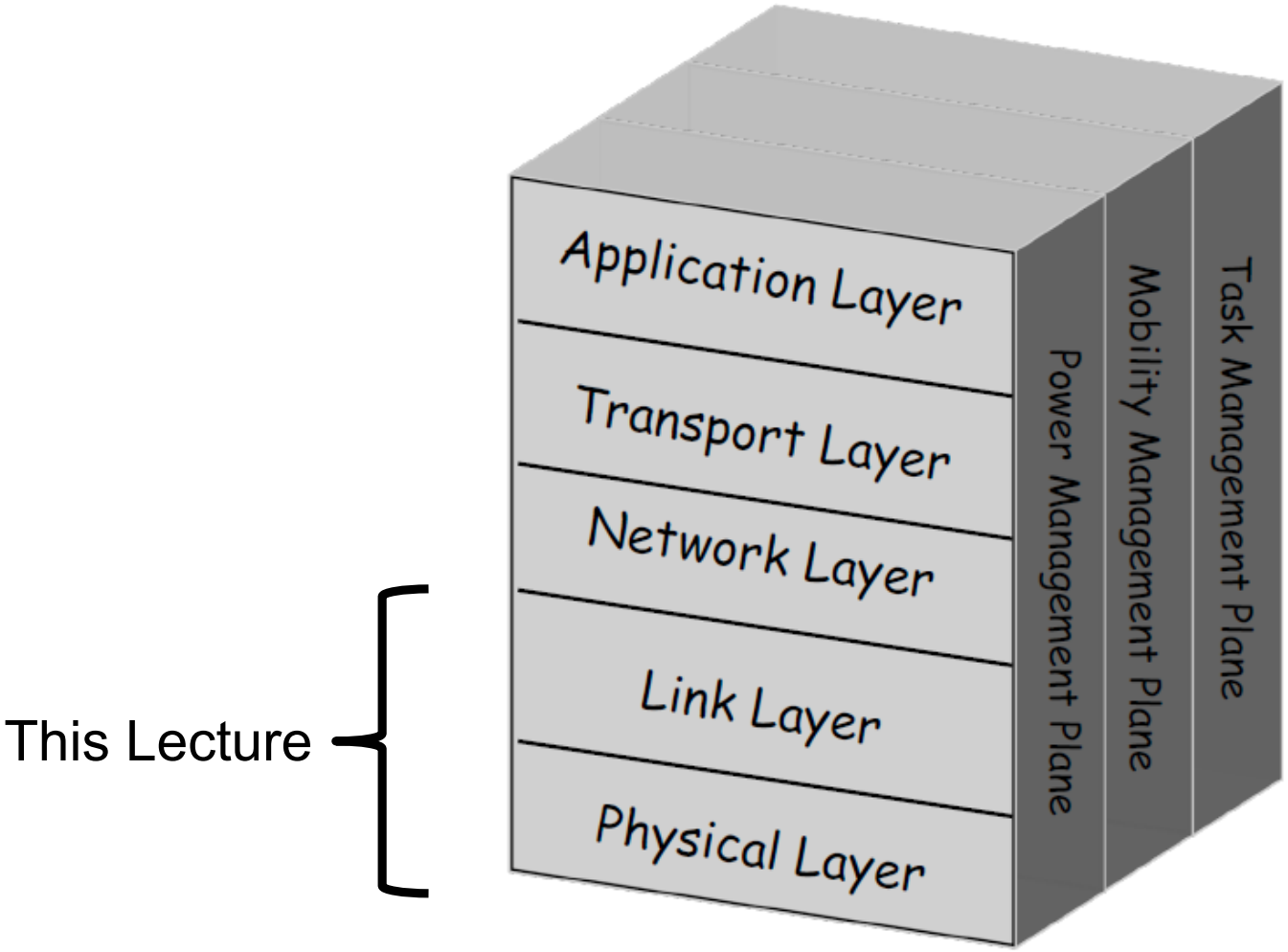
Upon completion of this chapter, the student should understand:

- The main attributes of the major competing wireless technologies
- How to interpret Physical/link layer specifications
- How to compare two radio modules
- The common impairments affecting radio performance
- How to select a suitable radio technology

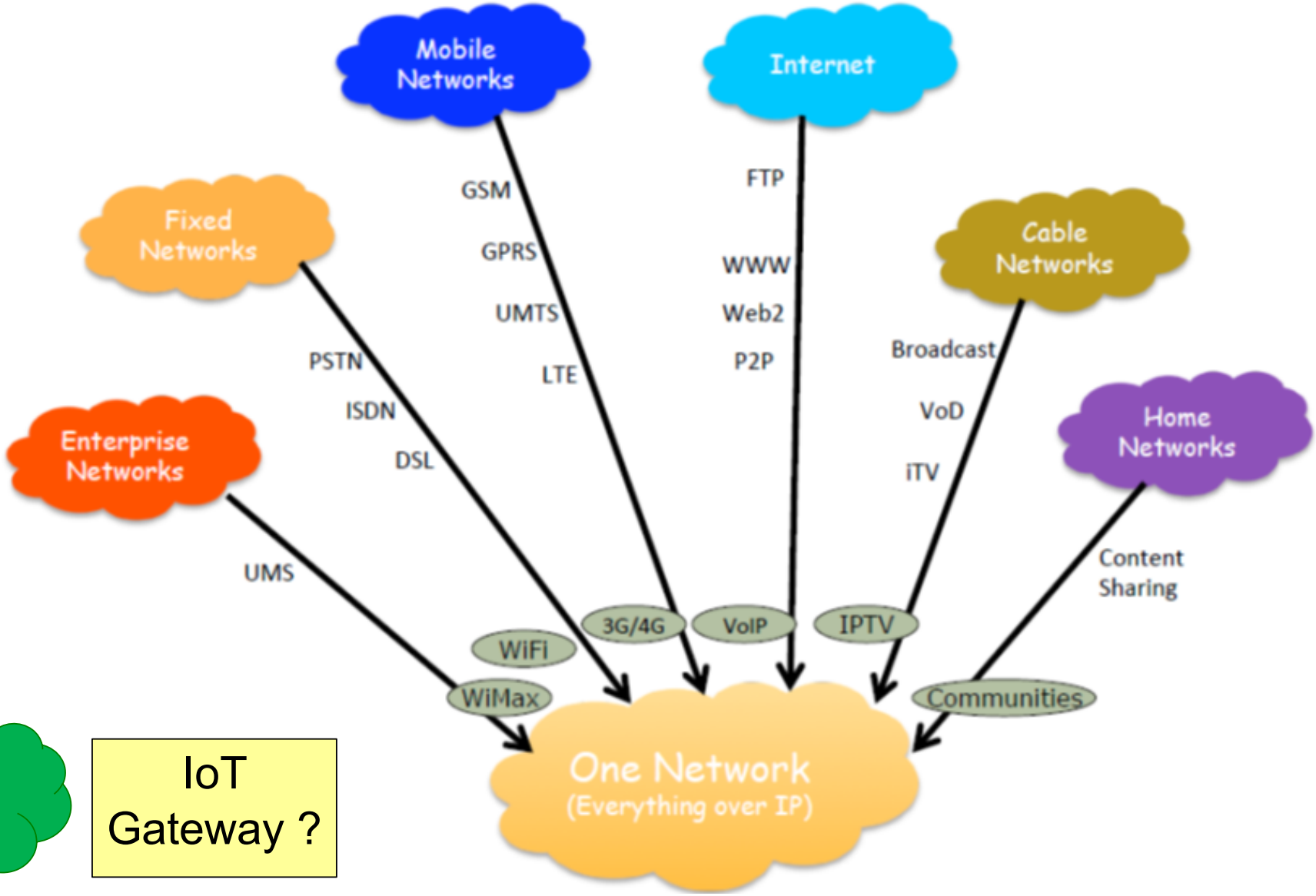
Outline

- Motivation
- Wireless Behaviour
- The Main Wireless Standards
  - Cellular, WiFi, ZigBee, Bluetooth, many others
- Selection of Suitable Technology for a given Application

The Protocol Stack



The Internet ..



IoT ?

IoT Gateway ?

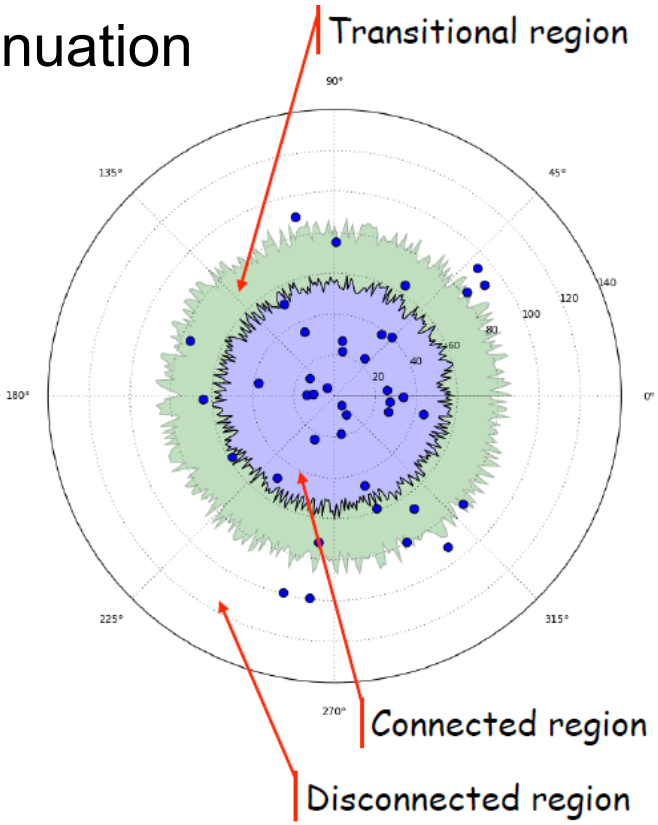
Frequency Allocation (Spectrum)

- Some frequencies are allocated to specific uses
  - Cellular phones, analog television/radio broadcasting, DVB-T, radar, emergency services, radio astronomy, ..
- Particularly interesting: ISM bands ("Industrial, scientific, medicine") – license-free operation

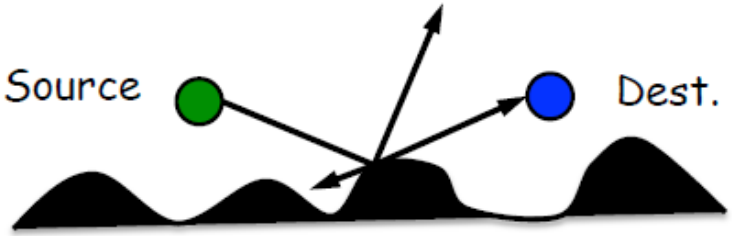
Some typical ISM bands	
Frequency	Comment
13,553-13,567 MHz	
26,957 - 27,283 MHz	
40,66 - 40,70 MHz	
433 - 464 MHz	Europe
900 - 928 MHz	Americas
2,4 - 2,5 GHz	WLAN/WPAN
5,725 - 5,875 GHz	WLAN
24 - 24,25 GHz	

Challenges on Physical Layer

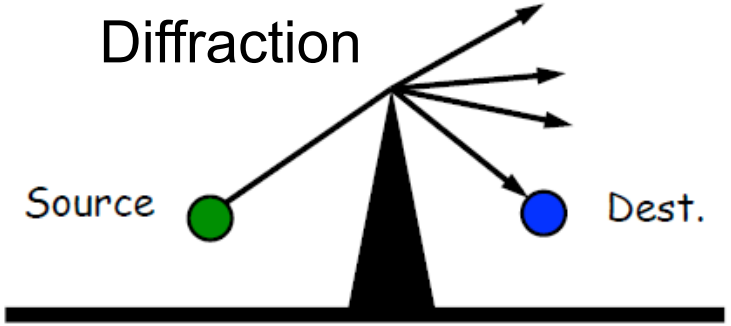
Attenuation



Scattering

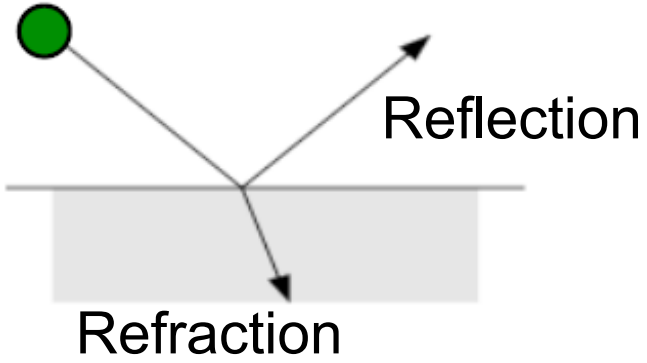


Diffraction

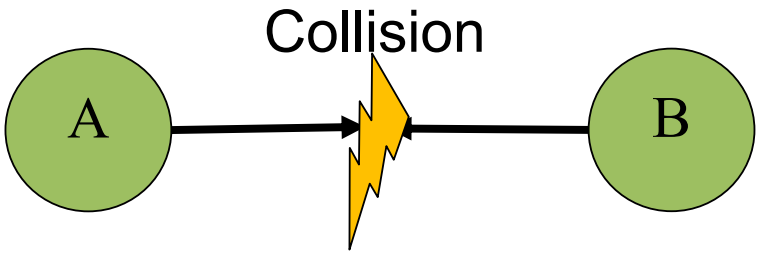


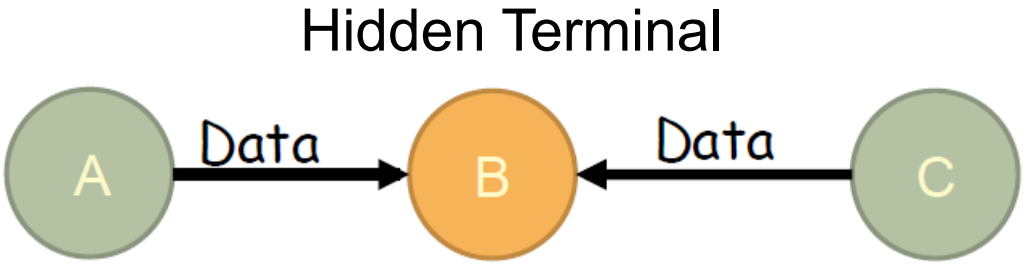
Line of Sight

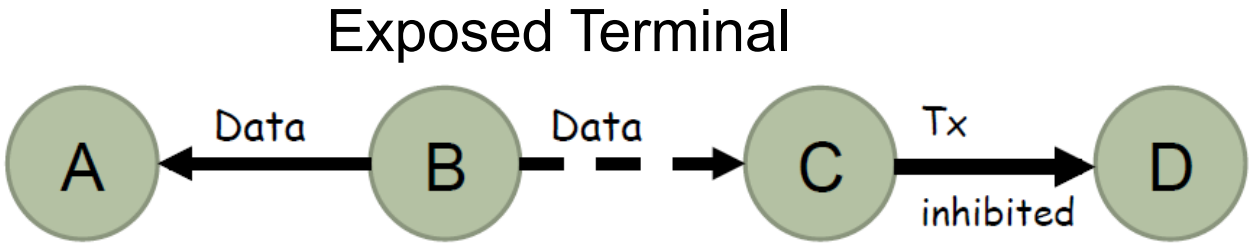
- Line of Sight
- Near Line of Sight
- No Line of Sight



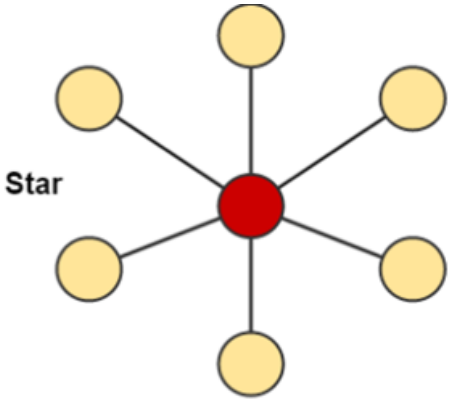
Challenges on Link Layer



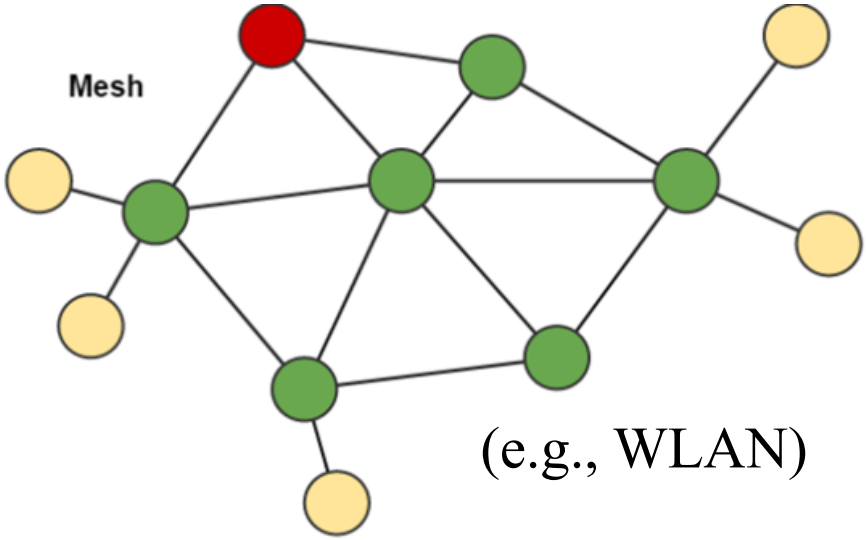




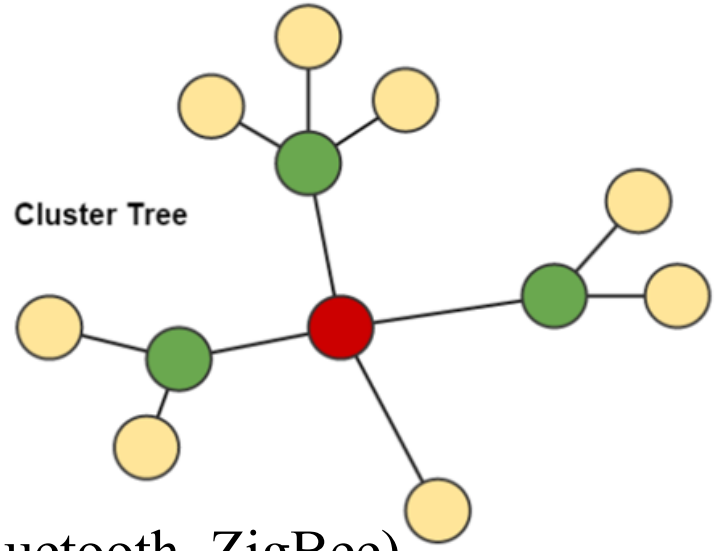
Topologies for Wireless Networks



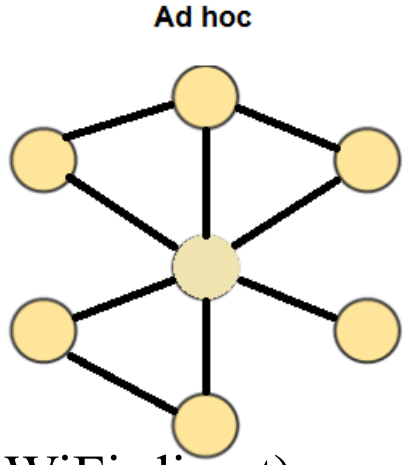
(e.g., cellular)



(e.g., WLAN)



(e.g., Bluetooth, ZigBee)



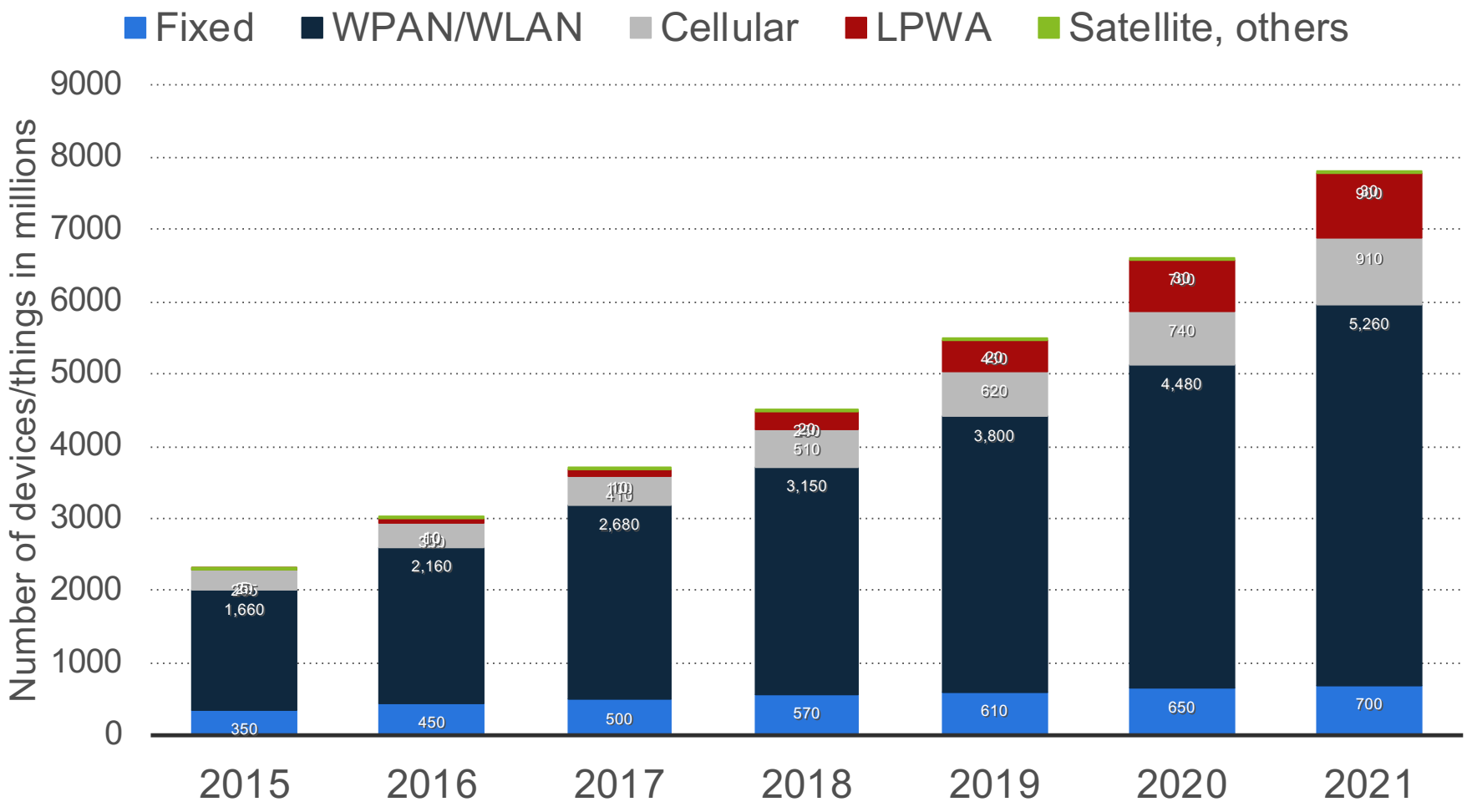
(e.g., WiFi-direct)

Coordinator Routers End Devices

Characteristics of wireless networks

- **Number of nodes:** number of all devices in the network, like routers, gateways, or hosts. The larger the number of nodes in a network, the more difficult it is to manage the network.
- **Mobility:** This key refers to mobile nodes in the network, for example mobile routers and mobile clients. A network with a higher degree of mobility usually exposes a higher dynamic topology.
- **Hop-Count:** number of hops between a source and destination. A high hop count is likely to increase the latency of transmissions and decrease the throughput of a network.
- **Self-Organization:** degree of human interaction required by a network, e.g., for configuration and management. Thus a network with a higher degree of self-organization is a network which demands less human interaction.
- **Energy-Awareness:** energy sensitivity of a network. A network has to be more energy-aware if the energy resource is finite.
- **Universality:** Characterizes whether the network is tailored to a specific application. A network is more universal if it can be used for more applications.
- **Data rate:** user-perceived throughput, for example the quality of a connection from a source to a destination. Usually, the higher the data rate, the better the connection throughput. However, this key has to be used carefully, since a wireless link may show low quality due to interference even with high data rates.

Enabling Technologies: Number of connected devices/things by technology worldwide



WPAN: Wireless Personal Area Network
WLAN: Wireless Local Area Network
LPWA(N): Low Power Wide Area (Network)

Source: Berg Insight; ID 626323

Connectivity Technology Considerations

Technical

Coverage determines where the devices can be deployed and connected

Energy efficiency affects battery life and maintenance cycle

Data rate limits the types of services that can be provided

Other technical features may be relevant for specific applications

Commercial

QoS ensures the value the IoT service can deliver

Security protects privacy and integrity of IoT users

Cost decides the business viability of implementing and operating the IoT service

Scalability determines the flexibility for managing growth

Ecosystem

Future proofness ensures the strategic investment in IoT is economically and technologically sustainable in the long run

Global reach and interoperability brings simplicity and efficiency to international IoT deployments

Src Telenor Connexion, 2016

Main Technologies for IoT

Considerations	Traditional Cellular			Cellular LPWA			Proprietary LPWA			Short Range		
	2G	3G	4G	LTE-CatM1	EC-GSM	NB-IoT	SigFox	LoRa	Ingenu	Wi-Fi low power	ZigBee 3.0	Bluetooth LE
Outdoor coverage	>10km	>10km	>10km	>10km	>15km	>15km	>15km	>10km	>15km	<1km	<300m	<100m
Indoor coverage	High	Medium	Medium	Medium	High	High	High	High	Very low	Very high	Medium	Low
Energy efficiency	2-5 years	<10 days	<10 days	>10 years	>10 years	>10 years	10-20 years	10-20 years	10-20 years	6-12 months	6-12 months	6-12 months
Typical uplink data rate	50 kbps	1 Mbps	10 Mbps	1 Mbps	200 kbps	20 kbps	100 bps	25 kbps	50 kbps	1 Mbps	250 kbps	1 Mbps
Bidirectional communication	Yes	Yes	Yes	Yes	Yes	Yes	Limited downlink	Yes in Class A	Yes	Yes	Yes	Yes
Mobility	Very high	Very high	Very high	Very high	High	High	Very low	Low	Medium	Medium	Low	Very low
Localization	Yes	Yes	Yes	Yes	Yes	n/a	No	Limited accuracy	n/a	Yes	Yes	yes
QoS & security	Very high	Very high	Very high	Very high	High	High	Very low	Low	Low	Medium	Medium	Medium
Device cost	$5-10	$15-30	$30-50	$20-40	$5-10	$5	$1-5	$1-5	$5-10	$5-10	$5	$5
Connectivity cost	Medium	High	Very high	High	Medium	Medium	Very Low	Low	Low	Medium	Medium	Medium
Scalability	High	High	High	High	Very high	Very high	High	High	High	Low	Low	Very low
Future proofness	Medium	Medium	Very high	High	Medium	Very high	Low	High	Low	Medium	High	High
Global reach & interoperability	Very high	Very high	Very high	High	High	High	Medium	Low	Very low	Low	Medium	High

Src Telenor Connexion, 2016

Main Characteristics of Short Range Communication (ZigBee, 6LowPAN, Z-Wave, BLE, Bluetooth)

		ZigBee	6LoWPAN (Over 802.15.4)	Z-Wave	Bluetooth Low Energy	Classic Bluetooth
Physical layer	RF band (MHz)	868/915/2400		868/908 (all chips) 2400 (400 serie chip)	2400	2400
	Bit rate (kbps)	20/40/250		9.6/40 (from 200 series chip) 200 (only 400 series chip)	1000	≤721(v1.2), 3000 (v2+EDR), ≤24,000 (v3+HS)
	Modulation	BPSK/BPSK/O-QPSK		BFSK	GFSK	GFSK (v1.2), GFSK/π/4-DQPSK/8DPSK (v2+EDR), 802.11 (v3+HS)
	Spreading technique	DSSS		No	FHSS (2 MHz channel width)	FHSS(1 MHz channel width)
	Receiver sensitivity (dBm)	−85 or better(2.4 GHz band)−92 or better(868/915 MHz bands)		−101 (at 40 kbps)	≤−70(required)	
					−87 to −93 (typical)	−90(typical)
	Transmit power (dBm)	−32 to 0		−20 to 0	−20 to 10	20/4/0(Class 1/2/3)
Link layer	MAC mecha-nism	TDMA+CSMA/CA (beacon mode) and CSMA/CA (beaconless mode)		CSMA/CA	TDMA	TDMA
	Message size (bytes)	127 (maximum)		64 (max. MAC payload in 200 series chip)	8 to 47	358 (maximum)
	Error control	16-bit CRC. ACKs (optional)		8-bit checksum. ACKs (optional)	24-bit CRC. ACKs	8-bit CRC (header); 16-bit CRC and 2/3 FEC (payload). ACKs
	Latency (ms)	<5 (beaconless mode, at 250 kbps)		<39 (at 40 kbps)	<3	<100
Identifiers		16- and 64-bit MAC addresses. 16-bit NWK identifiers	16- and 64-bit MAC addresses. 128-bit IPv6 addresses	32-bit (home ID), 8-bit (node ID)	48-bit public device Bluetooth address or random address	48-bit public device Bluetooth address
Device types or roles		Coordinator, Router and End device	Edge Router, Mesh Node (mesh under), Router (route over), Host	Controller and slave	Master and slave	Master and slave
Network layer	Multi-hop solution	Mesh routing, tree routing, and source routing	RPL (other protocols are not excluded)	Source routing	Not currently supported	Scatternet (routing protocol out of the scope of the Bluetooth specifications)
	Hop limit	30/10/5 (mesh routing/tree routing/source routing)	255	4	1	Outside scope of Bluetooth specifications
Security		Integrity, confidentiality, access control (IEEE 802.15.4 security, using 128-bit AES)		128-bit AES encryption (400 series chip)	Security Modes/Levels. Pairing. Key Gener./Distribution. Confidentiality, Authentication, and Integrity	Pairing and Link Key Generation. Authentication. Confidentiality. Trust Levels, Service Levels, and Authorization. E_X algorithms
		Key management	Key management currently out of scope			
Implementation size		45–128 kB (ROM), 2.7–12 kB (RAM)	24 kB (ROM), 3.6 kB (RAM)	32–64 kB (Flash), 2–16 kB (SRAM)	~40 kB (ROM), ~2.5 kB (RAM)	~100 kB (ROM), ~30 kB (RAM)

ZigBee, 6LowPAN

		ZigBee	6LoWPAN (Over 802.15.4)
Physical layer	RF band (MHz)	868/915/2400	
	Bit rate (kbps)	20/40/250	
	Modulation	BPSK/BPSK/O-QPSK	
	Spreading technique	DSSS	
	Receiver sensitivity (dBm)	−85 or better(2.4 GHz band)−92 or better(868/915 MHz bands)	
	Transmit power (dBm)	−32 to 0	
Link layer	MAC mecha-nism	TDMA+CSMA/CA (beacon mode) and CSMA/CA (beaconless mode)	
	Message size (bytes)	127 (maximum)	
	Error control	16-bit CRC. ACKs (optional)	
	Latency (ms)	<5 (beaconless mode, at 250 kbps)	

ZigBee, 6LowPAN

Identifiers		16- and 64-bit MAC addresses. 16-bit NWK identifiers	16- and 64-bit MAC addresses. 128-bit IPv6 addresses
Device types or roles		Coordinator, Router and End device	Edge Router, Mesh Node (mesh under), Router (route over), Host
Network layer	**Multi-hop solution**	Mesh routing, tree routing, and source routing	RPL (other protocols are not excluded)
	Hop limit	30/10/5 (mesh routing/tree routing/source routing)	255
Security		Integrity, confidentiality, access control (IEEE 802.15.4 security, using 128-bit AES)	
		Key management	Key management currently out of scope
Implementation size		45–128 kB (ROM), 2.7–12 kB (RAM)	24 kB (ROM), 3.6 kB (RAM)

Z-Wave

		Z-Wave
Physical layer	RF band (MHz)	868/908 (all chips) 2400 (400 serie chip)
	Bit rate (kbps)	9.6/40 (from 200 series chip) 200 (only 400 series chip)
	Modulation	BFSK
	Spreading technique	No
	Receiver sensitivity (dBm)	−101 (at 40 kbps)
	Transmit power (dBm)	−20 to 0
Link layer	MAC mecha-nism	CSMA/CA
	Message size (bytes)	64 (max. MAC payload in 200 series chip)
	Error control	8-bit checksum. ACKs (optional)
	Latency (ms)	<39 (at 40 kbps)

Z-Wave

Identifiers		32-bit (home ID), 8-bit (node ID)
Device types or roles		Controller and slave
Network layer	**Multi-hop solution**	Source routing
	Hop limit	4
Security		128-bit AES encryption (400 series chip)
Implementation size		32–64 kB (Flash), 2–16 kB (SRAM)

Bluetooth Low Energy, Classic Bluetooth

		Bluetooth Low Energy	Classic Bluetooth
Physical layer	RF band (MHz)	2400	2400
	Bit rate (kbps)	1000	≤721(v1.2), 3000 (v2+EDR), ≤24,000 (v3+HS)
	Modulation	GFSK	GFSK (v1.2), GFSK/π/4-DQPSK/8DPSK (v2+EDR), 802.11 (v3+HS)
	Spreading technique	FHSS (2 MHz channel width)	FHSS(1 MHz channel width)
	Receiver sensitivity (dBm)	≤−70(required)	
		−87 to −93 (typical)	−90(typical)
	Transmit power (dBm)	−20 to 10	20/4/0(Class 1/2/3)
Link layer	MAC mecha-nism	TDMA	TDMA
	Message size (bytes)	8 to 47	358 (maximum)
	Error control	24-bit CRC. ACKs	8-bit CRC (header); 16-bit CRC and 2/3 FEC (payload). ACKs
	Latency (ms)	<3	<100

Bluetooth Low Energy, Classic Bluetooth

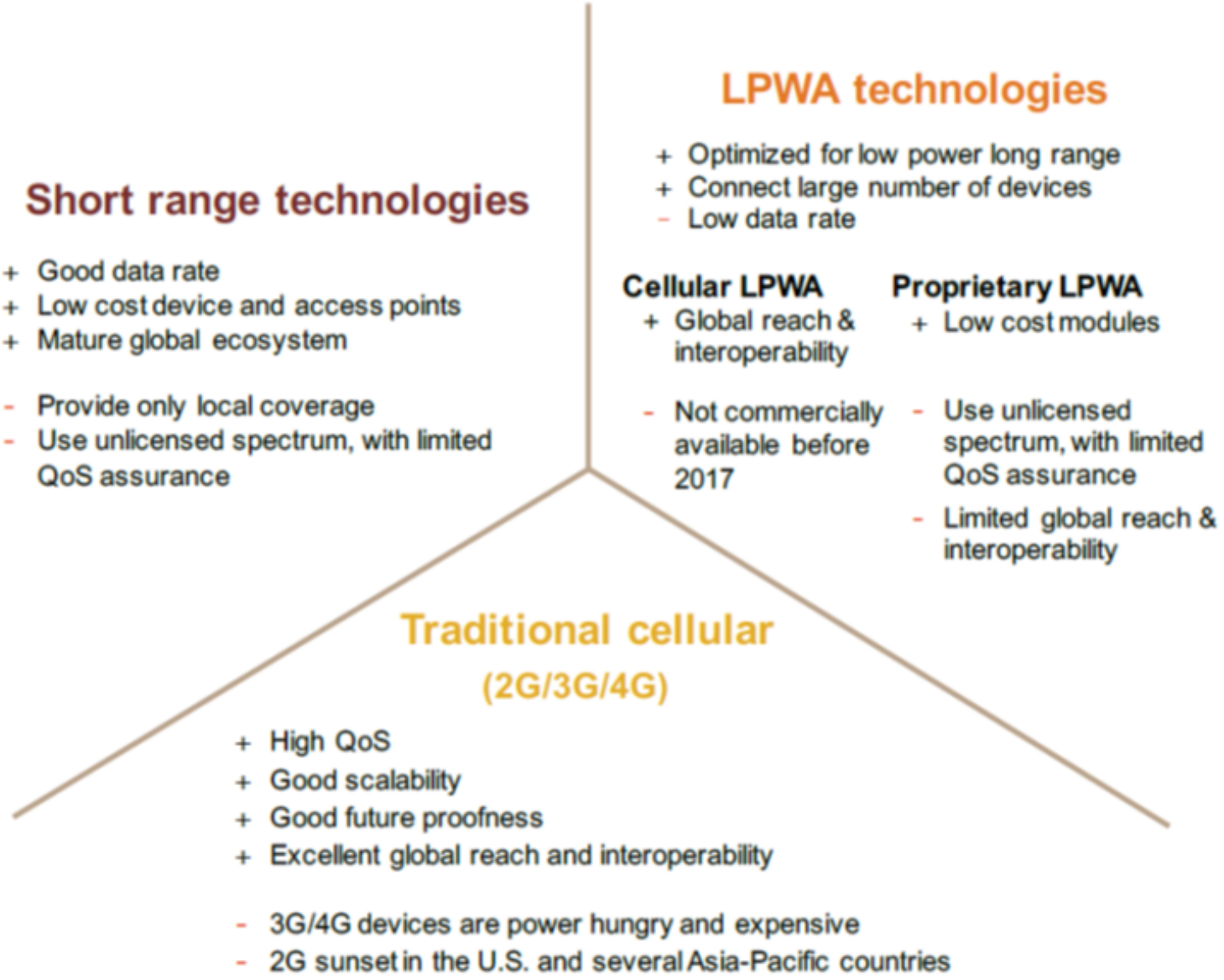
Identifiers		48-bit public device Bluetooth address or random address	48-bit public device Bluetooth address
Device types or roles		Master and slave	Master and slave
Network layer	**Multi-hop solution**	Not currently supported	Scatternet (routing protocol out of the scope of the Bluetooth specifications)
	Hop limit	1	Outside scope of Bluetooth specifications
Security		Security Modes/Levels. Pairing. Key Gener./Distribution. Confidentiality, Authentication, and Integrity	Pairing and Link Key Generation. Authentication. Confidentiality. Trust Levels, Service Levels, and Authorization. E_x algorithms
Implementation size		~40 kB (ROM), ~2.5 kB (RAM)	~100 kB (ROM), ~30 kB (RAM)

Near Field Communication (NFC)

	Near field communication
Frequency	13.56 MHz
Data rate	424 kbps
Communication range	<10 cm
Energy consumption (send, receive)	<15 mA

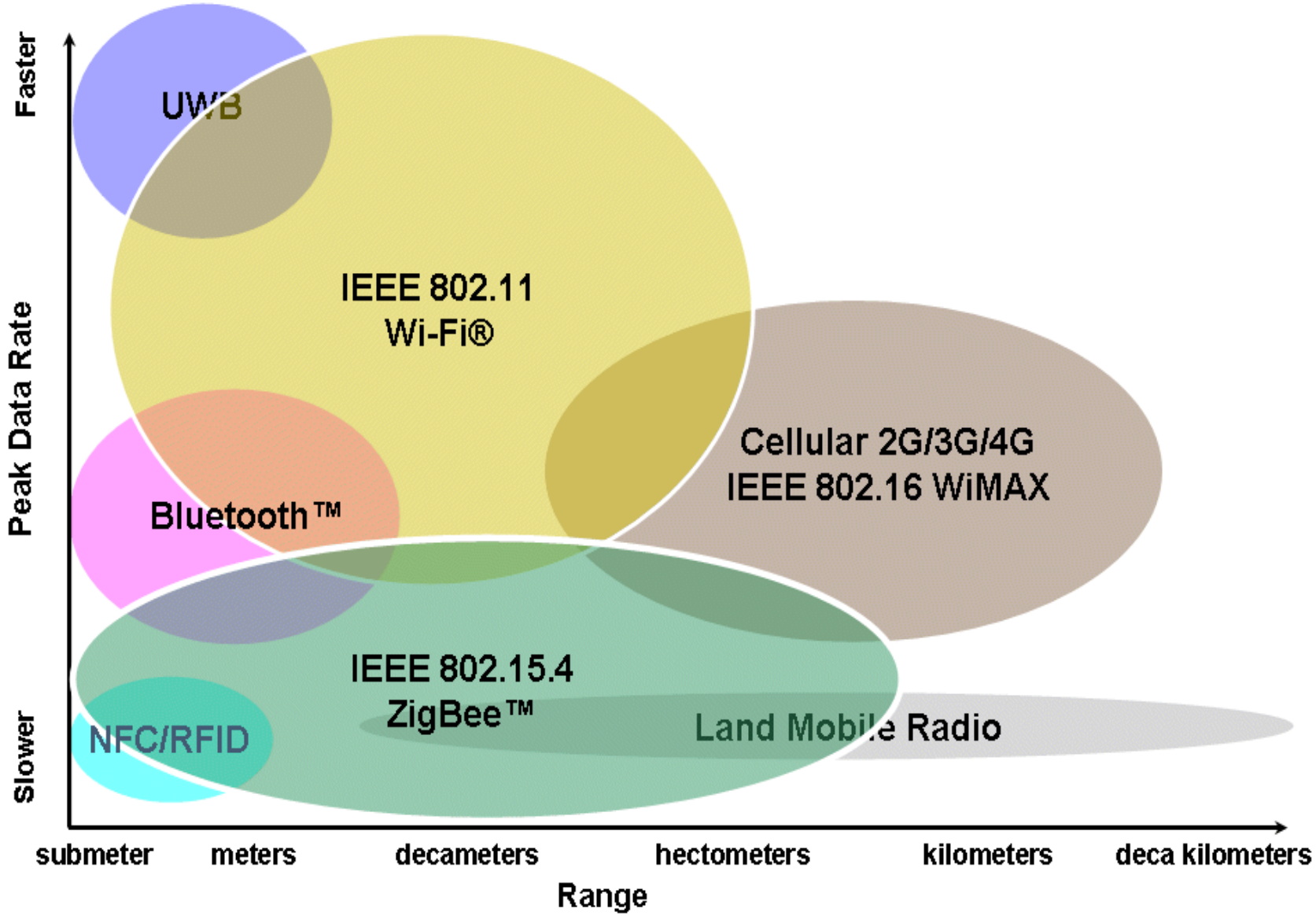
Selection of Suitable Technique for a planned IoT Application

Key Strengths and Weaknesses of Different Types of Connectivity Technologies



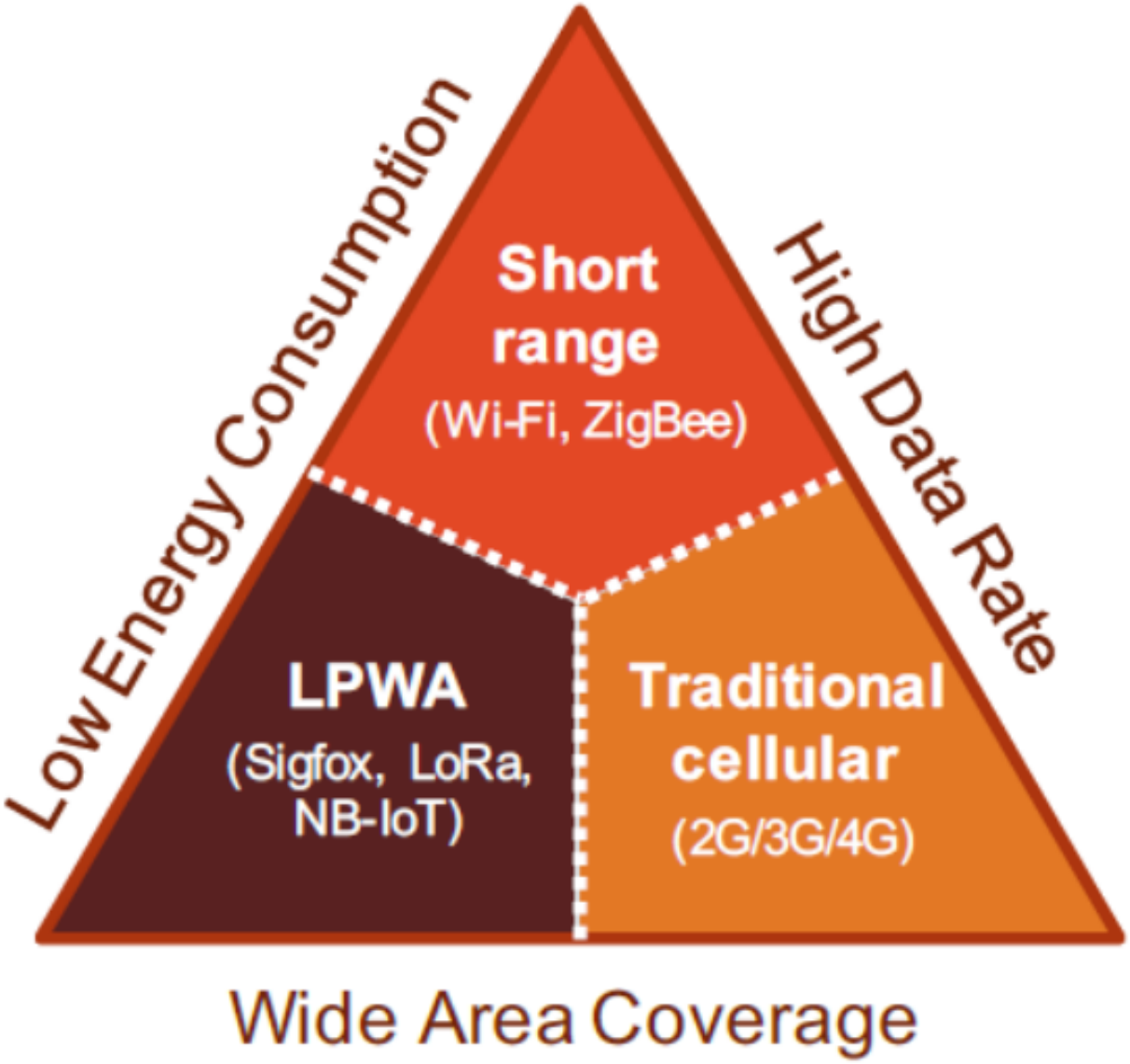
Src Telenor Connexion, 2016

Design Constraints on the Technical Level: Data Rate vs Coverage



SRC: LG

Design Constraints on the Technical Level: Data Rate vs Coverage vs Energy Consumption

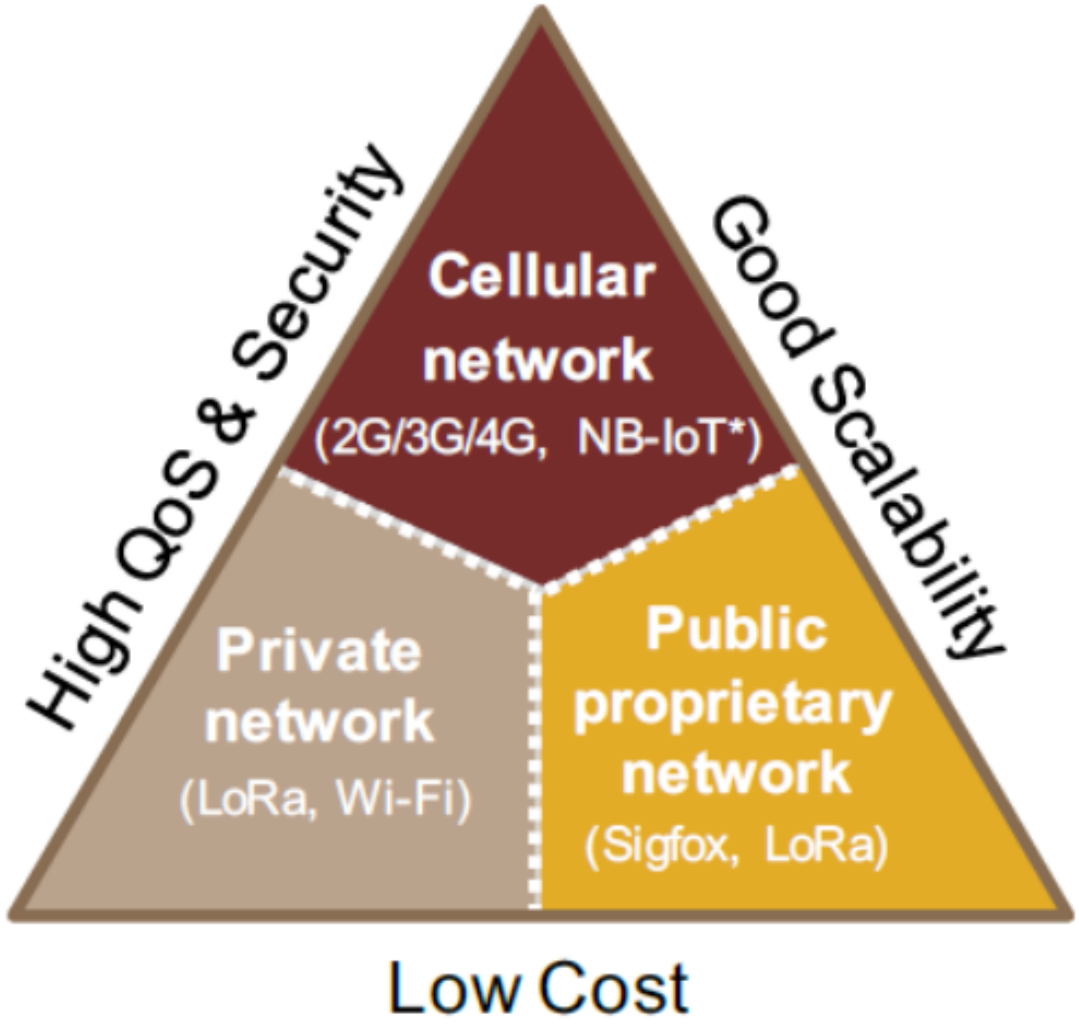


Src Telenor Connexion, 2016

Trade-offs on the Commercial Level: QoS vs Scalability vs. Cost

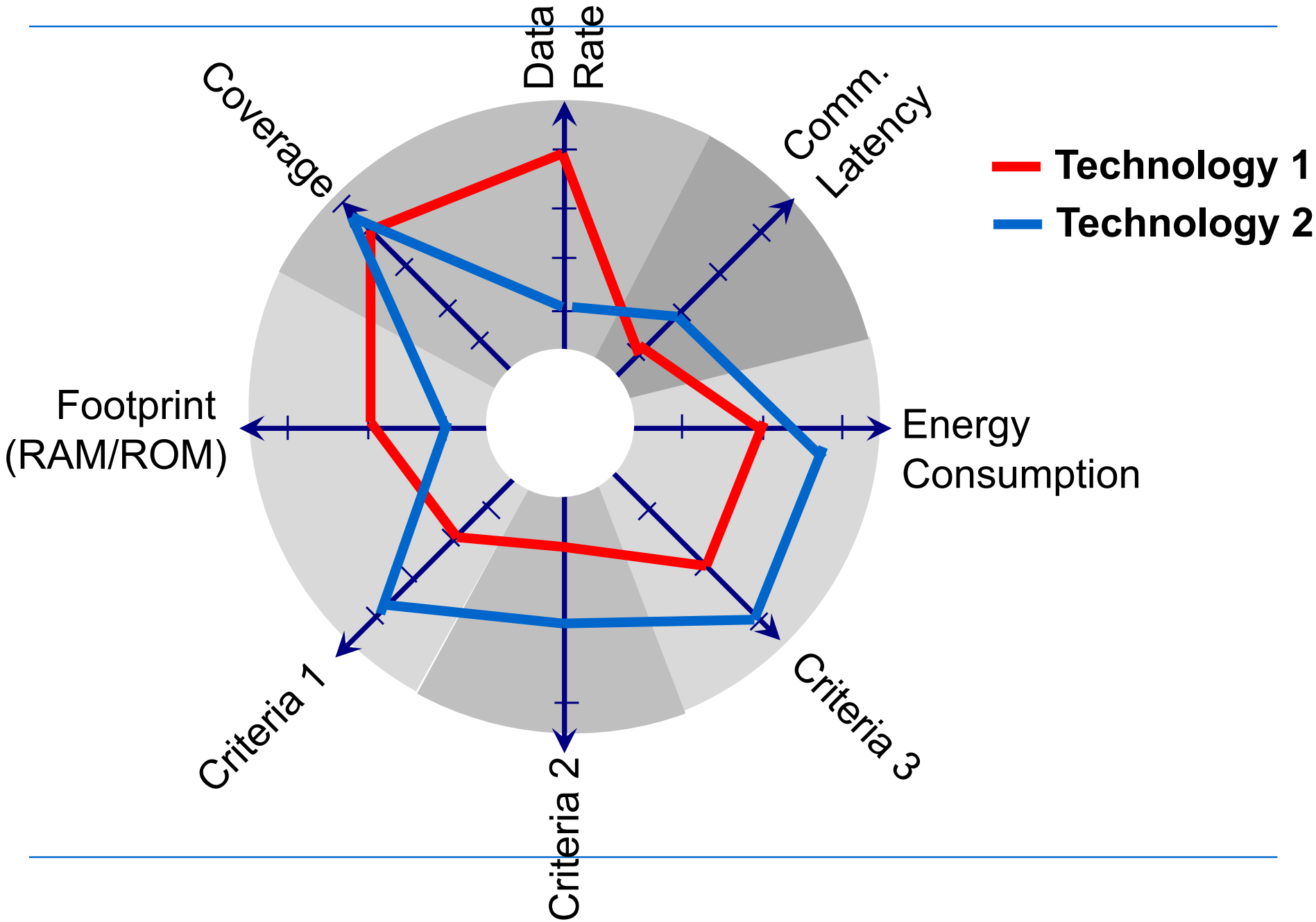
Main cost components of an IoT Device

- Connectivity module
- Sensors
- Battery
- Processor

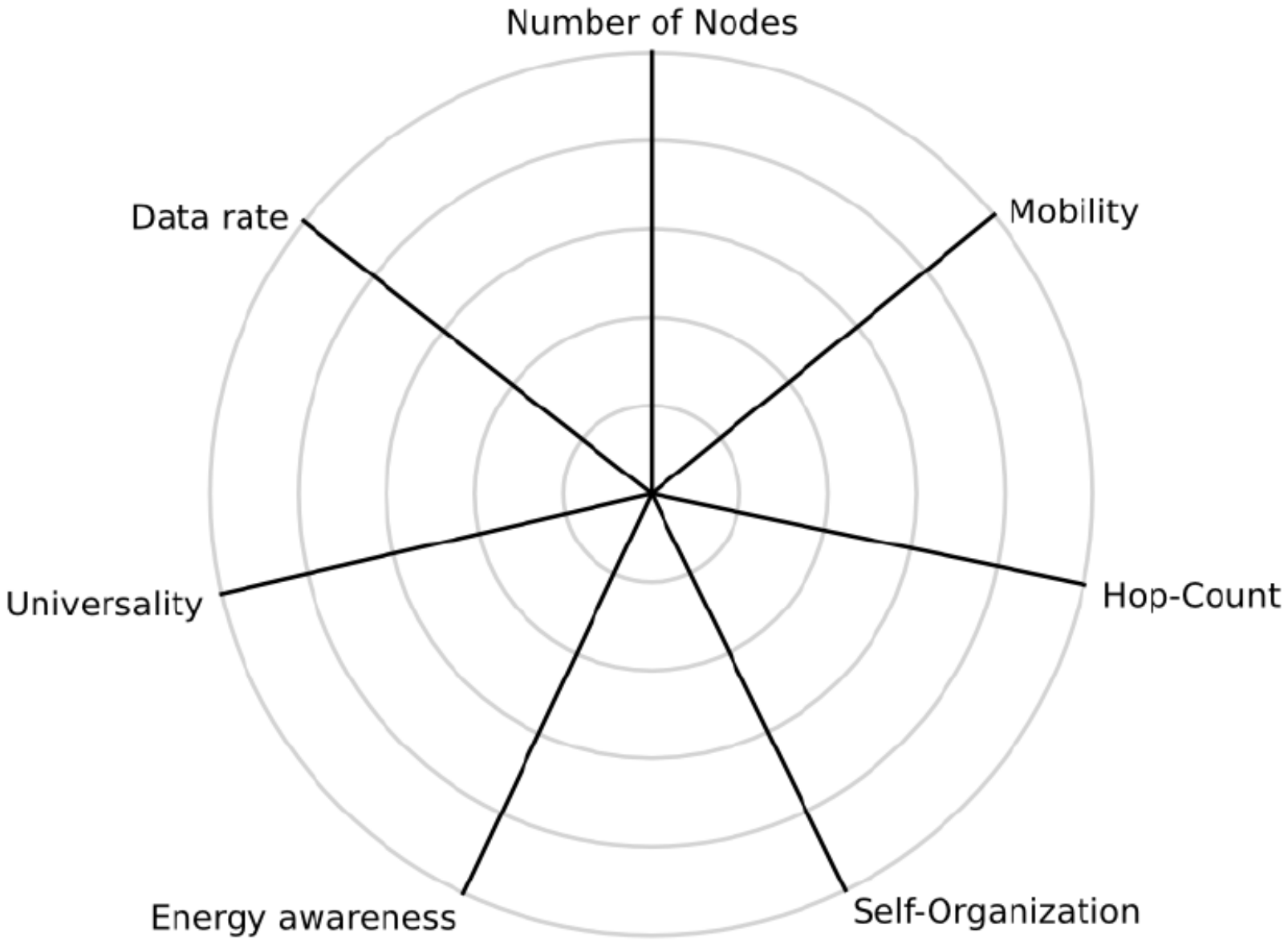


Src Telenor Connexion, 2016

Spider Diagrams

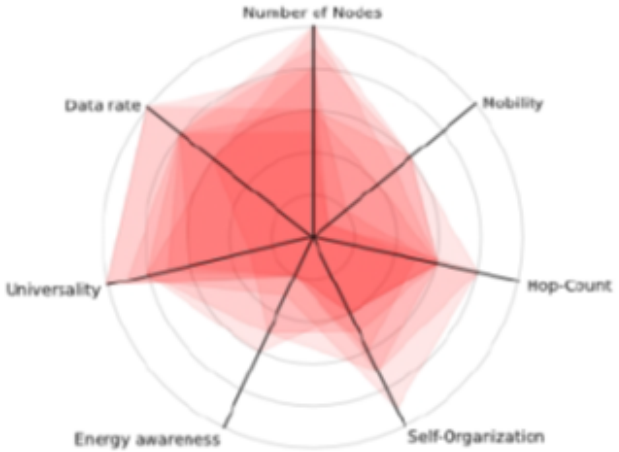


Characteristics of Wireless Networks (1)



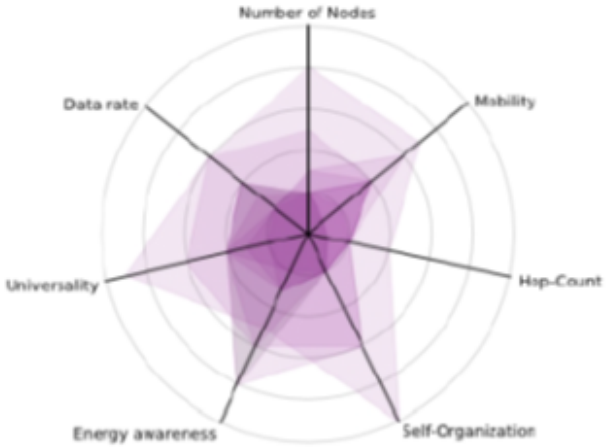
Characteristics of Wireless Networks (2)

Wireless Sensor Networks (WSN)



Wireless Mesch Network (WMN)

Mobile Ad Hoc Networks (MANET)



Wireless Personal Area Networks (WPAN)

Suitability for Application Areas

Application area	Need for:												Suitable connectivity technologies
	Indoor coverage	Wide area coverage	Energy efficiency	High data rate	Bidirectional comm.	Mobility	Localization	QoS & security	Low cost	Scalability	Future proofness	Global reach & interoperability	
Automotive	L	H	L	H	H	H	M	H	L	H	H	H	Cellular
Smart cities	L M	M	H	L	L M	L	L	L	H	H	M	L	LPWA, Wi-Fi
Safety and security	H	M	L	L H	H	L	L	H	L	M	M	L	Cellular, LPWA
Industrial manufacturing	H	L	L	M	M H	L	L	H	M	L M	M H	L	Wi-Fi, Cellular
Fleet and logistics	M	H	L	L	L	H	H	M	L M	M	M	H	Cellular, LPWA
Asset management	H	H	L	L	L M	L	M	L	M	L	H	H	Cellular, LPWA
Agriculture and environment	L	H	H	L	L	L	L	L	H	H	M	L	LPWA
Smart meters	H	H	H	L	M	L	L	L	M	M H	H	L	Cellular, LPWA
Wearables	H	L	L	L	L M	L	L	M	H	L	L	L	Bluetooth, ZigBee

L = Low, M = Medium, H = High

Src Telenor Connexion, 2016

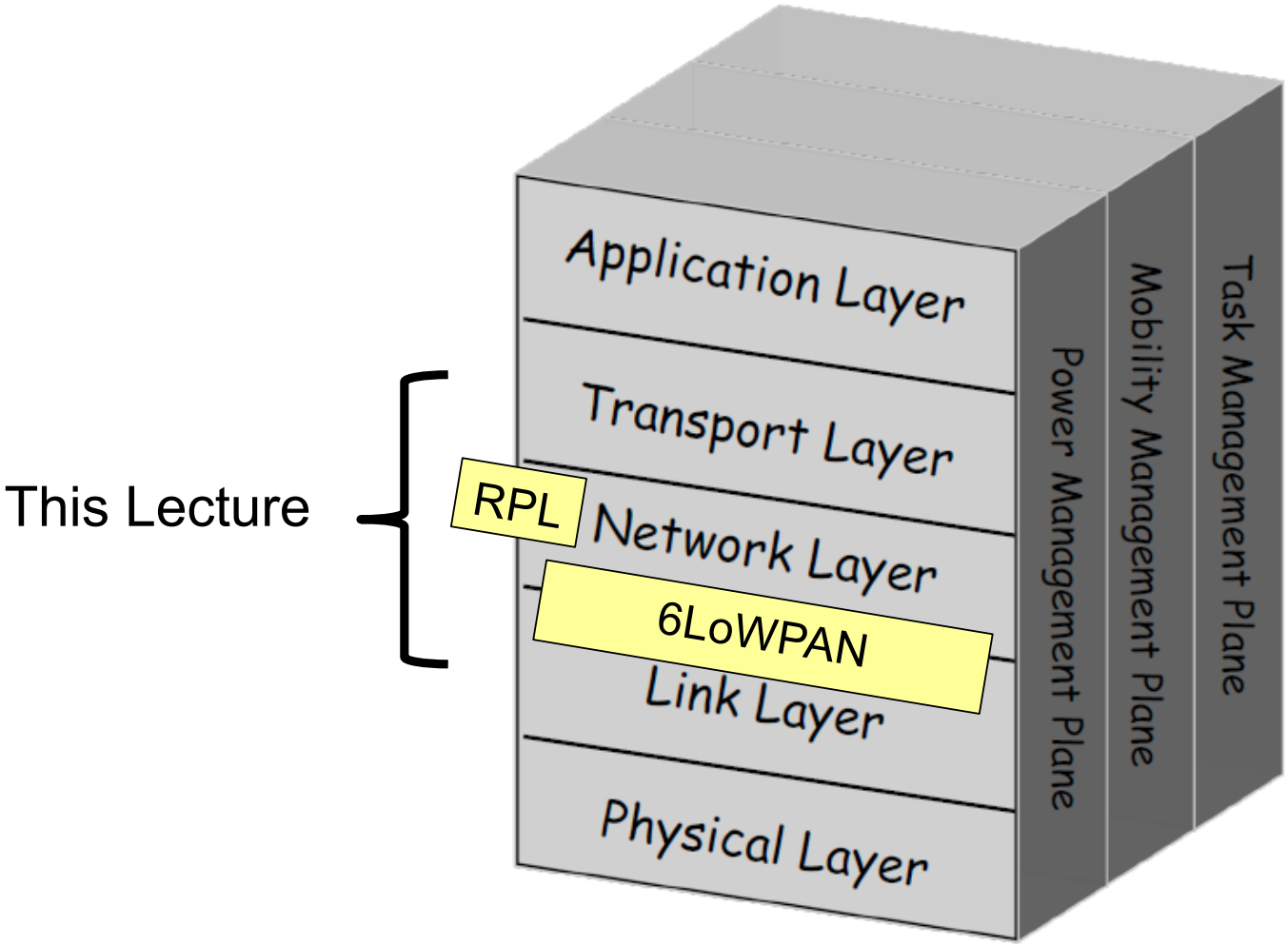
Takeaways

- Rich landscape of technologies
- Selection of one or more technologies is a fundamental decision for the realization of an IoT application

Networking for IoT

6LowPAN, RPL

The Protocol Stack



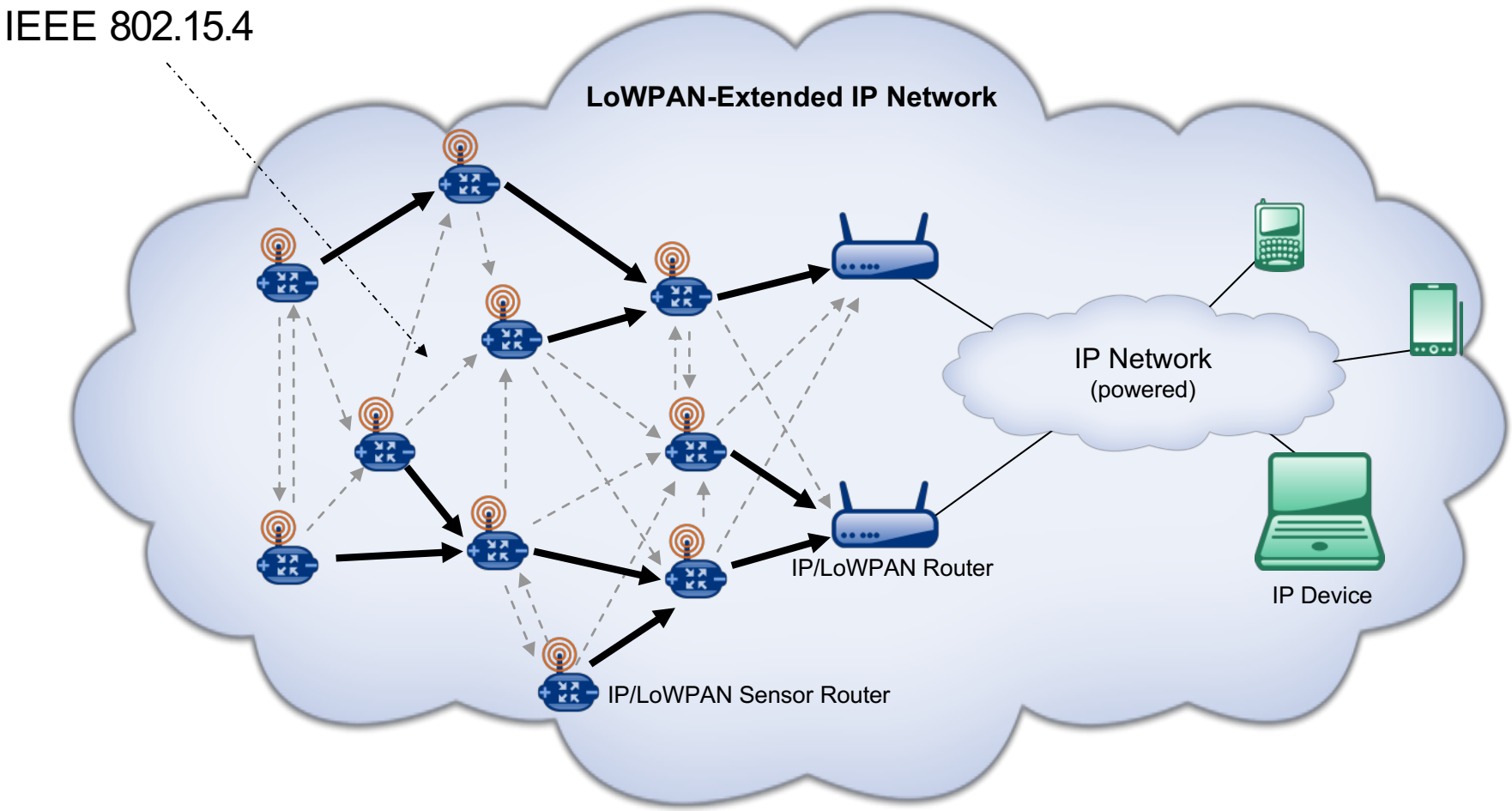
Chapter Outline

- Motivation
  - Low-Power and Lossy Networks (LLN)
  - Why all-IP?
- 6LowPAN
  - 6LowPAN Adaptation Layer
  - 6LowPAN Header Compression
  - 6LowPAN Fragmentation
- Routing Protocol for Low-Power and Lossy Networks (RPL)
  - Concept
  - RPL Control Messages
  - DODAG Formation Example

Low power and Lossy Networks (LLNs)

- LLNs are composed of many embedded devices:
  - restricted in processing power, memory and energy (battery)
    - Duty cycling to save energy (sleep, idle, active..)
  - interconnected by a variety of links, such as IEEE 802.15.4 or Low Power WiFi, characterized by
    - high loss rates,
    - low data rates and
    - instability
- There exist many protocols for LLNs and consider the network to follow the **source - sink architecture**. Thus, most of the protocols are designed to support **multipoint-to-point (M2P)** or **point-to-multipoint (P2M)** communications
- UDP rather than TCP like communication

LoWPAN-Extended IP Network



Src: D.E. Culler et al., 2008

Many Advantages of IP

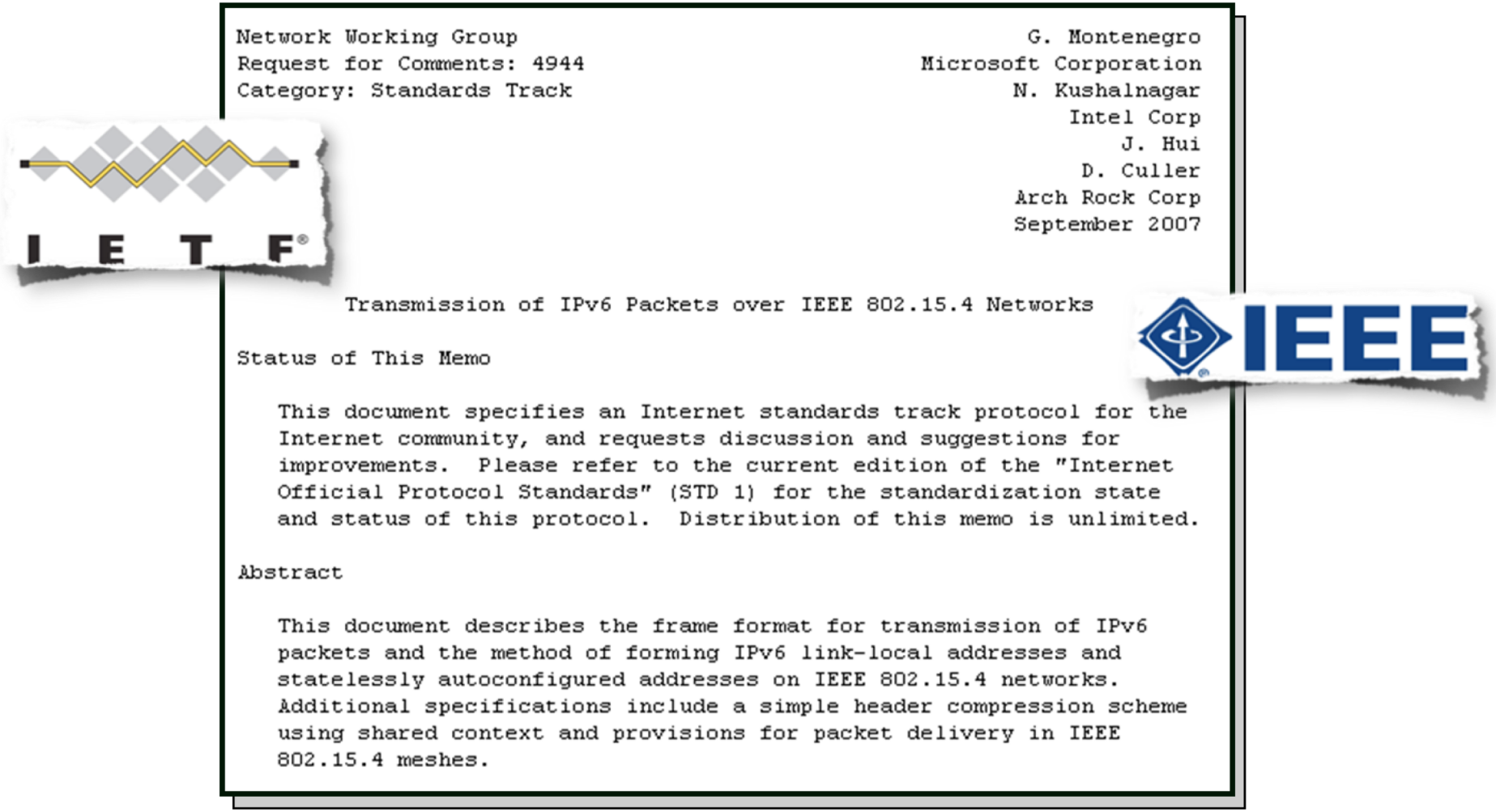
- Extensive interoperability
  - Other wireless embedded 802.15.4 network devices
  - Devices on any other IP network link (WiFi, Ethernet, GPRS, Serial lines, …)
- Established security
  - Authentication, access control, and firewall mechanisms
  - Network design and policy determines access, not the technology
- Established naming, addressing, translation, lookup, discovery
- Established proxy architectures for higher-level services
  - NAT, load balancing, caching, mobility
- Established application level data model and services
  - HTTP/HTML/XML/SOAP/REST, Application profiles
- Established network management tools
  - Ping, Traceroute, SNMP, … OpenView, NetManager, Ganglia, …
- Transport protocols
  - End-to-end reliability in addition to link reliability
- Most “industrial” (wired and wireless) standards support an IP option

→ Leverage existing standards, rather than “reinventing the wheel”

6LoWPAN: IPv6 over Low power lossy Personal Area Networks

(many slides are from D.E. Culler et al. 2008)

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

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.

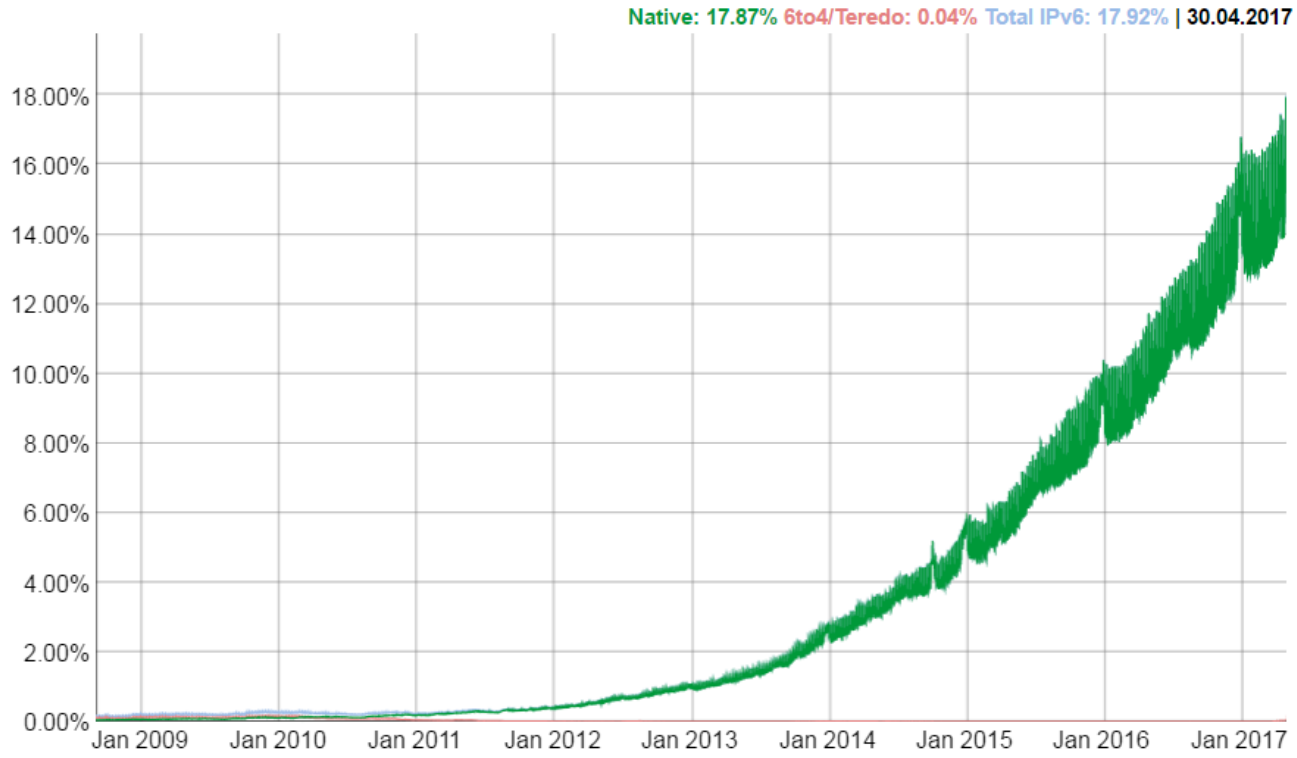
IEEE Wireless links

	802.15.4	802.15.1	802.15.3	802.11	802.3
Class	WPAN	WPAN	WPAN	WLAN	LAN
Lifetime (days)	100-1000+	1-7	Powered	0.1-5	Powered
Net Size	65535	7	243	30	1024
BW (kbps)	20-250	720	11,000+	11,000+	100,000+
Range (m)	1-75+	1-10+	10	1-100	185 (wired)
Goals	Low Power, Large Scale, Low Cost	Cable Replacement	Cable Replacement	Throughput	Throughput

- ZigBee
  - only defines communication between 15.4 nodes (“layer 2” in IP terms), not the rest of the network (other links, other nodes).
  - defines new upper layers, all the way to the application, similar to IRDA/USB/Bluetooth, rather than utilizing existing standards.

IPv6: Re-Designing Addressing

- IPv6 (RFC 2460)
  - Complete redesign of IP addressing
  - Hierarchical 128-bit address
- Majority of traffic not yet IPv6 but exponential increase

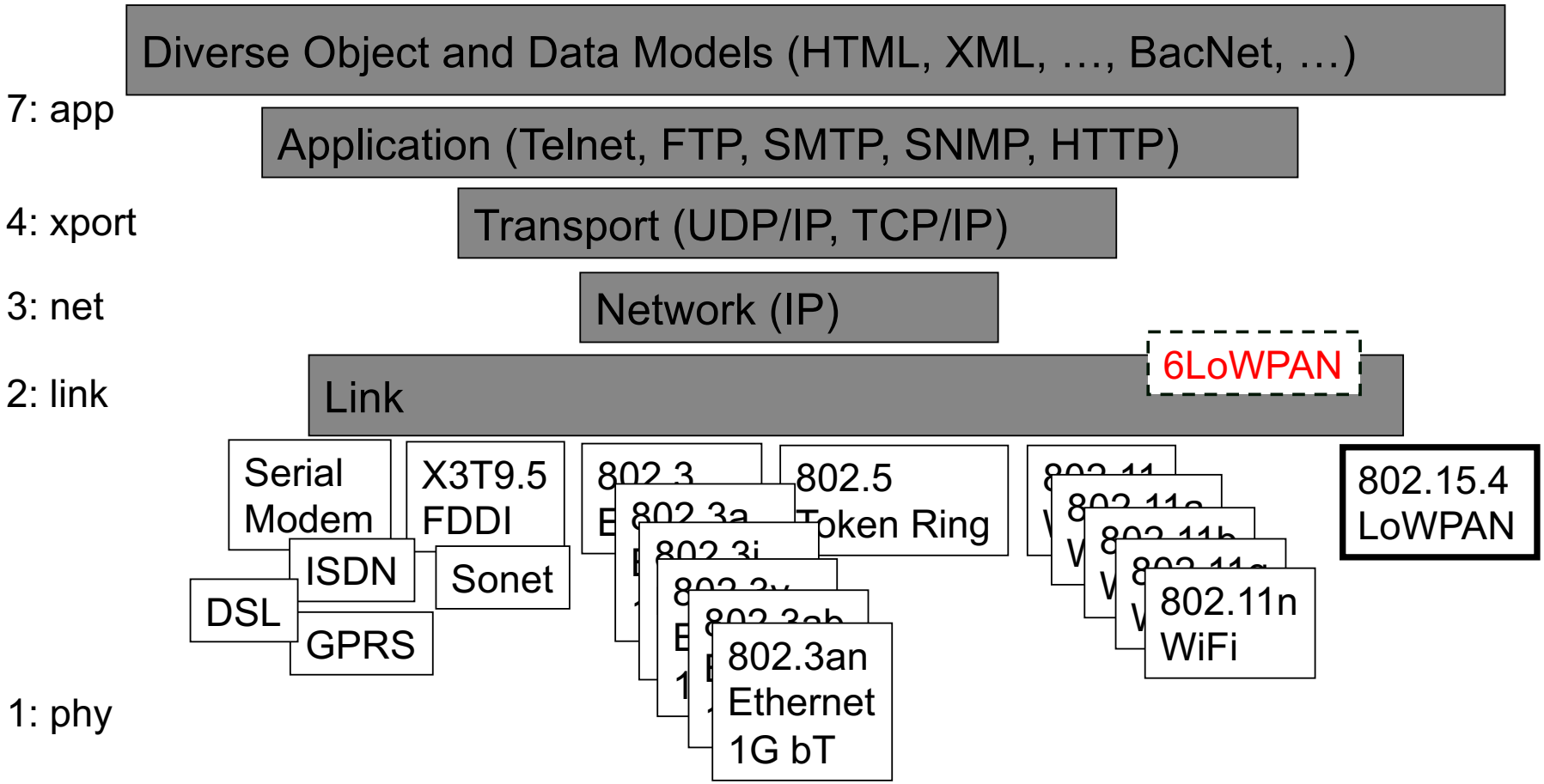


Src: Google

IEEE 802.15.4

- Allows mesh networking: Full function nodes can forward packets to other nodes
- A PAN coordinator (like WiFi Access Point) allows nodes to join the Internet
- Nodes have 64-bit addresses
  - Coordinator assigns 16-bit short addresses for use during the association
- Max Frame size is 127 bytes

Internet Concepts: Layering



Key Factors for IP over 802.15.4

- Header
  - Standard IPv6 header is 40 bytes [RFC 2460]
  - Entire 802.15.4 MTU is 127 bytes [IEEE]
  - Often data payload is small
- Fragmentation
  - Interoperability means that applications need not know the constraints of physical links that might carry their packets
  - IP packets may be large, compared to 802.15.4 max frame size
  - IPv6 requires all links support 1280 byte packets [RFC 2460]

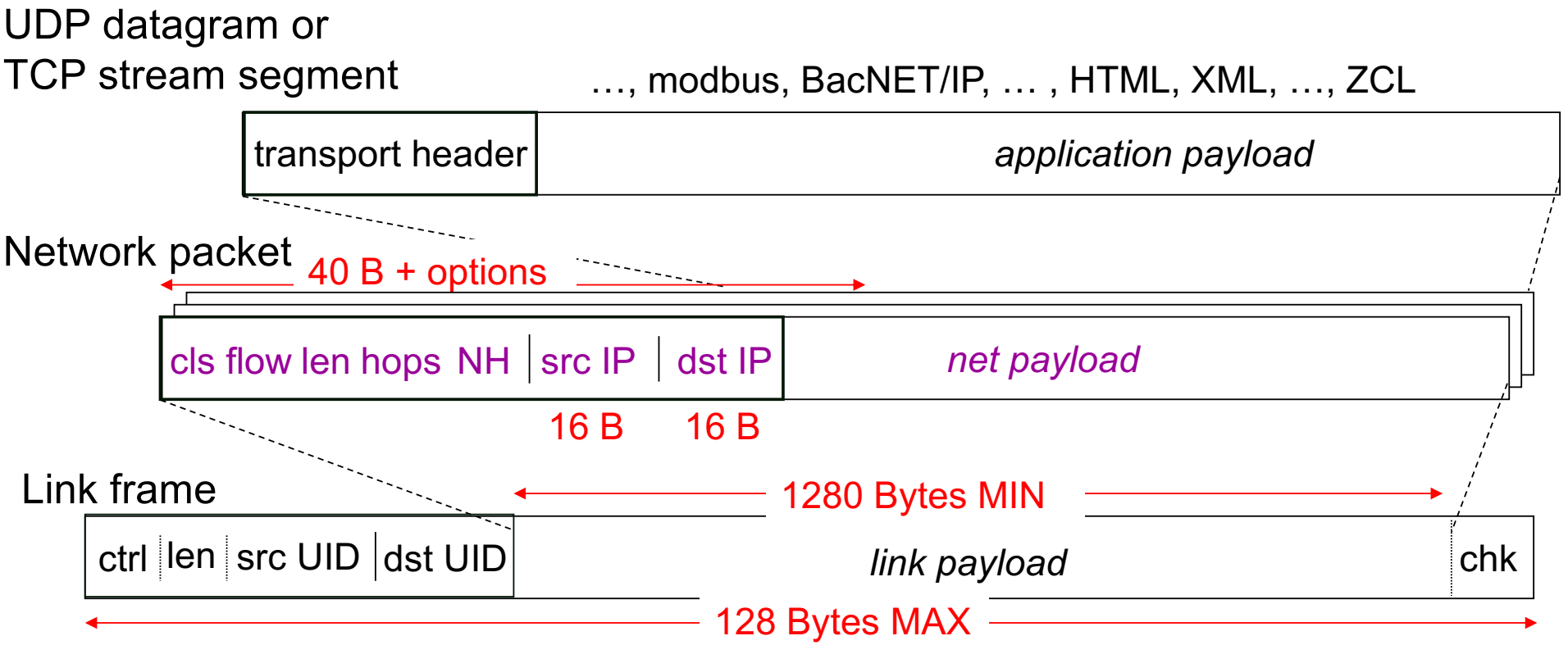
6LoWPAN

- The primary driving force behind the growth of IoT is the effectiveness of 6LoWPAN (IPv6 over Low power lossy personal area networks).
- 6LoWPAN is an **adaptation layer** between the network layer and the data link layer [2]. The primary function of the 6LoWPAN is to **convert IPv6 packets from the network layer to short IEEE802.15.4 frames**. To encapsulate IPv6 packets in IEEE802.15.4 frames [1], 6LoWPAN requires performing IPv6 **header compression**, **and fragmentation & defragmentation**. The 6LoWPAN adaptation layer also performs **routing between the nodes within the network**.

[1] IEEE, “Ieee standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans) amendment 3: Physical layer (phy) specifications for low-data-rate, wireless, smart metering utility networks,” IEEE Std 802.15.4g-2012 (Amendment to IEEE Std 802.15.4-2011), April 2012, pp. 1–252.

[2] N. Kushalnagar, G. Montenegro, and C. Schumacher, “Ipv6 over lowpower wireless personal area networks (6lowpans): overview, assumptions, problem statement, and goals,” Internet Requests for Comments, RFC 4914, August 2007.

6LoWPAN Challenges



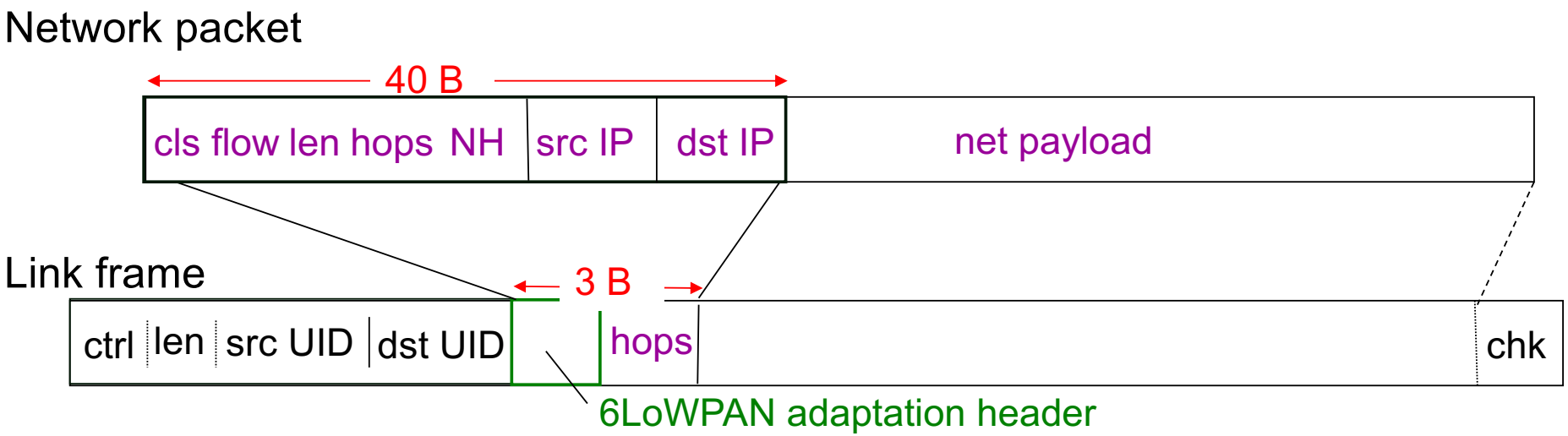
- Large IP Address & Header => 16 bit short address / 64 bit EUID
- Minimum Transfer Unit => Fragmentation
- Short range & Embedded => Multiple Hops

The Header Size Problem

Worst-case scenario calculation:

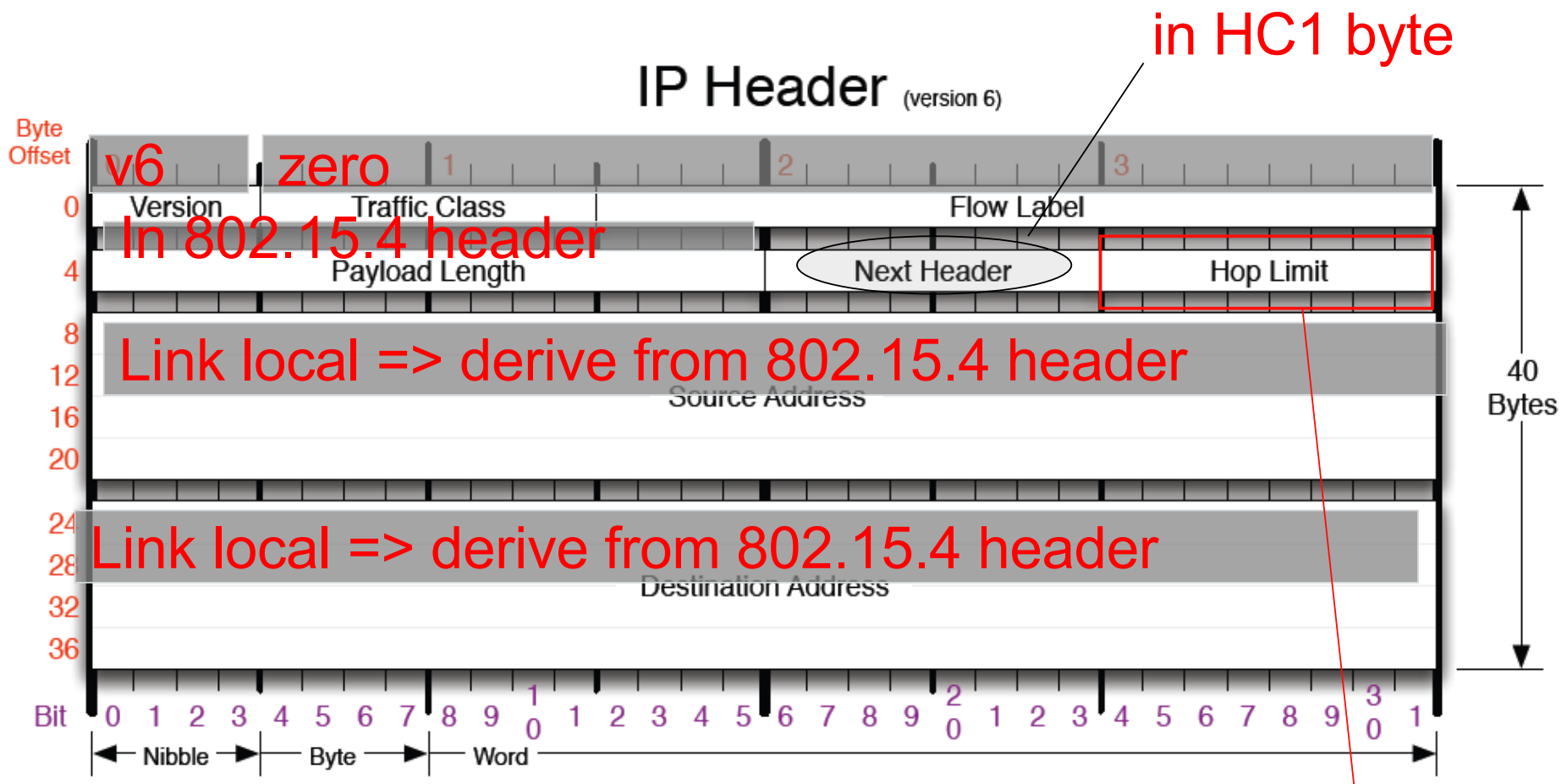
- Max Frame size in 802.15.4: 127 byte
- Reduced by the max frame header (25 byte): 102 byte
- Reduced by highest link-layer security (21 byte): 81 byte
- Reduced by standard Ipv6 header (40 byte): 41 byte
- Reduced by standard UDP header (8 byte): 33 byte
- This leaves **33 byte for actual payload** (the rest is used by headers!)

6LoWPAN – IP Header Optimization



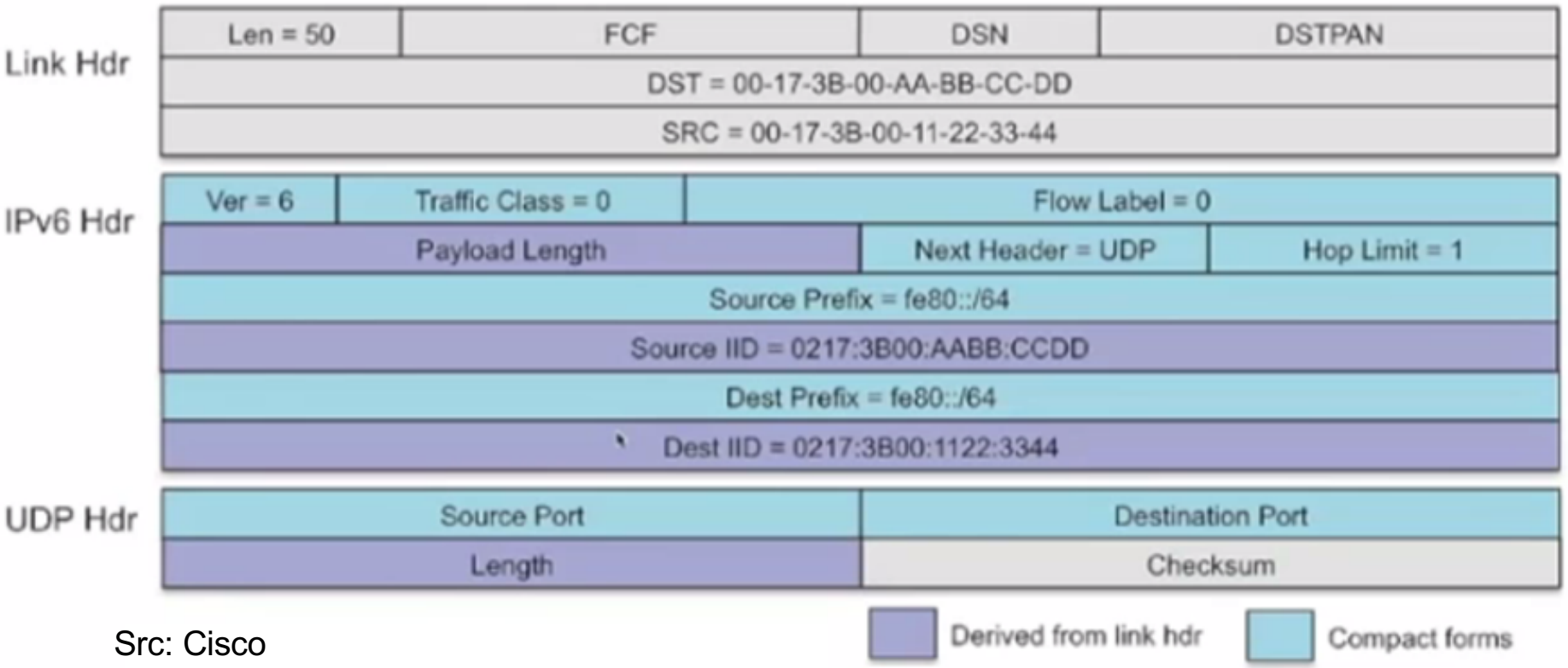
- Eliminate all fields in the IPv6 header that can be derived from the 802.15.4 header in the common case
  - Source address : derived from link address
  - Destination address : derived from link address
  - Payload length : derived from link frame length
  - Traffic Class & Flow Label : may be elided
  - Next header : UDP, TCP, or ICMP
- Additional IPv6 options follow as options

IPv6 Header Compression



http://www.visi.com/~mjb/Drawings/IP_Header_v6.pdf

Further Header Compression (Example)

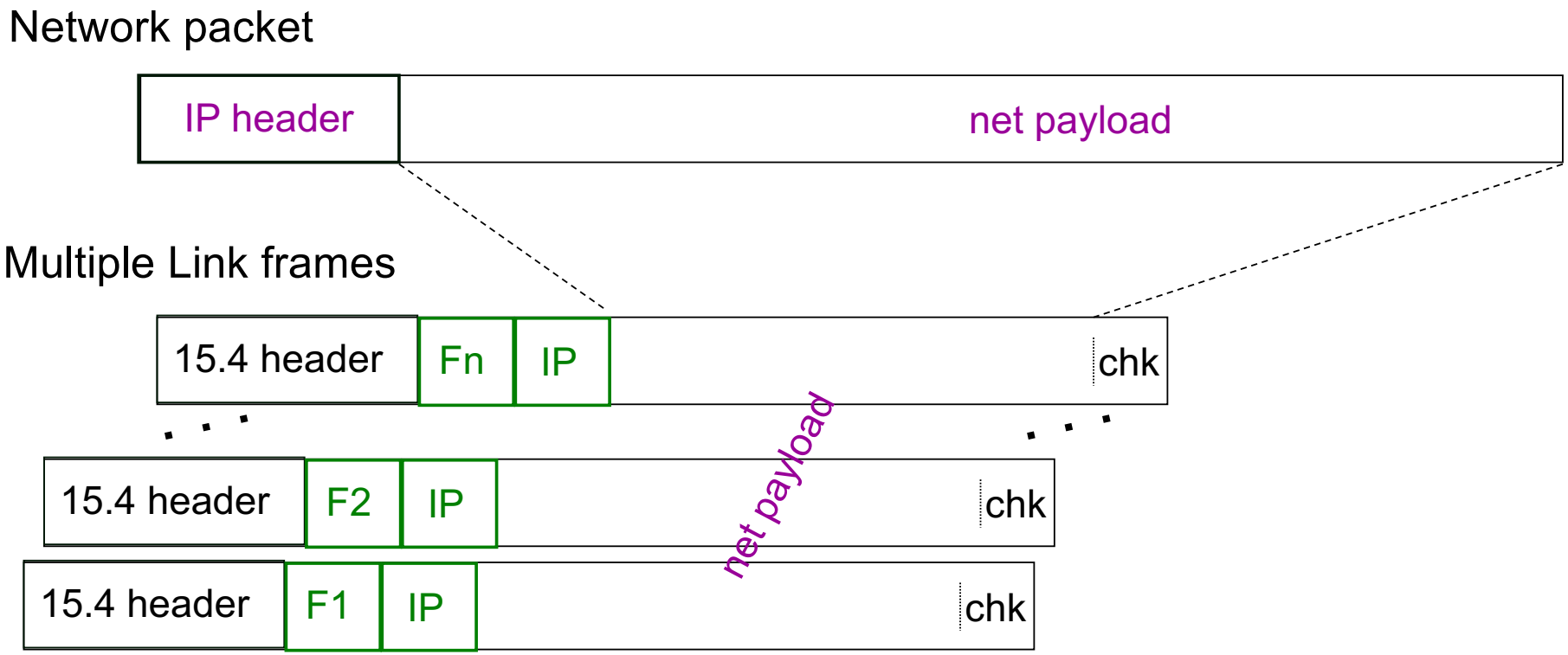


Src: Cisco

48-byte UDP/IPv6 header → 7-byte 6loWPAN header

6LoWPAN Fragmentation

- IP interoperability over many links => users not limited by frame size
- IP datagrams that are too large to fit in a 802.15.4 frame are fragmented into multiple frames
  - Self describing for reassembly

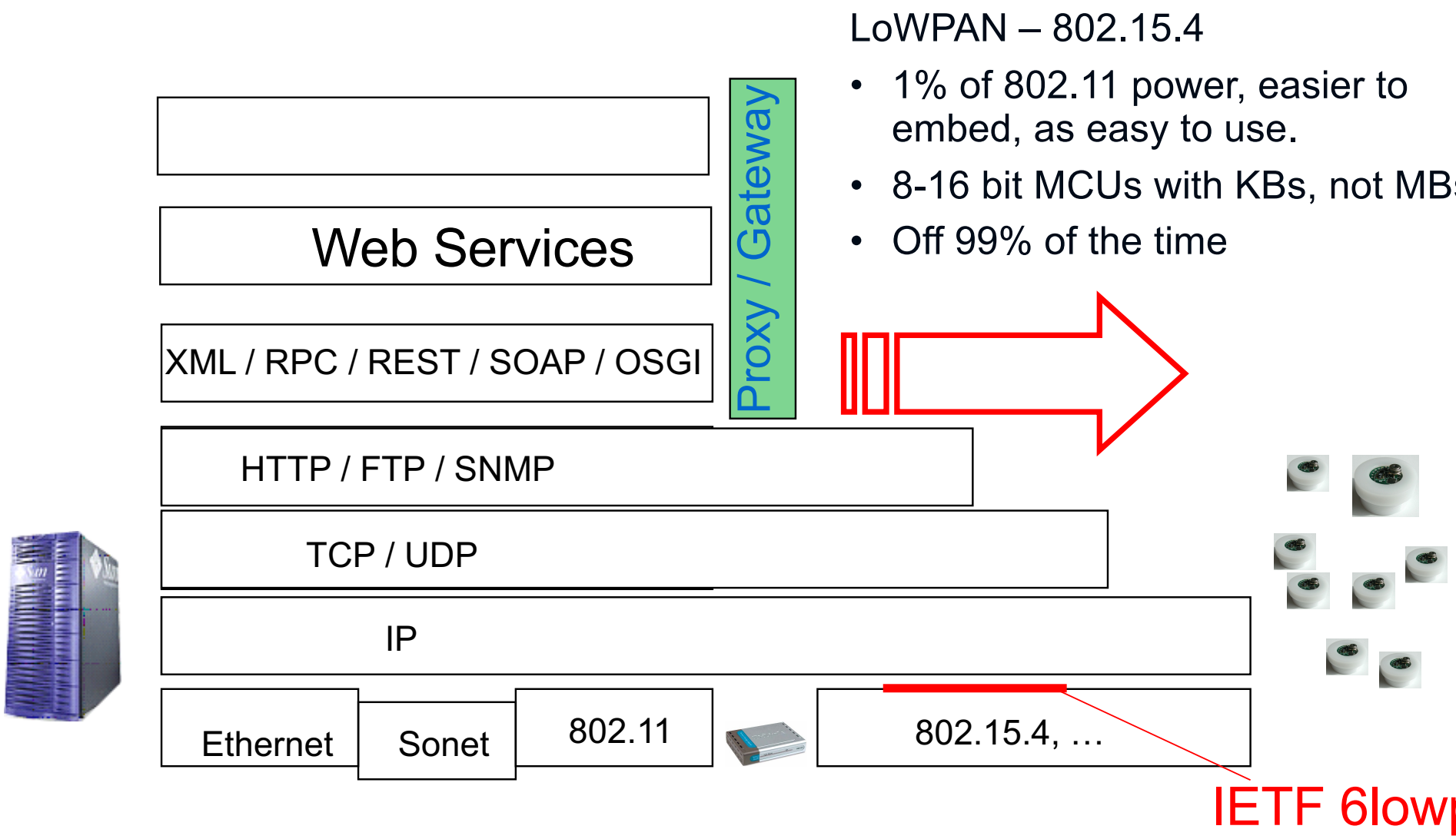


6LoWPAN … What it means for sensors

- Low-Power Wireless Embedded devices can now be connected using familiar networking technology,
  - like ethernet (but even where wiring is not viable)
  - and like WiFi (but even where power is not plentiful)
- all of these can interoperate in real applications
- **Interoperate** with traditional computing infrastructure
- Utilize modern **security** techniques

- **Application Requirements and Capacity Plan** dictate how the network is organized,
  - **not** artifacts of the underlying technology

… Making Sensor Nets Make Sense



RPL (ripple): Routing Protocol for Low Power and Lossy Networks

Routing Over Low Power and Lossy Networks (ROLL)

IETF working group focused on routing issues for LLNs:

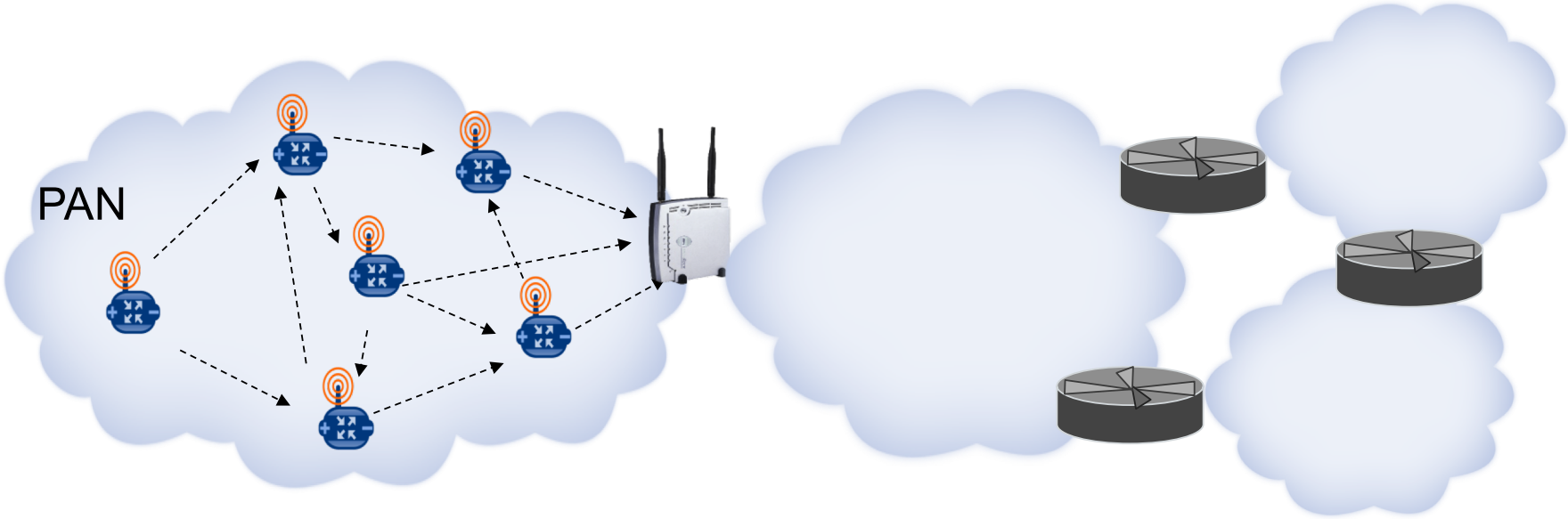
- Routing requirements specification for various application areas of LLNs
  - Home automation
  - Commercial buildings automation
  - Industrial automation
  - Urban environments
- Evaluation of existing routing protocols in the scope of LLNs
- Solution must work over IPv6 and 6LoWPAN
- Routing Protocol for Low power and lossy networks (RPL)

Survey of Existing Routing Protocols

Protocol	State	Loss	Control	Link Cost	Node Cost
OSPF/IS-IS	Fail	Fail	Fail	Pass	Fail
OLSRv2	Fail	?	?	Pass	pass
TBRPF	fail	Pass	Fail	Pass	?
RIP	pass	Fail	pass	?	fail
AODV	pass	fail	pass	fail	fail
DYMO	pass	?	pass	?	?
DSR	fail	pass	pass	Fail	fail

- Routing State - limited memory resources of low-power nodes.
- Loss Response - what happens in response to link failures.
- Control cost - constraints on control traffic.
- Link&Node cost - link and node properties are considered when choosing routes.

RPL („Ripple“)

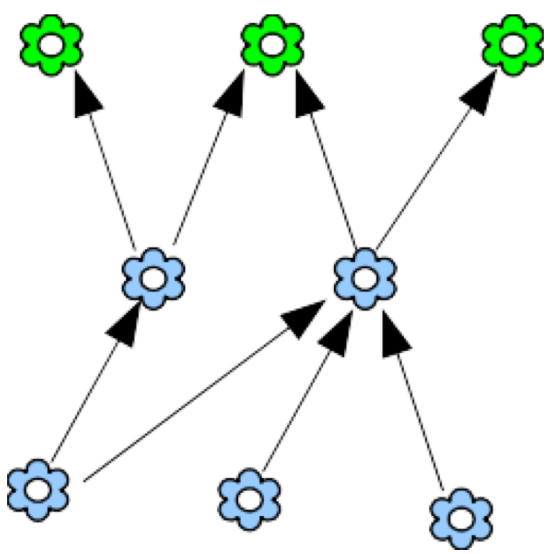


- Short-range radios & Obstructions => Multi-hop Communication is often required
  - i.e. Routing and Forwarding
- Proactive distance-vector approach

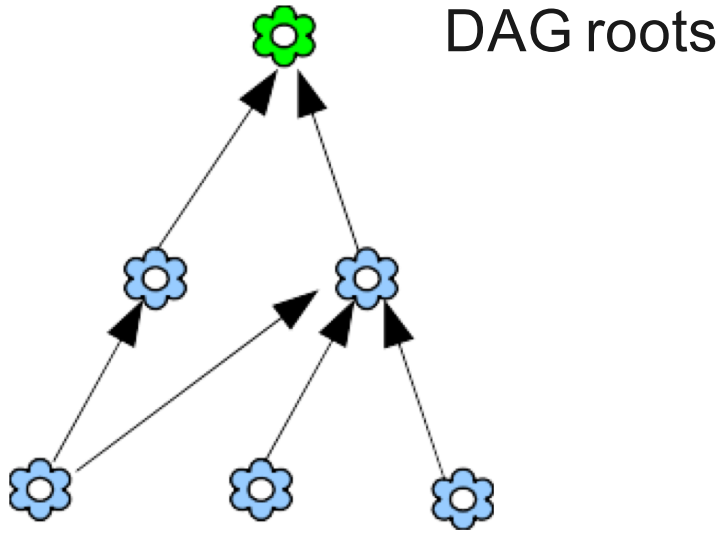
RPL: IPv6 Routing Protocol for LLNs

Definitions:

- Directed Acyclic Graph (DAG) - a directed graph where no cycles exist.
- Destination Oriented DAG (DODAG) - a DAG rooted at a single destination.



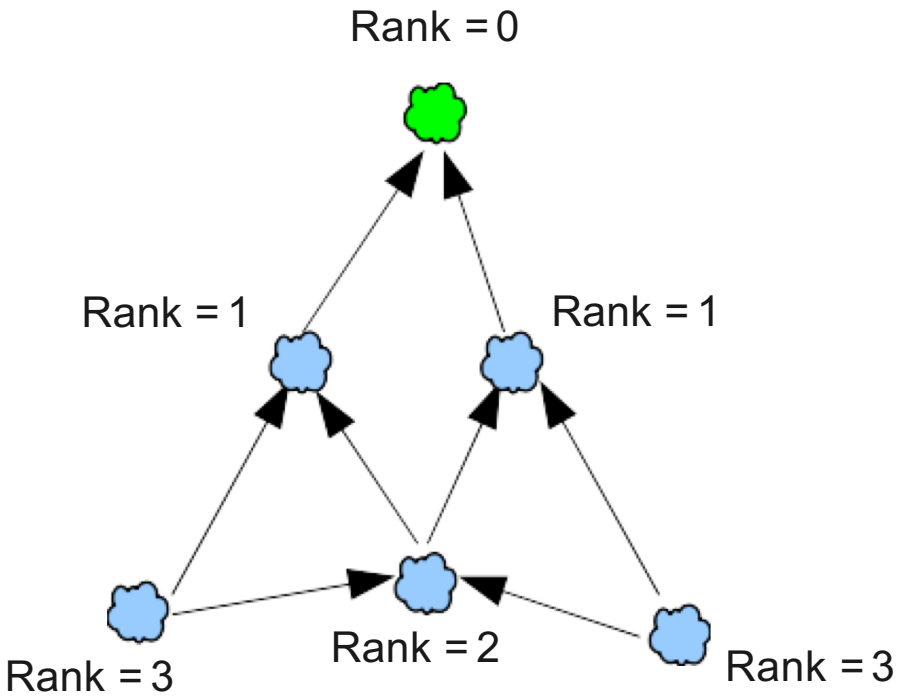
DAG



DODAG

RPL Node Rank

- Defines a node's relative position within a DODAG with respect to the DODAG "root".

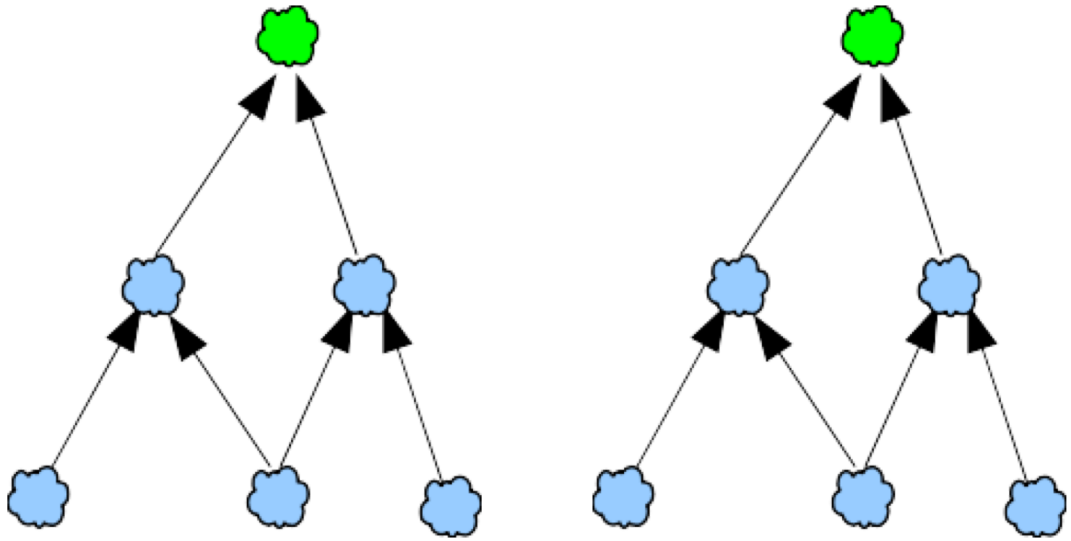


RPL: IPv6 Routing Protocol for LLNs

- Assumption: most traffic in LLNs flows through few nodes
  - many-to-one
  - one-to-many
- Approach: build a topology (Instance) where routes to these nodes are optimized (DODAG(s) rooted at these nodes)

RPL Instance

- Defines Optimization Objective when forming paths towards roots based on one or more metrics
- Metrics may include both Link properties (Reliability, Latency) and Node properties (Powered or not)
- A network may run multiple instances concurrently with different optimization criteria



Instance may include several DODAGs

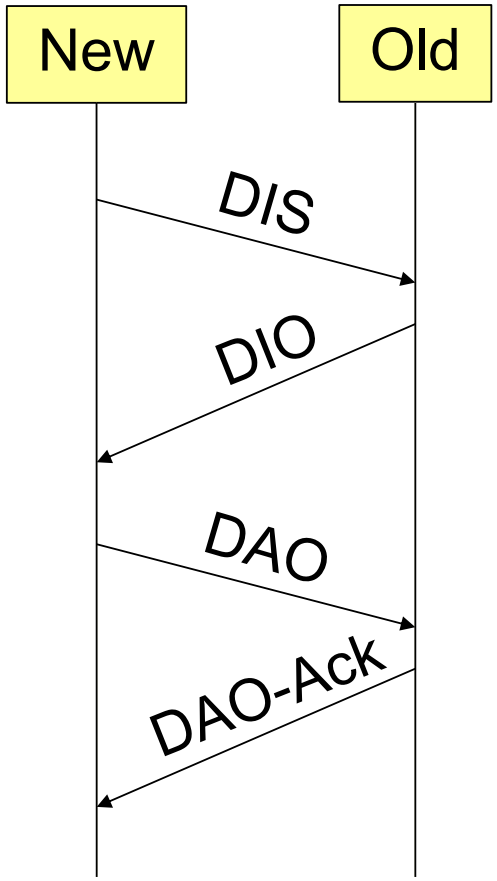
RPL Control Messages

RPL defines a new ICMPv6 message with three possible types:

- DAG Information Object (DIO) - carries information that allows a node to discover an RPL Instance, learn its configuration parameters and select DODAG parents
- DAG Information Solicitation (DIS) - solicit a DODAG Information Object from a RPL node
- Destination Advertisement Object (DAO) - used to propagate destination information upwards along the DODAG.

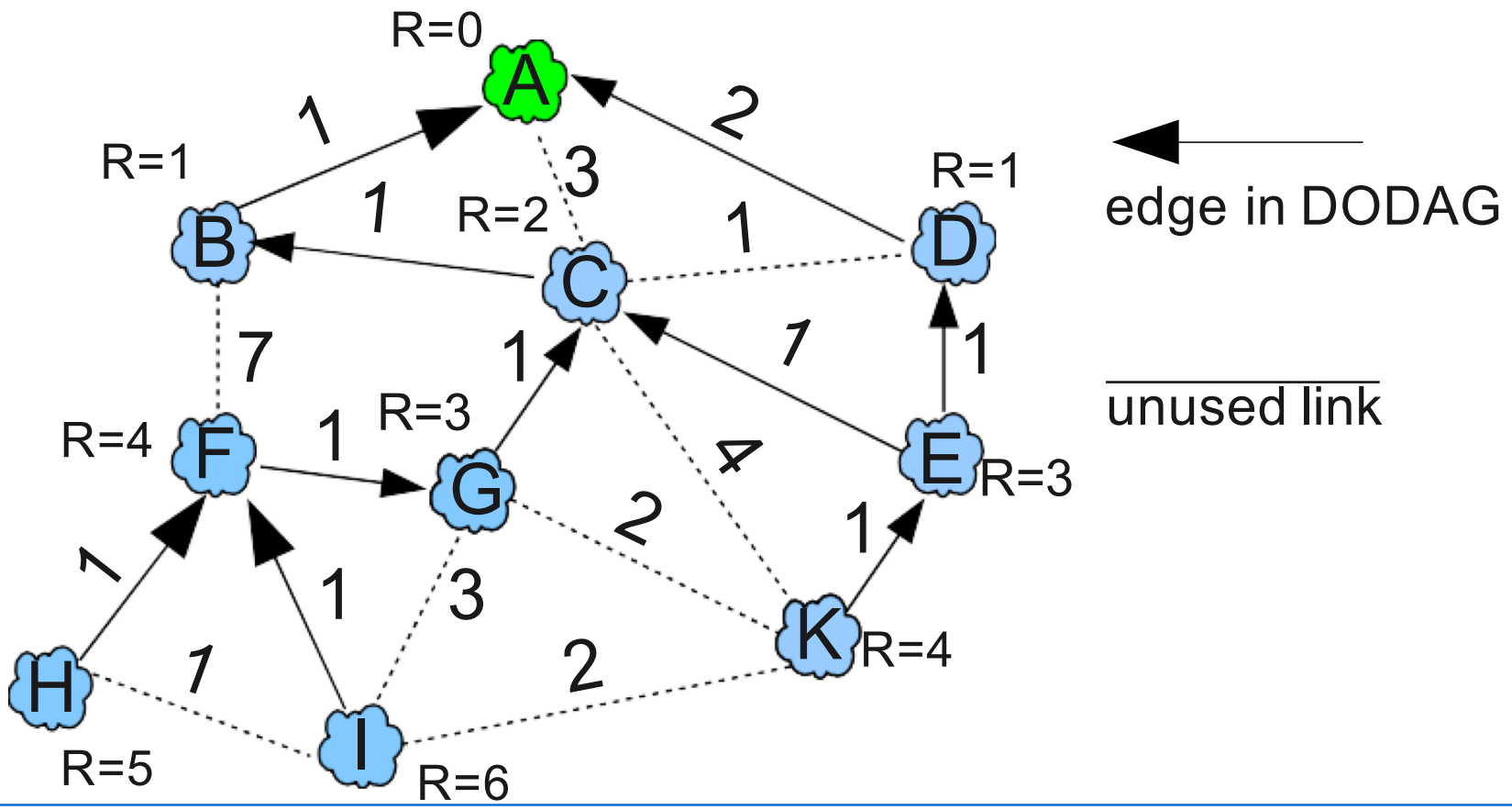
DODAG Construction

- Nodes periodically send link-local multicast DIO messages
  - Stability or detection of routing inconsistencies influence the rate of DIO messages
  - Nodes listen for DIOs and use their information to join a new DODAG, or to maintain an existing DODAG
- Nodes may use a DIS message to solicit a DIO
  - Based on information in the DIOs the node chooses parents that minimize path cost to the DODAG root
- Result: Upward routes towards the DODAG root



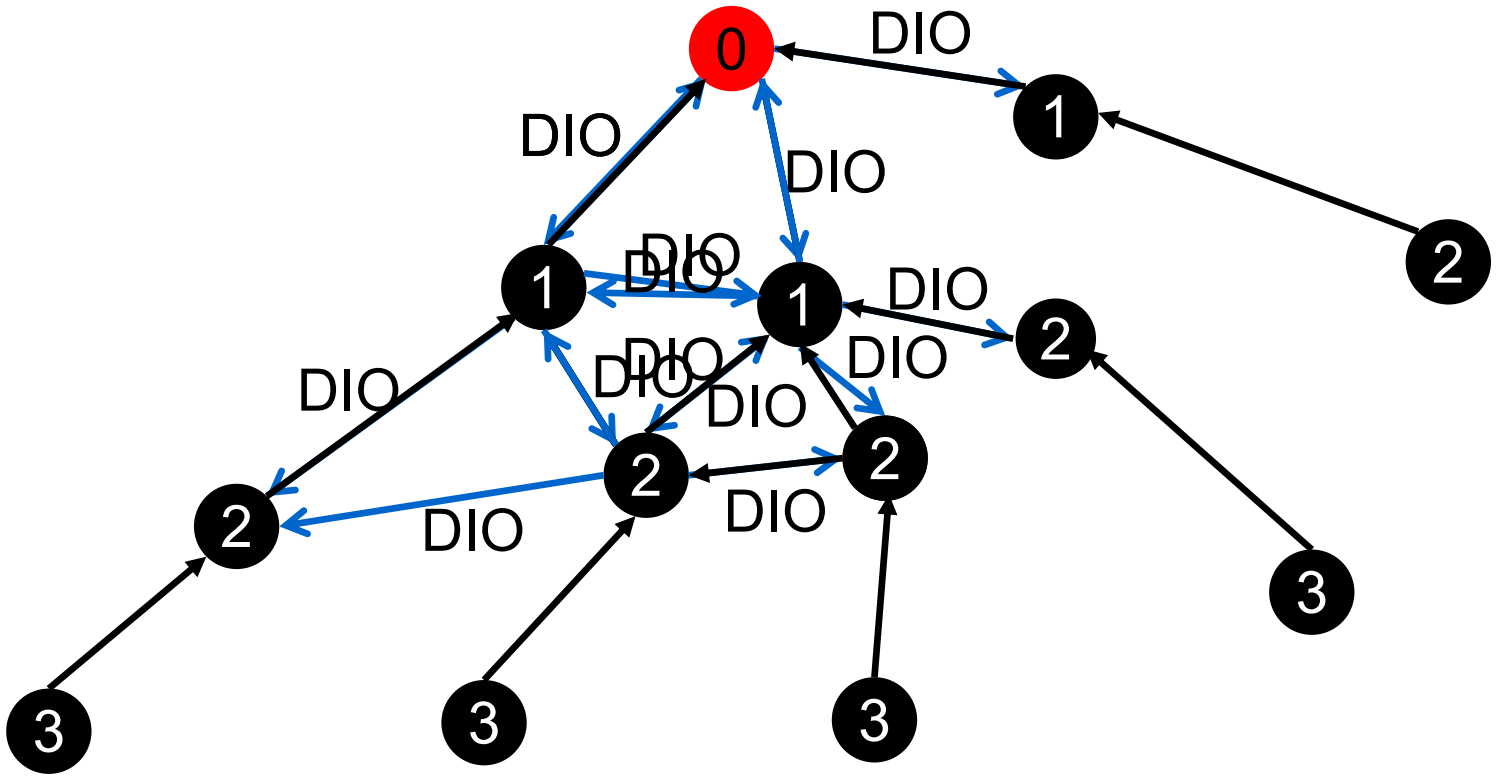
DODAG Example

- Each node has a set of parent nodes
- A node has no knowledge about children → ONLY upward routes



RPL Operation

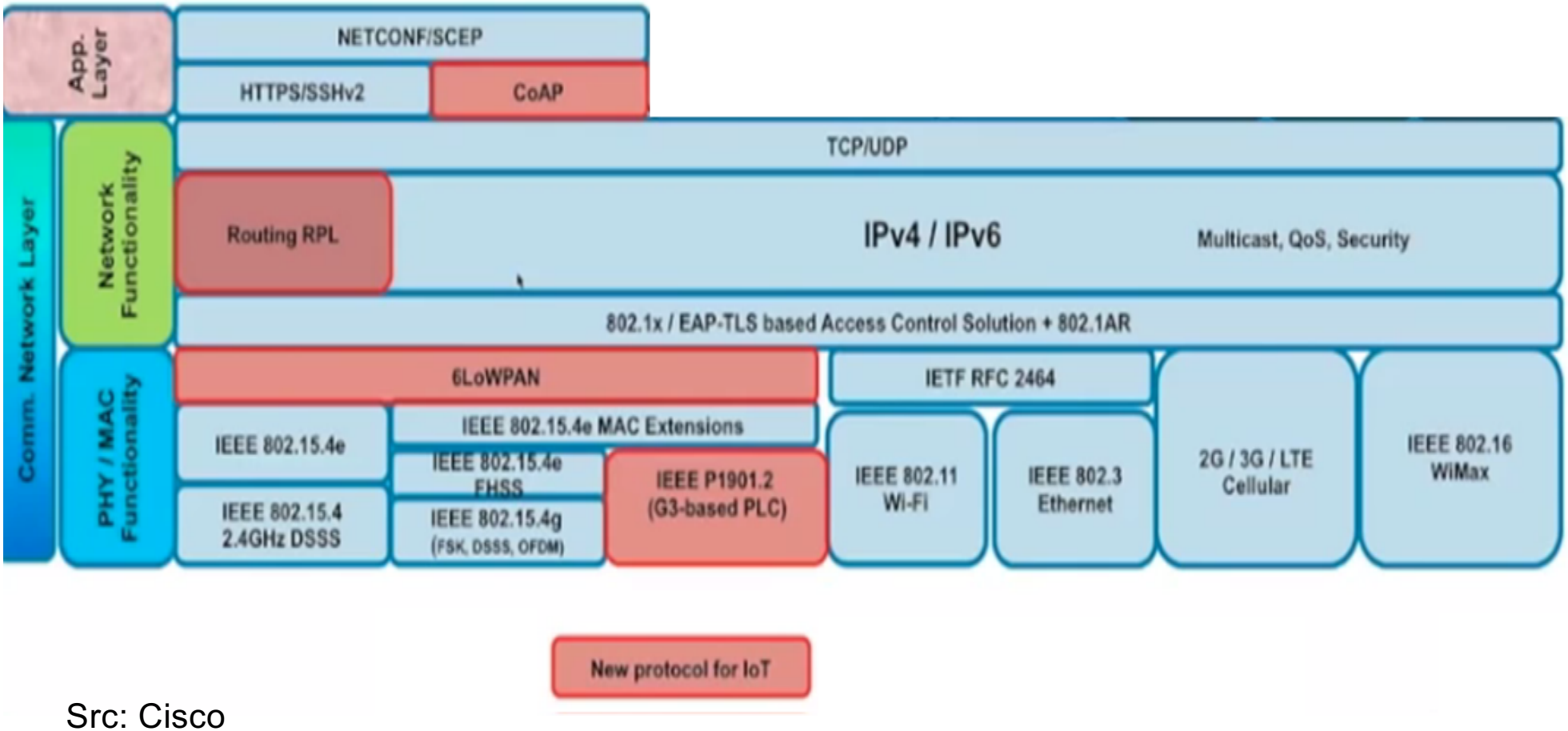
- Directed Acyclic Graph (DAG) Information Option (DIO) messages are broadcast to build the tree; includes a node's rank (its level), ETX (Expected **Transmission** Count), etc.
- ETX probe is sent periodically to probe neighboring ETX



RPL Summary

- RPL forms a Destination Oriented Directed Acyclic Graph (DODAG), with the root of the tree being the AP
- DODAG minimizes the cost to the root per Objective Function (using ETX for example)
- Directed Acyclic Graph (DAG) Information Option (DIO) messages are broadcast to build the tree; includes a node's rank (its level), ETX, etc.
- A node selects a parent based on the received DIO msgs and calculates its rank
- Destination Advertisement Option (DAO) msg sent periodically to notify parent about routes to downward nodes

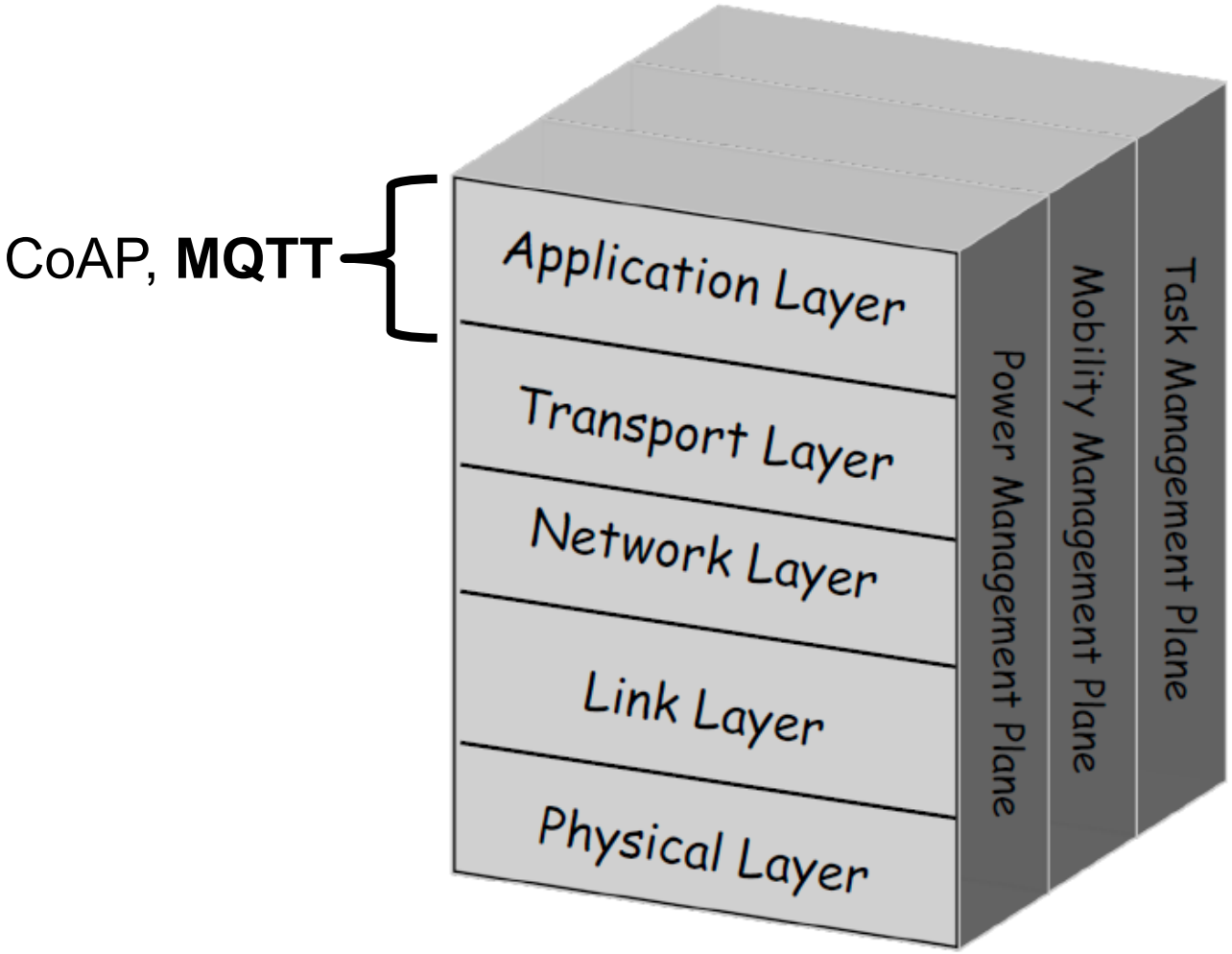
Conclusion: The New „Pieces“ for IoT



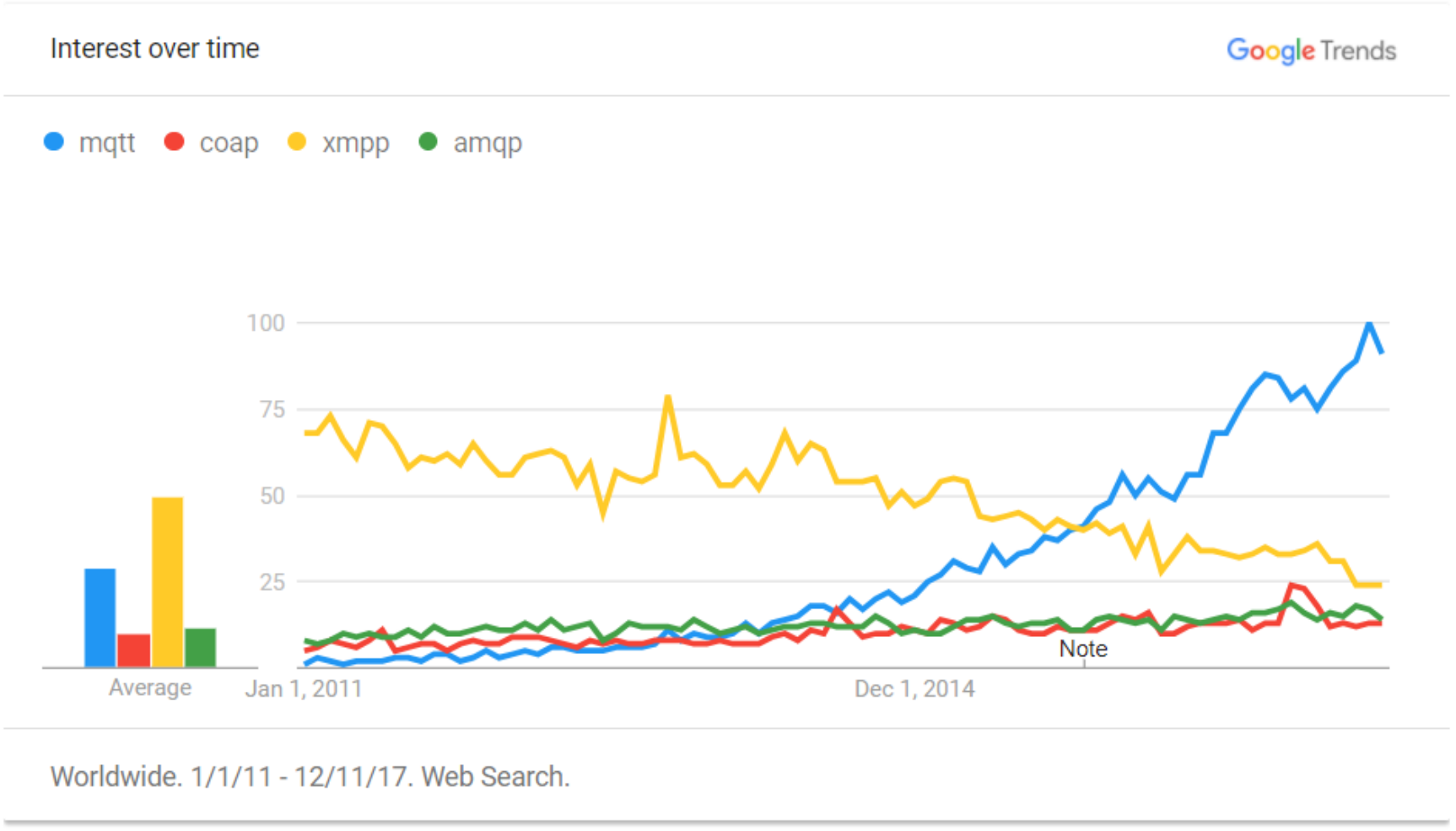
Src: Cisco

IoT Application Protocols: MQTT

The Protocol Stack



MQTT vs CoAP


Interest over time
Google Trends
mqtt
coap
xmpp
amqp
100
75
50
25
Note
Average
Jan 1, 2011
Dec 1, 2014
Worldwide. 1/1/11 - 12/11/17. Web Search.

Chapter Outline

- What is MQTT?
  - Message Queuing Telemetry Transport (MQTT)
- Design Objectives
- MQTT Protocol
  - Message Format
  - Quality of Service (QoS) Support
  - Message Sequences
- MQTT-SN
- Conclusions

MQ Telemetry Transport (MQTT)

MQTT

- MQTT was co-invented by IBM and Eurotech in 1999
- The current MQTT specification is available here:
  - www.mqtt.org

  - Abstract from the MQTT spec web site:

    The MQTT protocol is a **lightweight** **publish/subscribe** protocol flowing over TCP/IP for remote sensors and control devices through low bandwidth, unreliable or intermittent communications.

- In 2014, MQTT was adopted and published as **an official standard by OASIS** (published V3.1.1). The OASIS TC (Technical Committee) is tasked with the further development of MQTT.
  - 2018: MQTT 5 released

MQTT in a Nutshell

- In a nutshell

  A lightweight event and message oriented protocol allowing devices to **asynchronously** communicate **efficiently** across **constrained** networks **to remote systems**

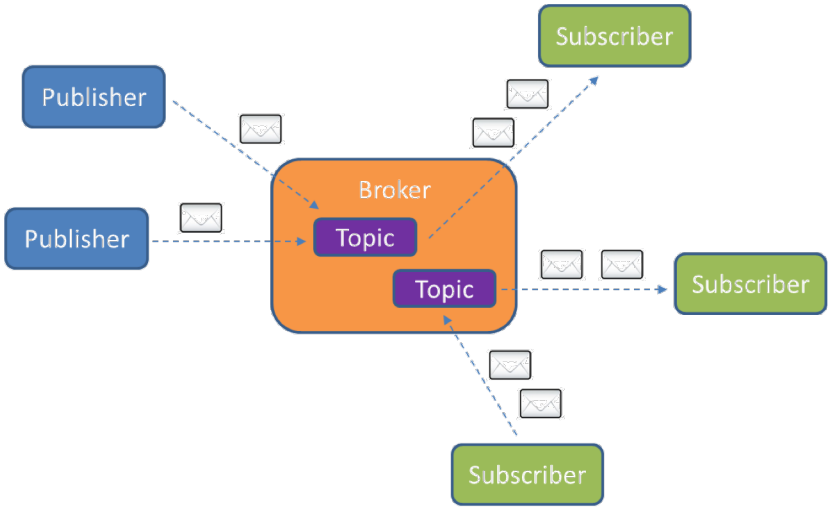
- Suited to
  - Constrained networks and devices
    - Low bandwidth
    - High latency
    - Unreliable
    - High cost (per byte)
  - Constrained devices in processing, storage and energy (8 Bit controllers upwards)
- Assumption
  - TCP/IP

Original Design Goals

- To provide **loose coupling** between devices and the processing systems, and between things that produce data and things that consume that data.
- To provide multiple deterministic message delivery Qualities of Service (QoS) to reflect **tradeoffs** between bandwidth, availability and delivery guarantees
- To support **large numbers of things** (Millions and more)
- To be **simple** for application developers and implementers of the protocol
- To have **open** specification for ease of adoption by device/thing vendors
- To be **industry-agnostic**

Publish Subscribe Messaging (One to Many)

A publish/subscribe messaging protocol allowing a message to be published once, and multiple consumers (applications/devices) to receive the message providing decoupling between the producers and the consumer(s).



- A producer sends (**publishes**) a message (**publication**) to a **topic** (subject)
- A consumer **subscribes** (makes a subscription) for messages on a topic (subject)
- A message server (**broker**) matches publications to subscriptions
  - If no matches, the message is discarded
  - If matches, the message is delivered to each matching subscriber/consumer

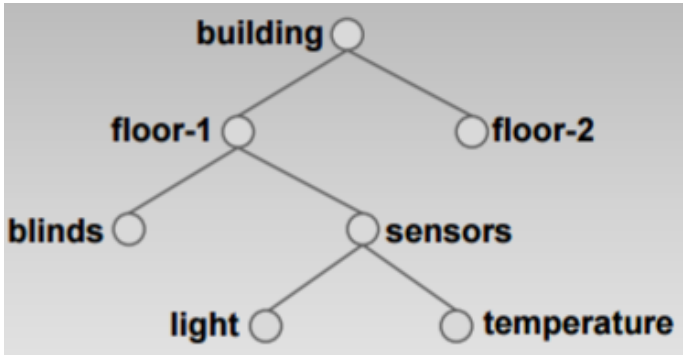
Topic Wildcards (1)

- Problem: Subscribers are often interested in a great number of topics. Individually subscribing to each named topic is time-consuming.
- MQTT solution: Topics can be hierarchically organized through wildcards with path-type topic strings and the wildcard characters
  - +: can appear anywhere in the topic string (Single-level wildcard)
  - #: must appear at the end of the string (Multi-level wildcard)
- Subscribers can subscribe for an entire sub-tree of topics thus receiving messages published to any of the sub-tree‘s nodes. Wildcards can not be used when publishing.
- A topic forms the namespace
  - Is hierarchical with each “sub topic” separated by a /
  - An example topic space
    - A house publishes information about itself on:
      - *<country>/<region>/<town>/<postcode>/<house>/energyConsumption*
      - *<country>/<region>/<town>/<postcode>/<house>/waterConsumption*
      - *<country>/<region>/<town>/<postcode>/<house>/solarEnergy*
      - *<country>/<region>/<town>/<postcode>/<house>/alarmState*

    And subscribes for control commands:
      - *<country>/<region>/<town>/<postcode>/<house>/thermostat/setTemp*

Topic Wildcards (2)

Example topic tree:



Topic string special character	Description
/	Topic level separator. Example: *building / floor-1 / sensors / temperature*
+	Single level wildcard. Matches one topic level. Examples: *building / floor-1 / +* (matches *building / floor-1 / blinds* and *building / floor-1 / sensors*) *building / + / sensors* (matches *building / floor-1 / sensors* and *building / floor-2 / sensors*)
#	Multi level wildcard. Matches multiple topic levels. Examples: *building / floor-1 / #* (matches all nodes under *building / floor-1*) *building / # / sensors* (invalid, '#' must be last character in topic string)

MQTT - Publish Subscribe Messaging (One to Many)

- A subscription can be durable or non durable
  - Durable:
    - Once a subscription is in place a broker will forward matching messages to the subscriber:
      - Immediately if the subscriber is connected
      - If the subscriber is not connected messages are stored on the server/broker until the next time the subscriber connects
  - Non-durable/transient: The subscription lifetime is the same as the time the subscriber is connected to the server / broker
- A publication may be retained
  - A publisher can mark a publication as retained
  - The broker / server remembers the last known good message of a retained topic
  - The broker / server gives the last known good message to new subscribers
    - i.e. the new subscriber does not have to wait for a publisher to publish a message in order to receive its first message

MQTT: The Protocol

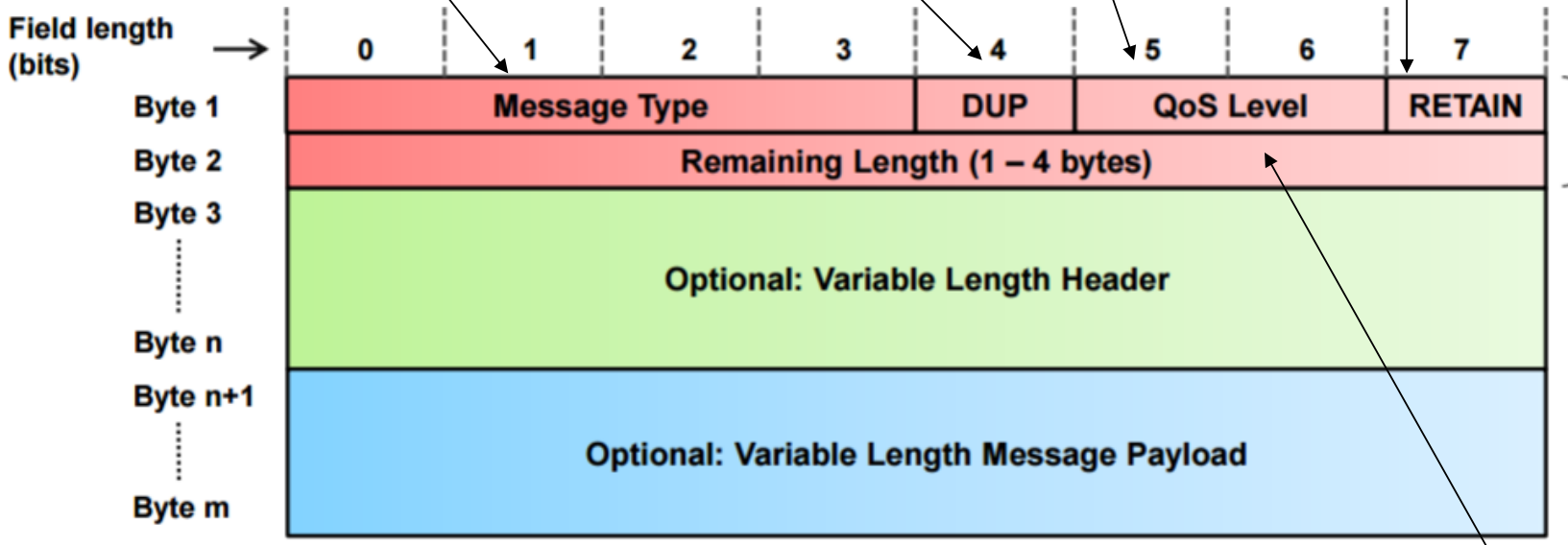
Message Format

0: Reserved	8: SUBSCRIBE
1: CONNECT	9: SUBACK
2: CONNACK	10: UNSUBSCRIBE
3: PUBLISH	11: UNSUBACK
4: PUBACK	12: PINGREQ
5: PUBREC	13: PINGRESP
6: PUBREL	14: DISCONNECT
7: PUBCOMP	15: Reserved

Duplicate message flag. Indicates to the receiver that this message may have already been received.
1: Client or server (broker) re-delivers a PUBLISH, PUBREL, SUBSCRIBE or UNSUBSCRIBE message (duplicate message).

Indicates the level of delivery assurance of a PUBLISH message.
0: At-most-once delivery, no guarantees, «Fire and Forget».
1: At-least-once delivery, acknowledged delivery.

1: Instructs the server to retain the last received PUBLISH message and deliver it as a first message to new subscriptions.

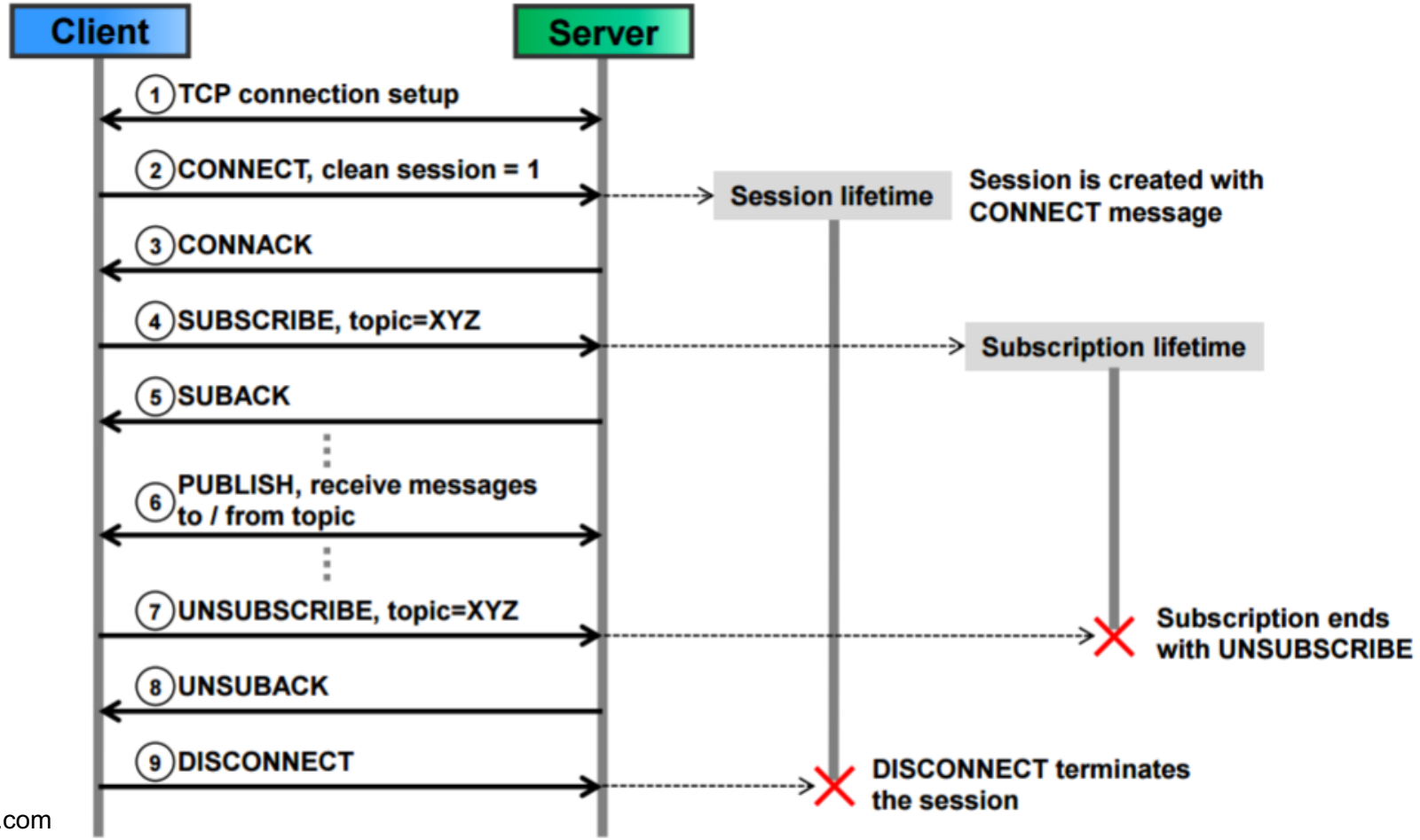


Indicates the number of remaining bytes in the message, i.e. the length of the (optional) variable length header and (optional) payload.

Src: indigoo.com

CONNECT and SUBSCRIBE msg Sequence (1)

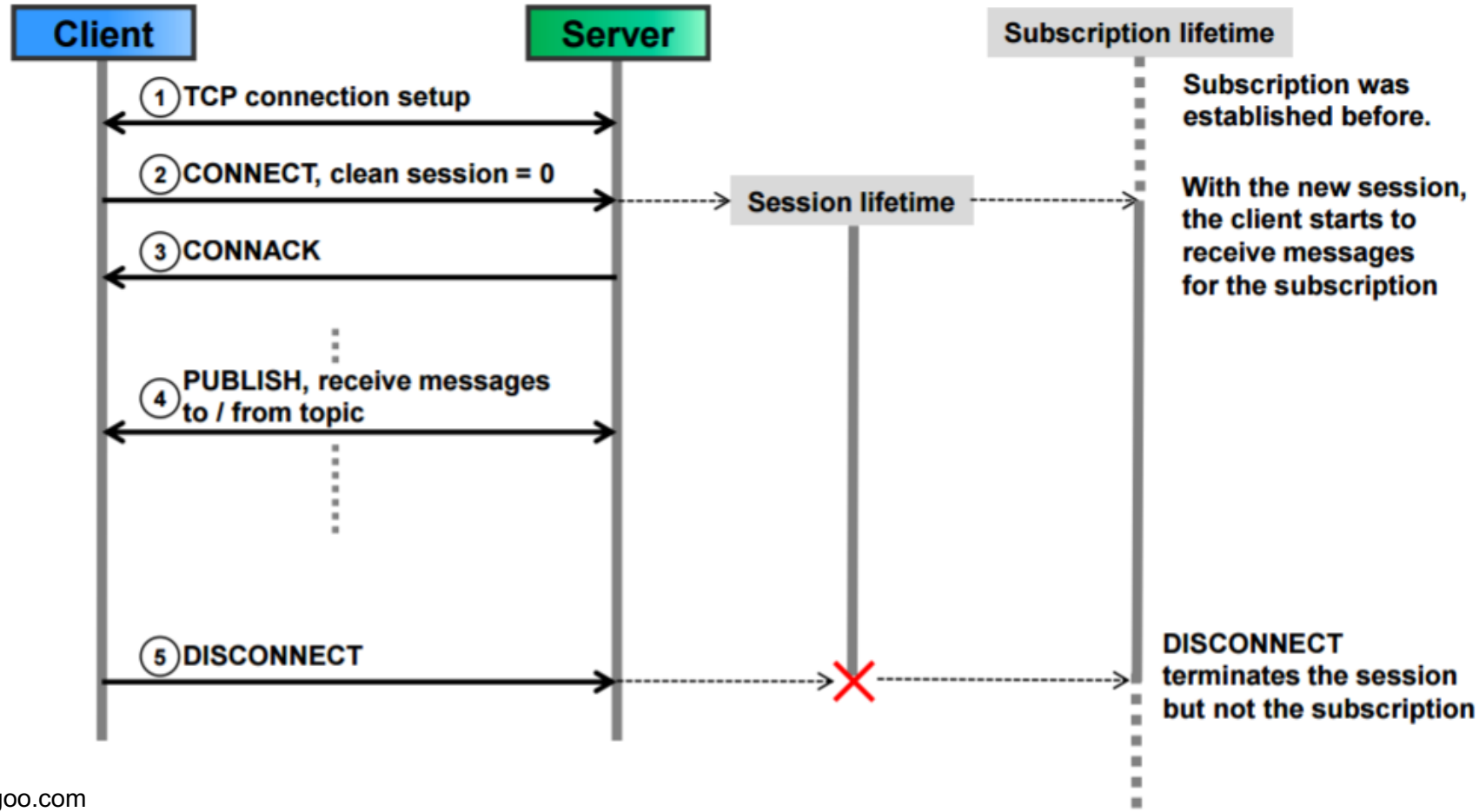
- Case 1: Session and subscription setup with *clean session flag* = *1* („transient“ subscription)



Src: indigoo.com

CONNECT and SUBSCRIBE msg Sequence (2)

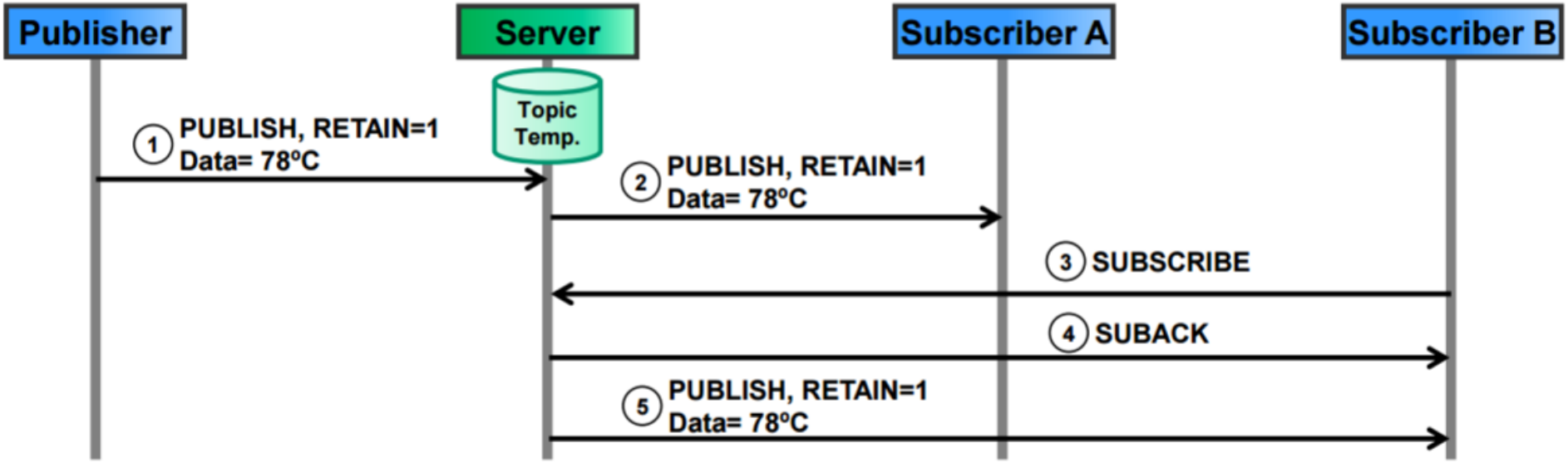
- Case 2: Session and subscription setup with *clean session flag* = *0* („durable subscription“)



Src: indigoo.com

RETAIN (Keep Last Message)

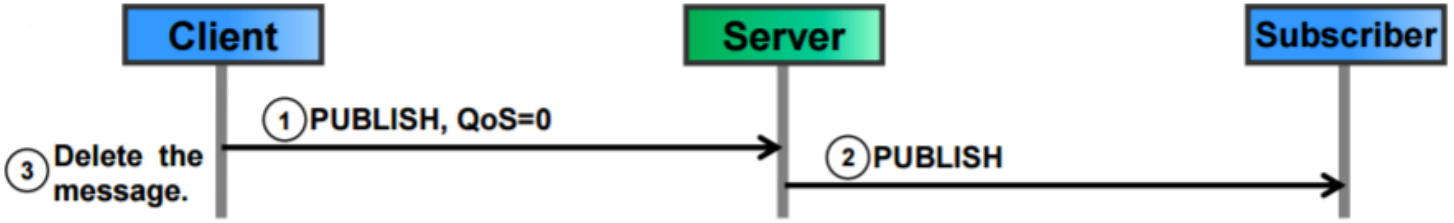
- RETAIN=1 in a PUBLISH msg → instructs the server to keep the message for this topic. When a new client subscribes to this topic, the server sends the retained msg → Subscribers receive the last know good value.



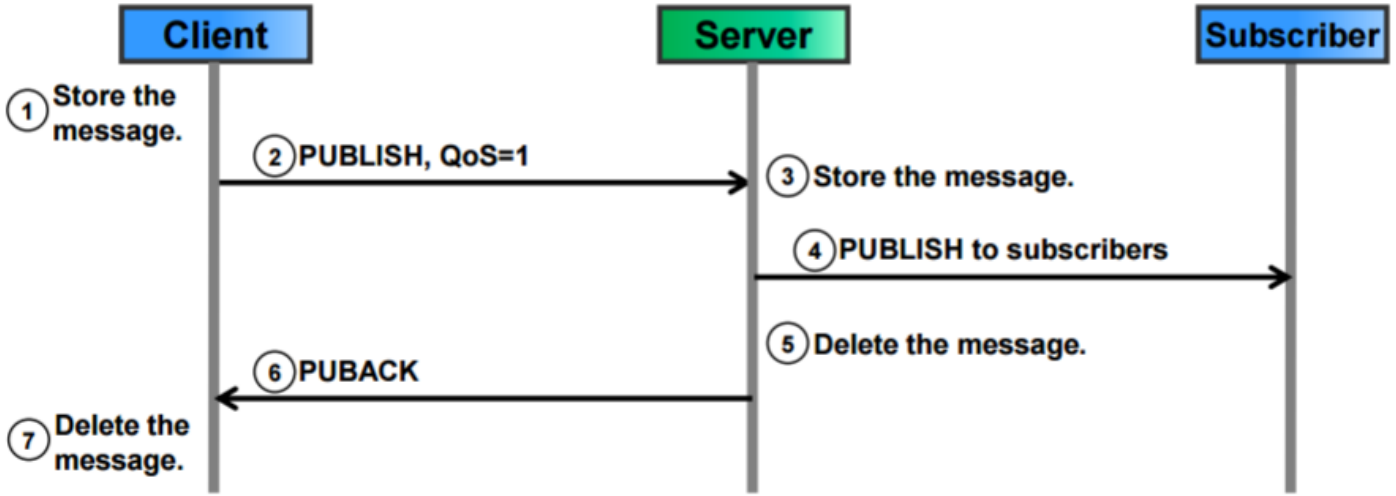
Src: indigoo.com

PUBLISH Message Flows (1)

- QoS Level 0: A msg is delivered with **at-most-once** delivery semantics („fire-and-forget“).



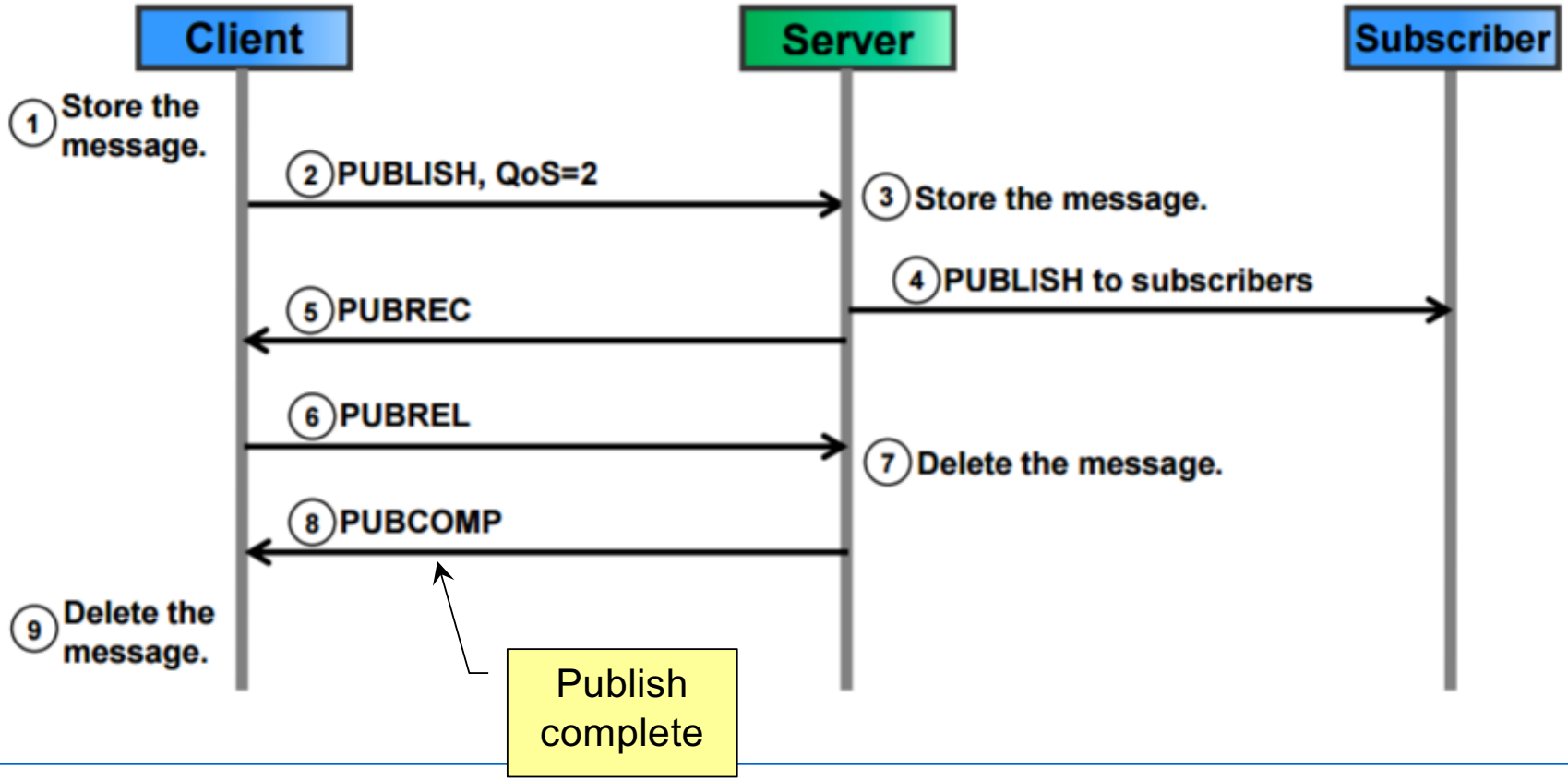
- QoS Level 1: Affords **at-least-once** delivery semantics. If the client does not receive the PUBACK in time, it re-sends the msg.



Src: indigoo.com

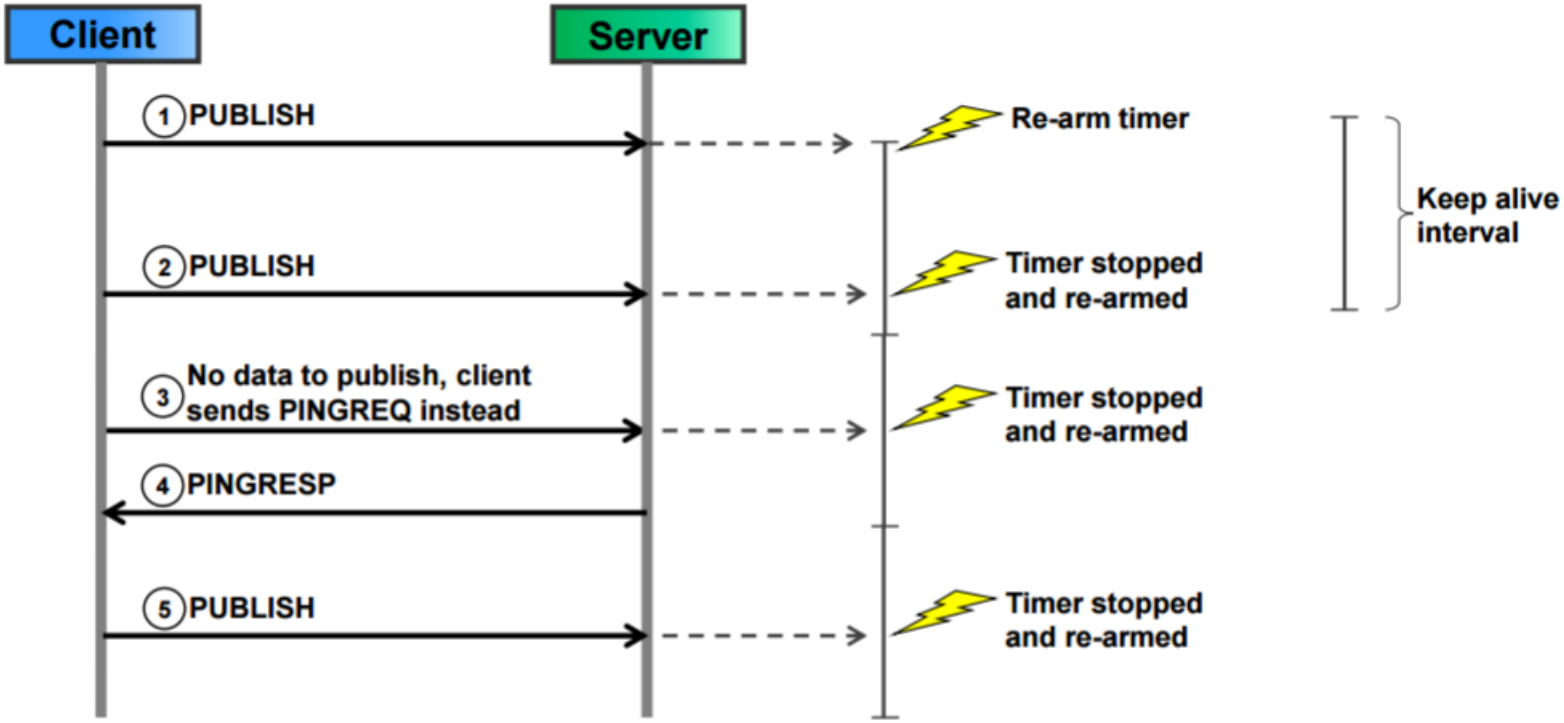
PUBLISH Message Flows (2)

- QoS level 2: Affords the highest quality delivery semantics **exactly-once**, but comes with the cost of additional control messages.



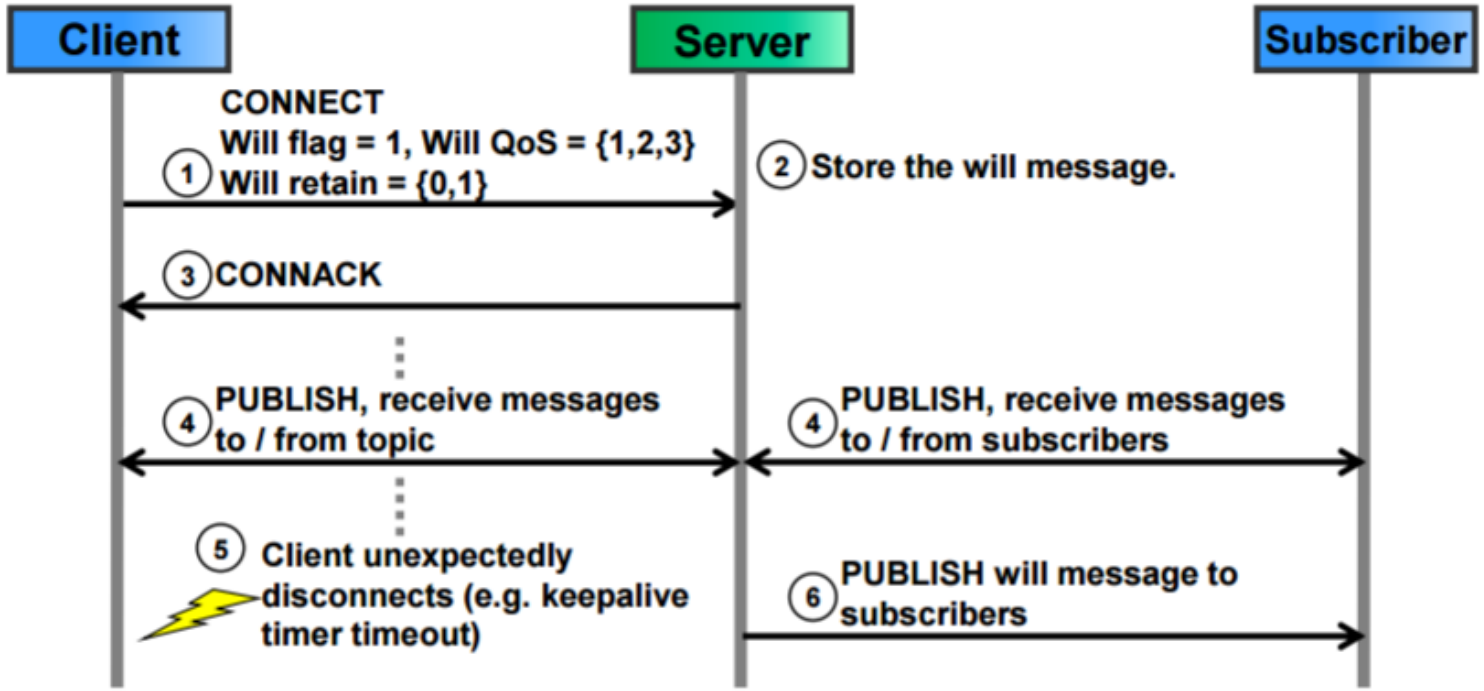
Keepalive and PINGREQ

- Keepalive timer: Defines the max allowable time interval between client msgs. The timer is used by the server to check client‘s connection status. After 1,5* keepalive-time elapsed, the server disconnects the client. Typical value for keepalive timer are a couple of minutes.
- Breath of live with PINGREQ: In the absence of data to be sent, the client sends a PINGREQ message instead.



MQTT will Message

- Problem: In case of an unexpected client disconnect, depending applications (subscribers) do not receive any notification of the client‘s departure.
- MQTT solution: Client can specify a **will msg** along with a *will QoS* and *will retain* in the CONNECT msg payload. If the client unexpectedly disconnects, the server sends the will msg on behalf of the client to all subscribers („**last will**“).



Suits Constrained Networks & Devices

- Designed for constrained networks:
  - Protocol compressed into **bit-wise** headers and variable length fields
  - Smallest possible packet size is 2 bytes
  - Tiny **footprint** MQTT client (and server) libraries, e.g., C/C++ client lib in 30Kb and a Java lib in 64Kb
  - Asynchronous bidirectional "**push**" delivery of messages to apps **(no polling)**
    - Client to server/cloud and server/cloud to client
  - Supports always-connected and **sometimes-connected** models
  - Provides Session awareness
    - Configurable keep alive providing granular session awareness
    - "Last will" enable apps to know when a client goes offline abnormally
  - Typically utilises TCP based networks e.g. Websokets
  - Tested on many networks – vsat, gprs, 2G….
- Provides multiple deterministic message delivery QoS.
  QoS maintained over fragile network even if connection breaks
  - 0 – message delivered at most once.
  - 1 – message will be delivered but may be duplicated
  - 2 – once and once only delivery

MQTT – Further Properties

- Client simple to develop (spec about 70 pages)
- Payload agnostic
  - no data types
  - no metadata
  - any data format (text, binary, JSON, XML, BSON, ProtoBuf, ...)
  - peer must agree on serialization/deserialization
- Assumptions
  - TCP/IP stack
  - Persistent TCP connections
  - Clean and persistent sessions

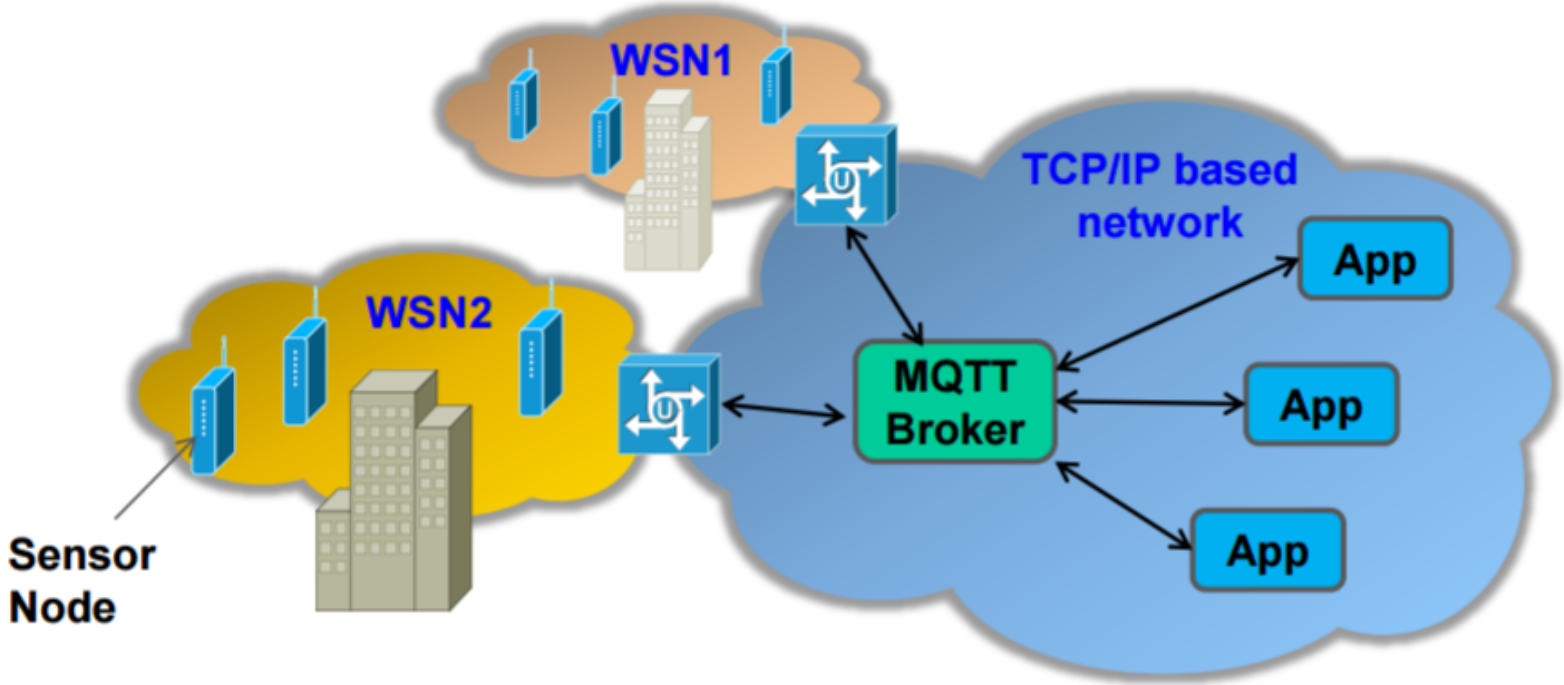
MQTT for Sensor Networks
(MQTT-SN renamed from MQTT-S)

MQTT-SN

- MQTT is a connectivity protocol that requires TCP/IP.
- MQTT-SN is designed to be as close as possible to MQTT, but is adapted to the peculiarities of a wireless communication environment such as low bandwidth, high link failures, short message length, etc
  - It does not require TCP/IP

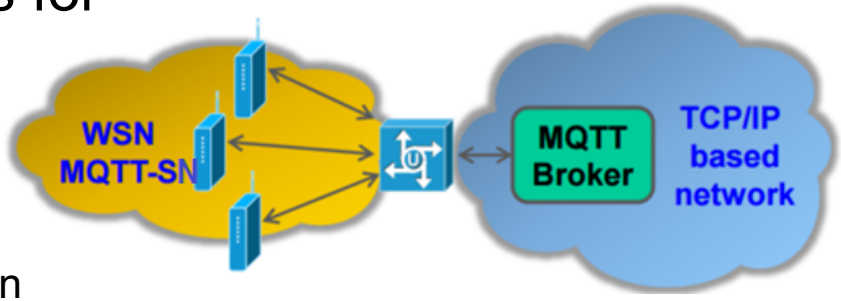
MQTT-SN (2)

- WSNs (Wireless Sensor Networks) usually do not have TCP/IP as transport layer. They have their own protocol stack such as ZigBee on top of IEEE 802.15.4 MAC layer. Thus, MQTT which is based on TCP/IP cannot be directly run on WSNs. WSNs are connected to traditional TCP/IP networks through gateway devices.
- MQTT-SN is an extension of MQTT for WSNs.

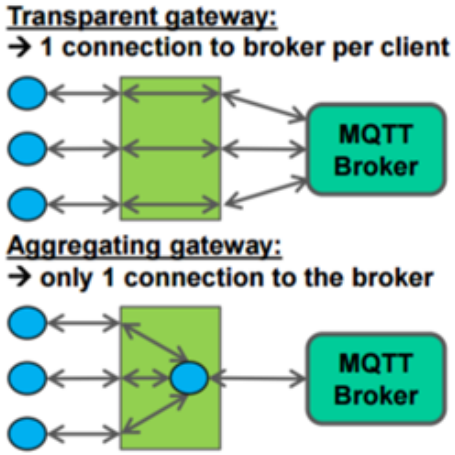


MQTT-SN (3)

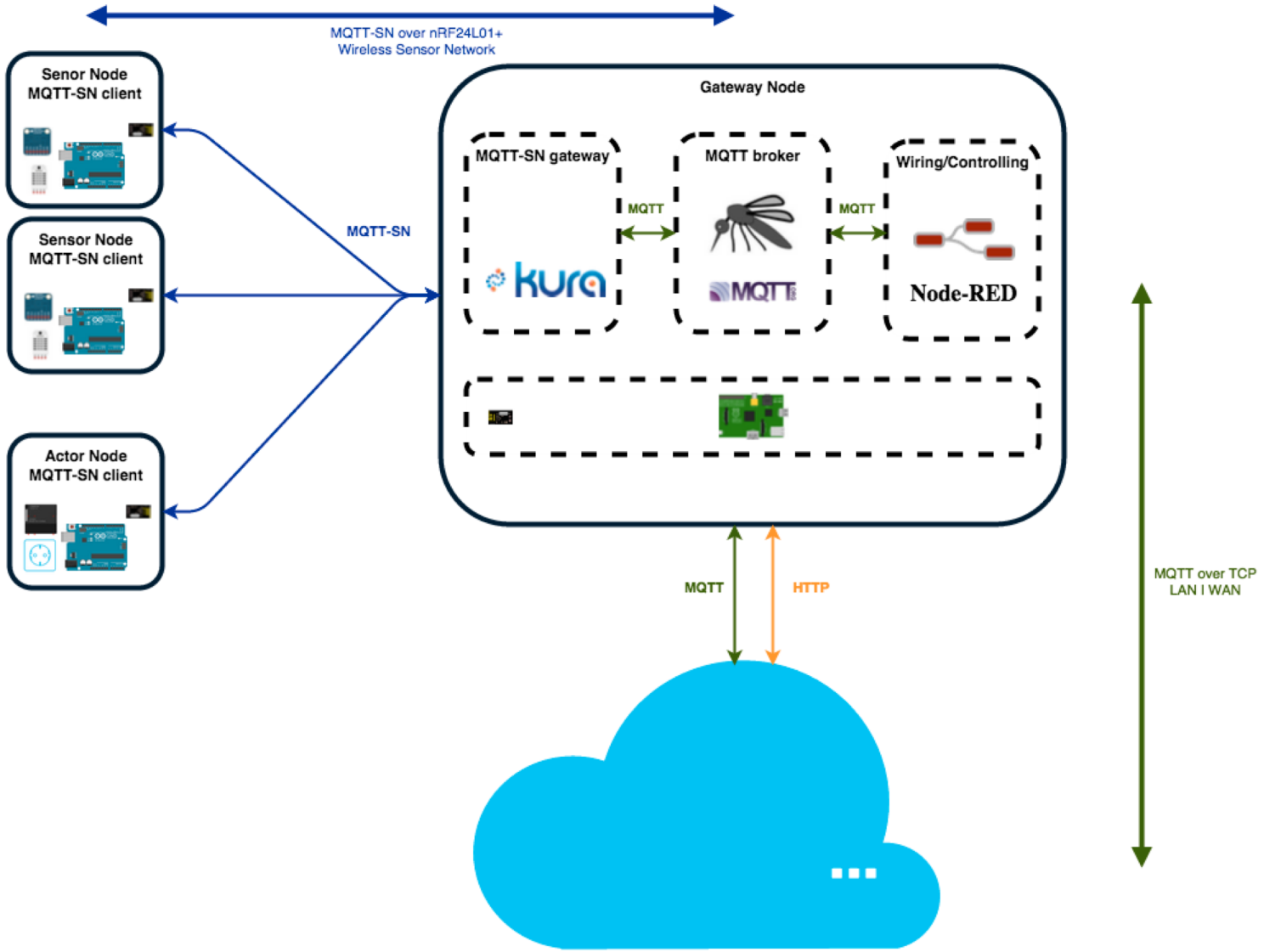
MQTT-SN is a largely based on MQTT, but implements some important optimizations for WSN:



- Topic string replaced by a topic ID (fewer bytes necessary)
- Predefined topic IDs that do not require a registration
- Discovery procedure for clients to find brokers (no need to statically config broker address)
- Persistent **will message** (in addition to persistent subscriptions)
- Off-line keepalive supporting sleeping clients (will receive buffered messages from the server once they wake up)
- MQTT-SN gateways (transparent or aggregating) connect MQTT-S domains (WSNs) with MQTT domains (traditional TCP/IP based networks).



Example of MQTT-SN



Src: http://openiotchallenge.tumblr.com/post/114361695760/project-wrap-up

MQTT: Eclipse Support

MQTT Brokers

- HiveMQ introduction to MQTT (invited lecture)
  - Commercial, enterprise use, highly scalable

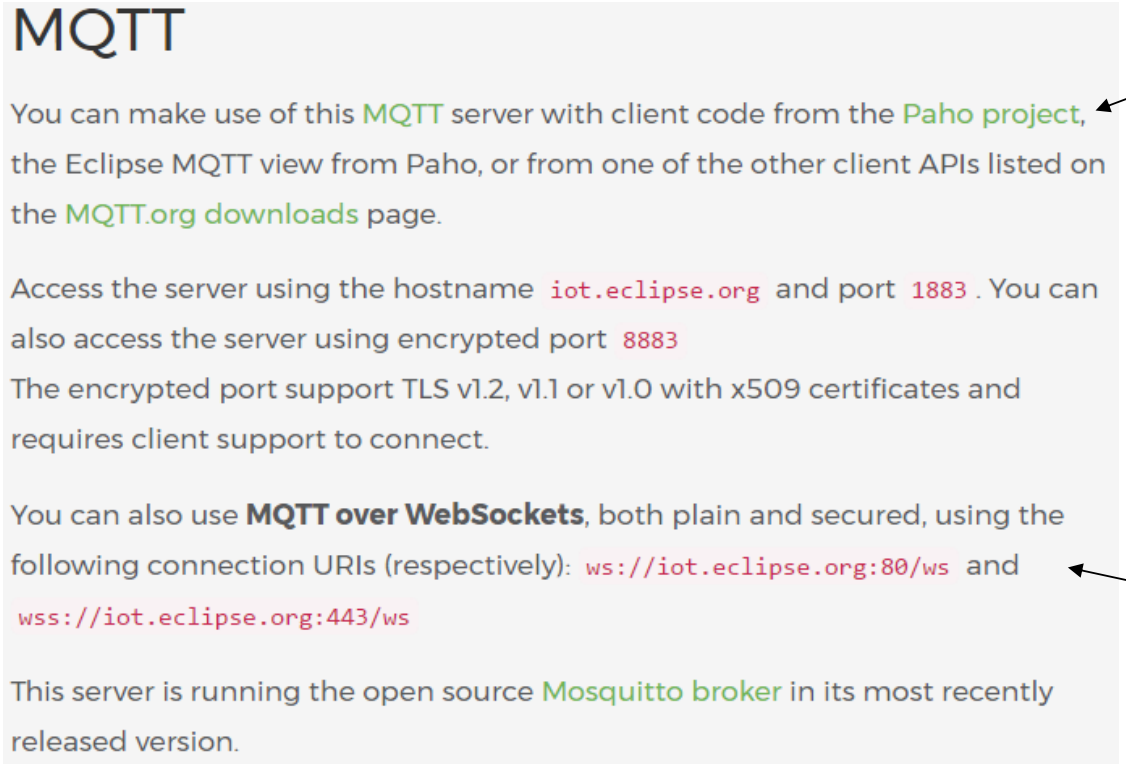
Mosquitto on Raspberry Pi

- Mosquitto
  - Open source
- CloudMQTT

- Ardulink MQTT
  - Java based
- Etc.

Eclipse MQTT Support

- See getting started under: https://iot.eclipse.org/getting-started



MQTT

You can make use of this MQTT server with client code from the Paho project, the Eclipse MQTT view from Paho, or from one of the other client APIs listed on the MQTT.org downloads page.

Access the server using the hostname `iot.eclipse.org` and port `1883`. You can also access the server using encrypted port `8883`
The encrypted port support TLS v1.2, v1.1 or v1.0 with x509 certificates and requires client support to connect.

You can also use **MQTT over WebSockets**, both plain and secured, using the following connection URIs (respectively): `ws://iot.eclipse.org:80/ws` and `wss://iot.eclipse.org:443/ws`

This server is running the open source Mosquitto broker in its most recently released version.

MQTT over TCP

- Client IDs for stateful MQTT broker sessions
- Standard ports: **1883 for TCP, 8883 for TLS**
- Configurable mapping of public and private MQTT ports

MQTT over Websockets

- Cookies instead of Client IDs for stateful MQTT broker sessions
- Name mapping

MQTT @ Eclipse IoT

- Paho (MQTT und MQTT-S) for devices
- Mosquitto MQTT broker
- Kura (MQTT Application framework)

Conclusions (1)

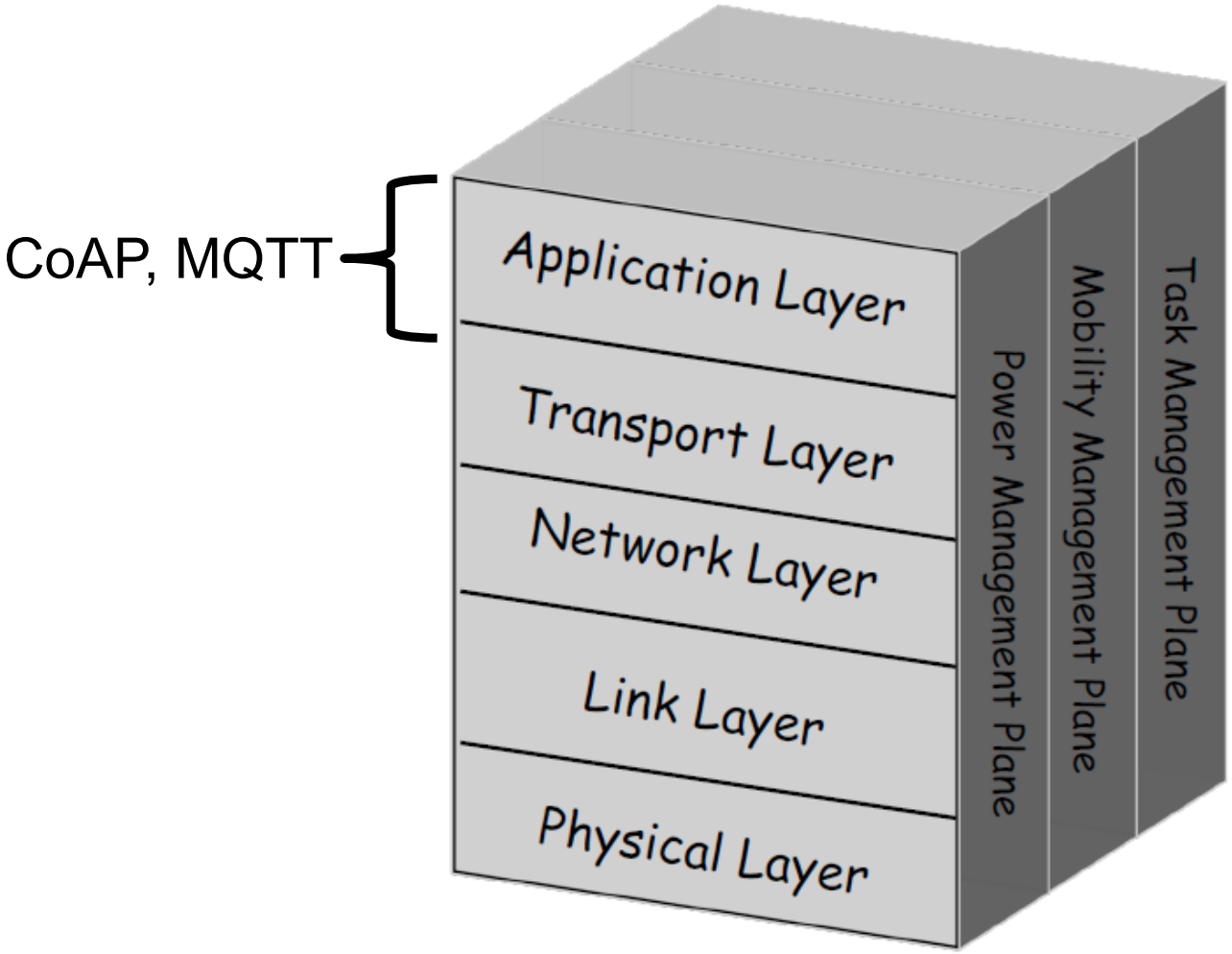
+ Telemetry
  - born for telemetry with “publish/subscribe” pattern
  - no flow control for a lot of data at high rate
  - QoS (at most once, at least once, exactly once)
+ Notification
  - born for notification with “publish/subscribe” pattern
  - no flow control for a lot of data at high rate
  - QoS (at most once, at least once, exactly once)
- Inquiry
  - no built in response path support
  - needed to define a response topic pattern (over)
    - group_id, device_id, req_id in custom format (as payload)

Conclusions (2)

- Command
  - no built in result command path support
  - needed to define a result topic pattern (over)
    - addressing result (req_id) in custom payload
  - if device is offline
    - no TTL (Time To Live) for command
    - old command could be delivered (if “retain” flag)
    - new command could be lost (if not “retain” flag)
    - commands are enqueued only if not “clean session”
- Security
  - SSL/TLS
  - username/password on connection
  - encryption only payload
- Scalability?

IoT Application Protocols: CoAP: The Web of Things Protocol

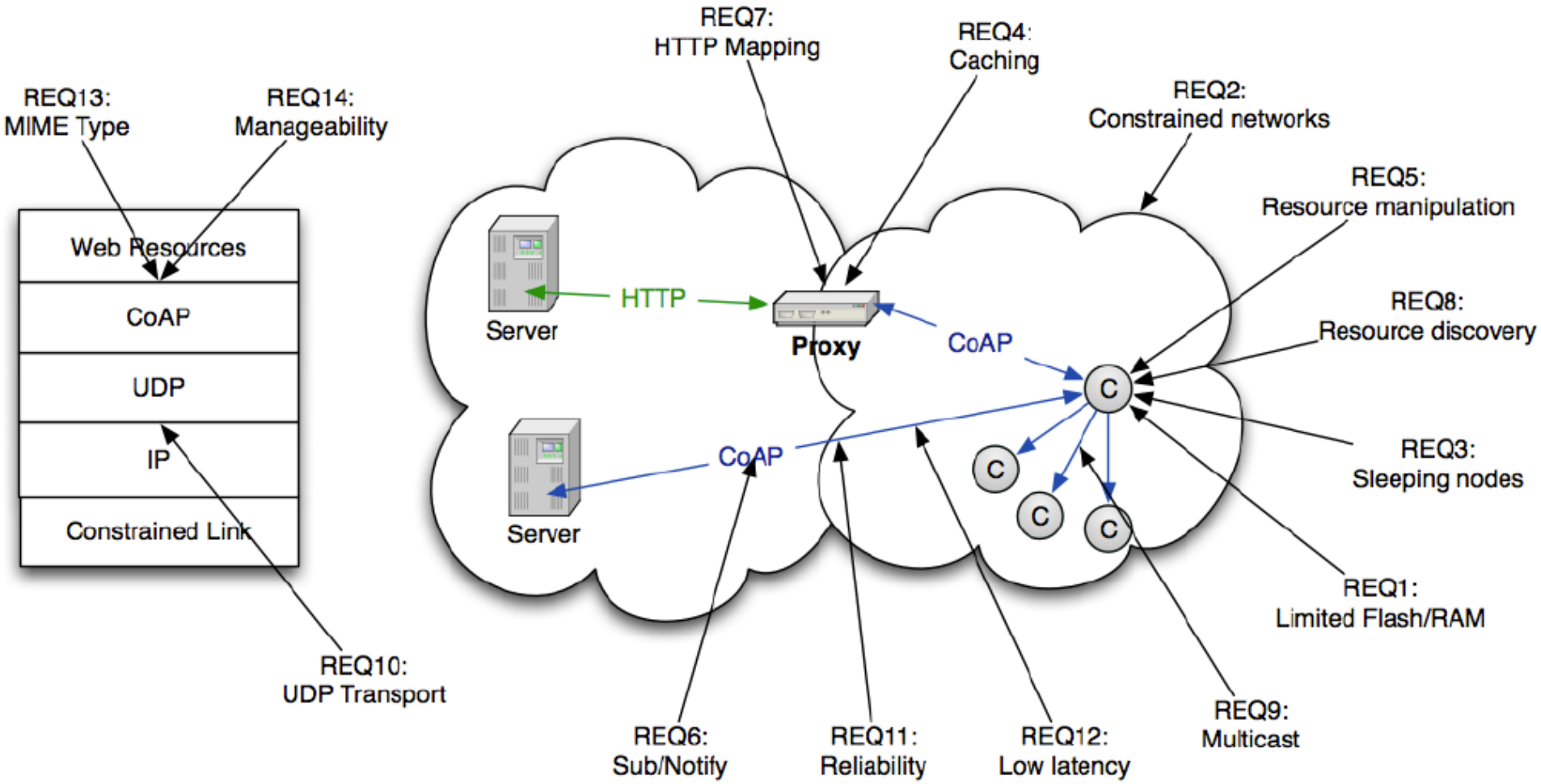
The Protocol Stack



Chapter Outline

- What is CoAP?
- HTTP vs. CoAP
- CoAP Protocol
  - Methods: GET, PUT, POST
  - Observation
  - Resource Discovery
  - Proxying and Caching
- Eclipse Support
- Conclusions

CoAP – Design Requirements



Src: draft-shelby-core-coap-req

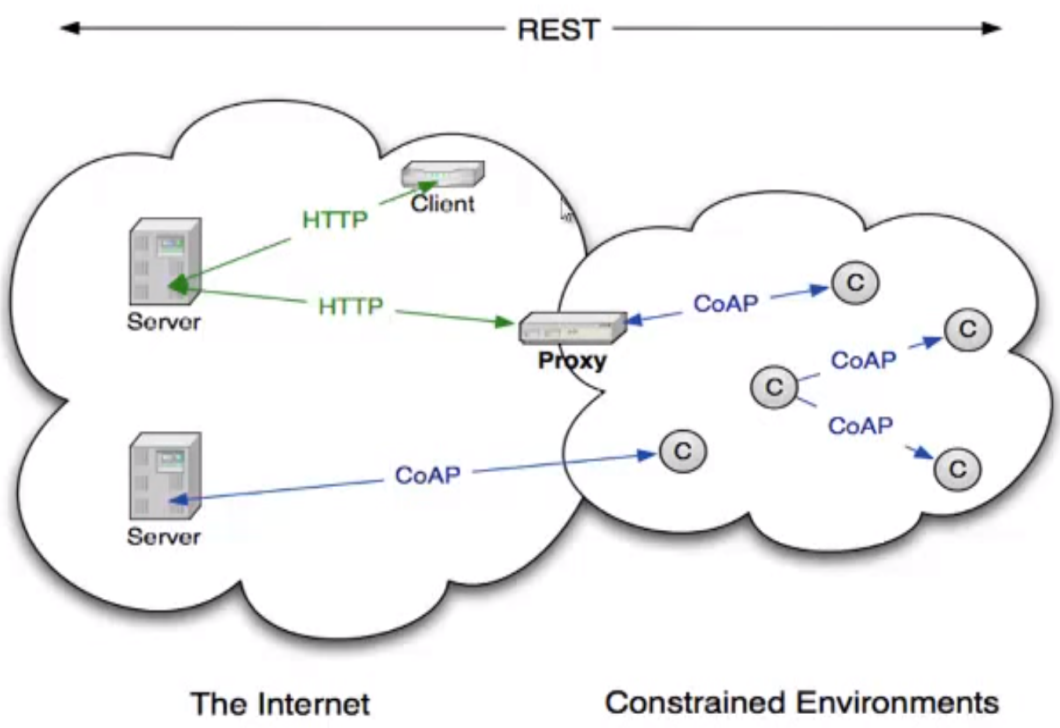
(Helper slide) REST

- Representational State Transfer (REST) is a software architectural style for Web client-server
- Resources are represented as URL:
  - “example.com/profile/john”
  - “example.com/domain/sensor3/temp1“
- Resources can be retrieved and manipulated using methods:
  - GET, POST, PUT, DELETE
  - Example protocol: HTTP

CoAP: Constrained Application Protocol

- Open IETF Standard
- Inspired by HTTP
  - Methods: `GET`, `POST`, `PUT`, `DELETE`
  - But **UDP** binding
- Small, simple **4-Byte header**
- UDP, SMS, (TCP) support
- Strong DTLS Security

→ "**Web of Things**"



CoAP Architecture

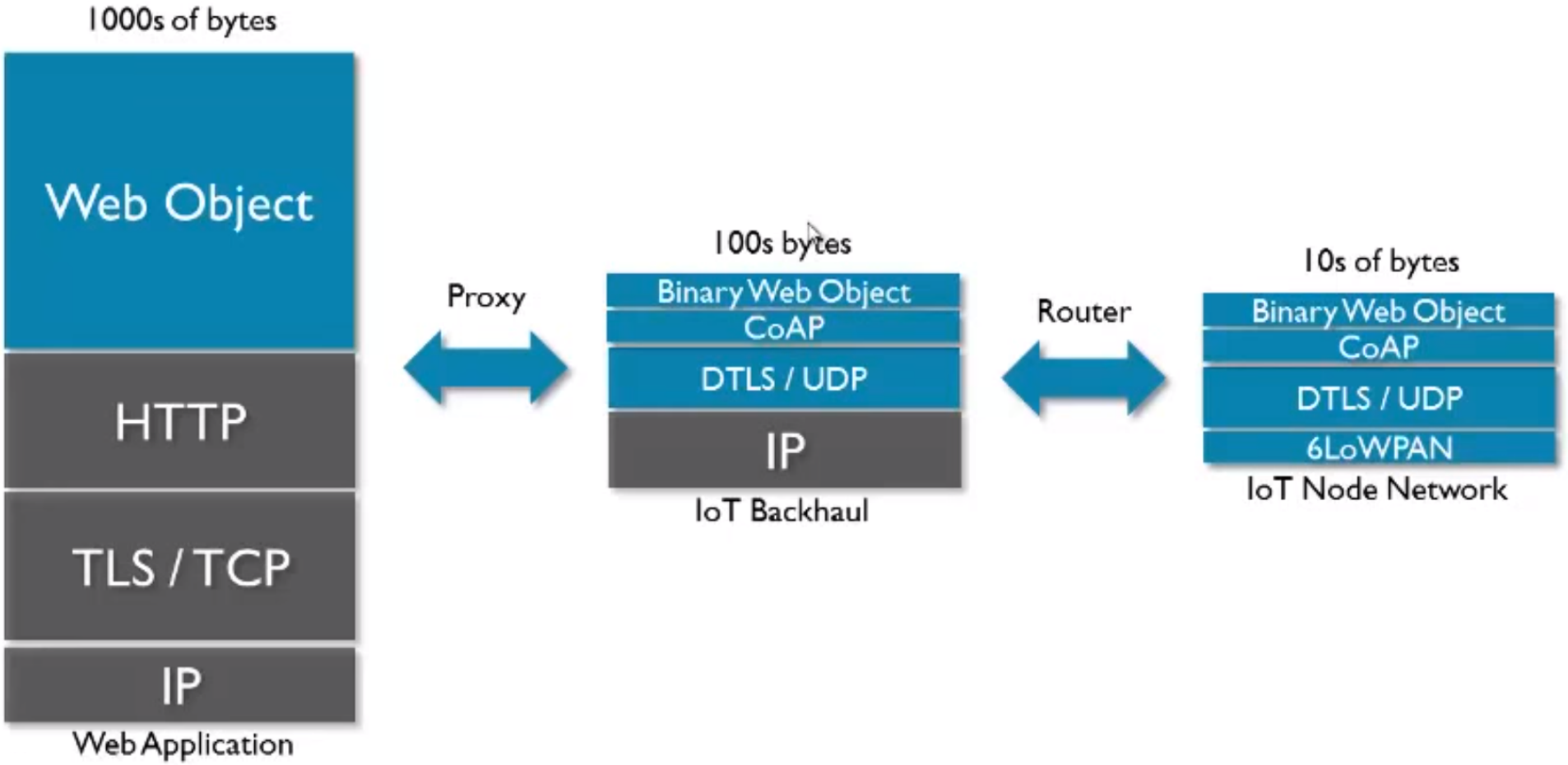
Src: Presentation Shelby: CoAP: the IoT protocol

CoAP: Constrained Application Protocol

- RESTful Web services for networked embedded devices
  - Idealized architectural style of the Web
  - Usually implemented with HTTP

- Central mechanisms
  - Reliable transport («confirmable»)
  - Group communication (IP multicast)
  - Push notifications («observing»)
  - Resource discovery («CoAP link format»)
  - Larger data transport («blockwise transfers»)

From Web Applications to IoT Nodes



Src:ARM

Message Header (4 Bytes)

Ver - CoAP version (2-bit)
T - Transaction type (2-bit)
OC - Option count (4-bit)
Code - Request method (1-10) or response code (40-255) (8-bit). E.g:
- GET: 1
- POST: 2
- PUT: 3
- DELETE: 4

Transaction ID - Unique identifier for matching response (16-bit)

Transaction/Message ID:

- CON: Confirmable message (=00)
- NON: Non-confirmable message (=01)
- ACK: Acknowledgment message (=10)
- RST: Reset message (=11)
  - Piggy-backed
  - Seperate

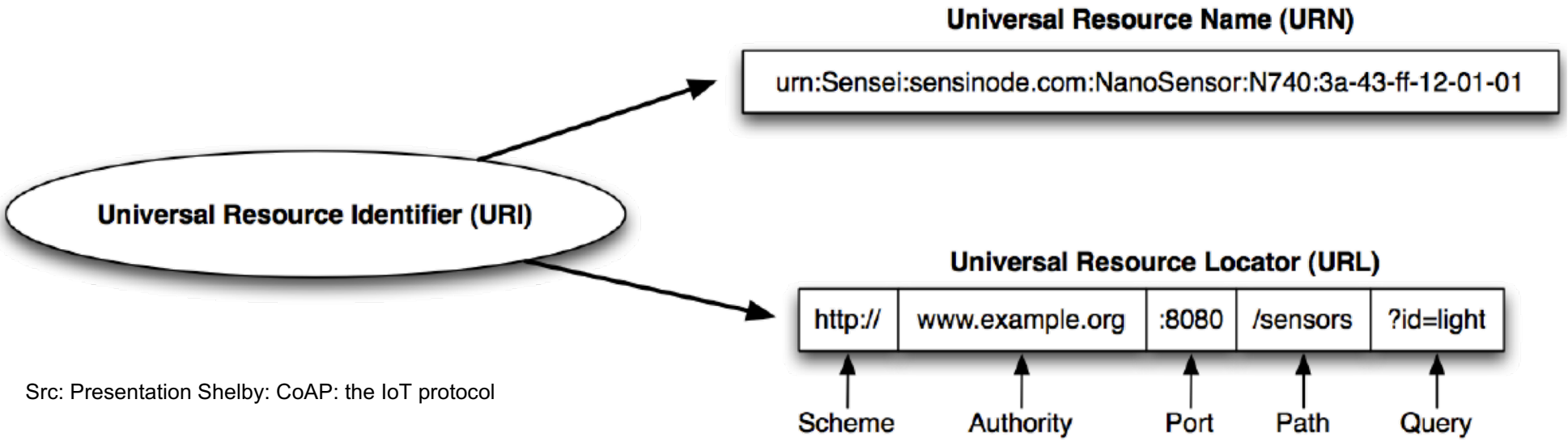
Requests and Responses

- Request Methods
  - GET uri
    - Retrieve (resource identified by) uri
  - PUT uri [parameter]
    - Update uri
  - DELETE uri
    - Delete uri
  - POST uri [parameter]
    - Create new resource under uri
- GET is safe (does not modify resources)
- GET, PUT and DELETE are (must be) idempotent (can be called many times without different outcomes)

- Responses
  - Class 2: success
    - 2.01: created, 2.02: deleted, 2.03: valid, 2.04: changed, 2.05: content
  - Class 4: client error
    - 4.00 bad request, 4.01: unauthorized, 4.02: bad option, 4.03: forbidden, 4.04: not found, etc
  - Class 5: server error
    - 5.00: internal server error, 5.01: not implemented, 5.02: bad gateway, 5.03: service unavailable, etc

Web Naming

- URI is
  - Name (identity, e.g., URN) +
  - Locator (address, e.g., URL)

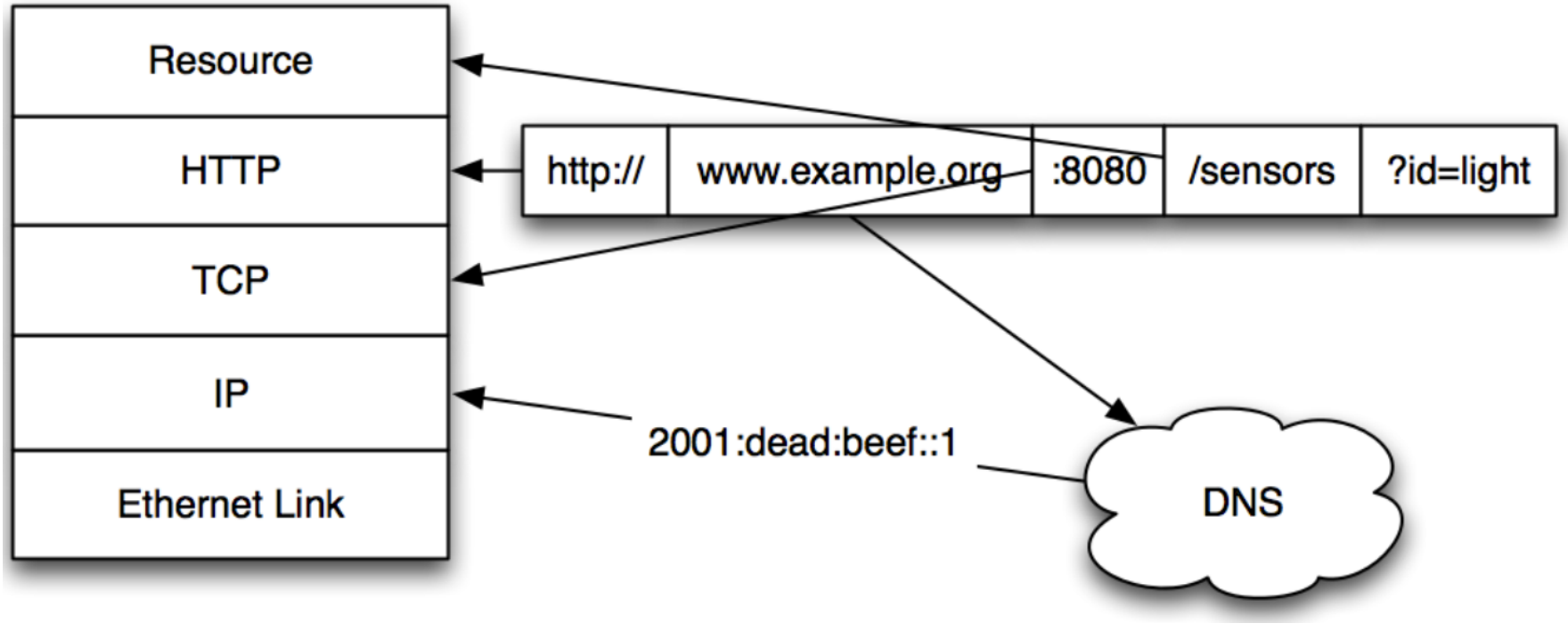


Src: Presentation Shelby: CoAP: the IoT protocol

- coap://example.org:5683/sensors/light1 (**default port** of 5683)
- coaps://example.org:5684/sensors/light1 (**default port** of 5684)

URL Resolution

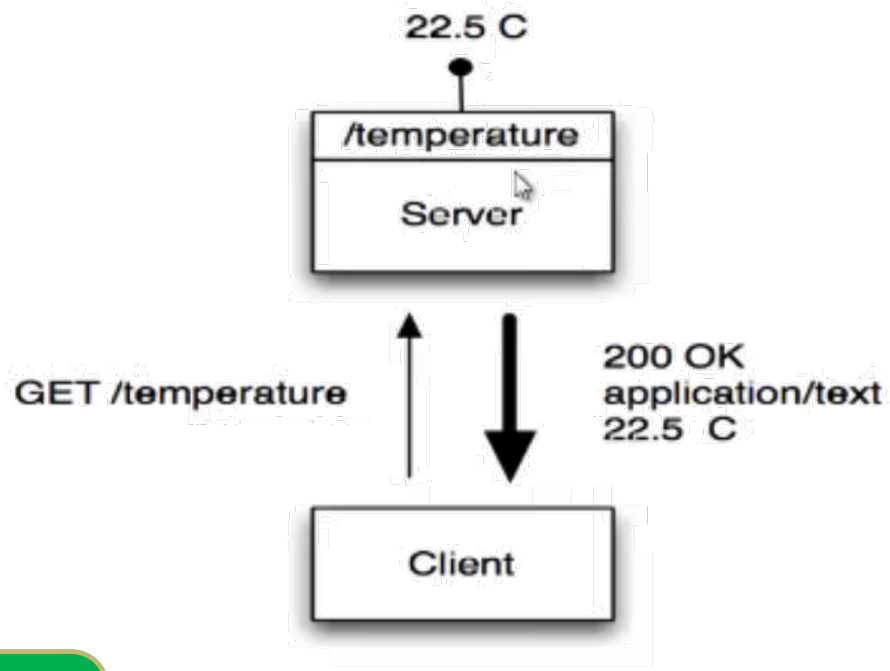
coap://example.org:5683/sensors/light1

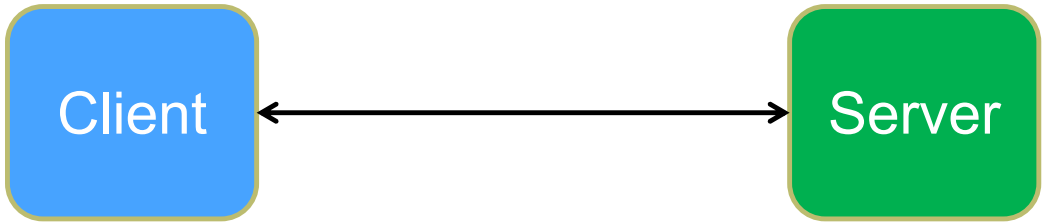


Architecture : HTTP & CoAP

- Client/Server
  - Request/response
    - HTTP : synchronous
    - CoAP : (also) asynchronous
- HTTP is ASCII based
- CoAP is binary based

Example:





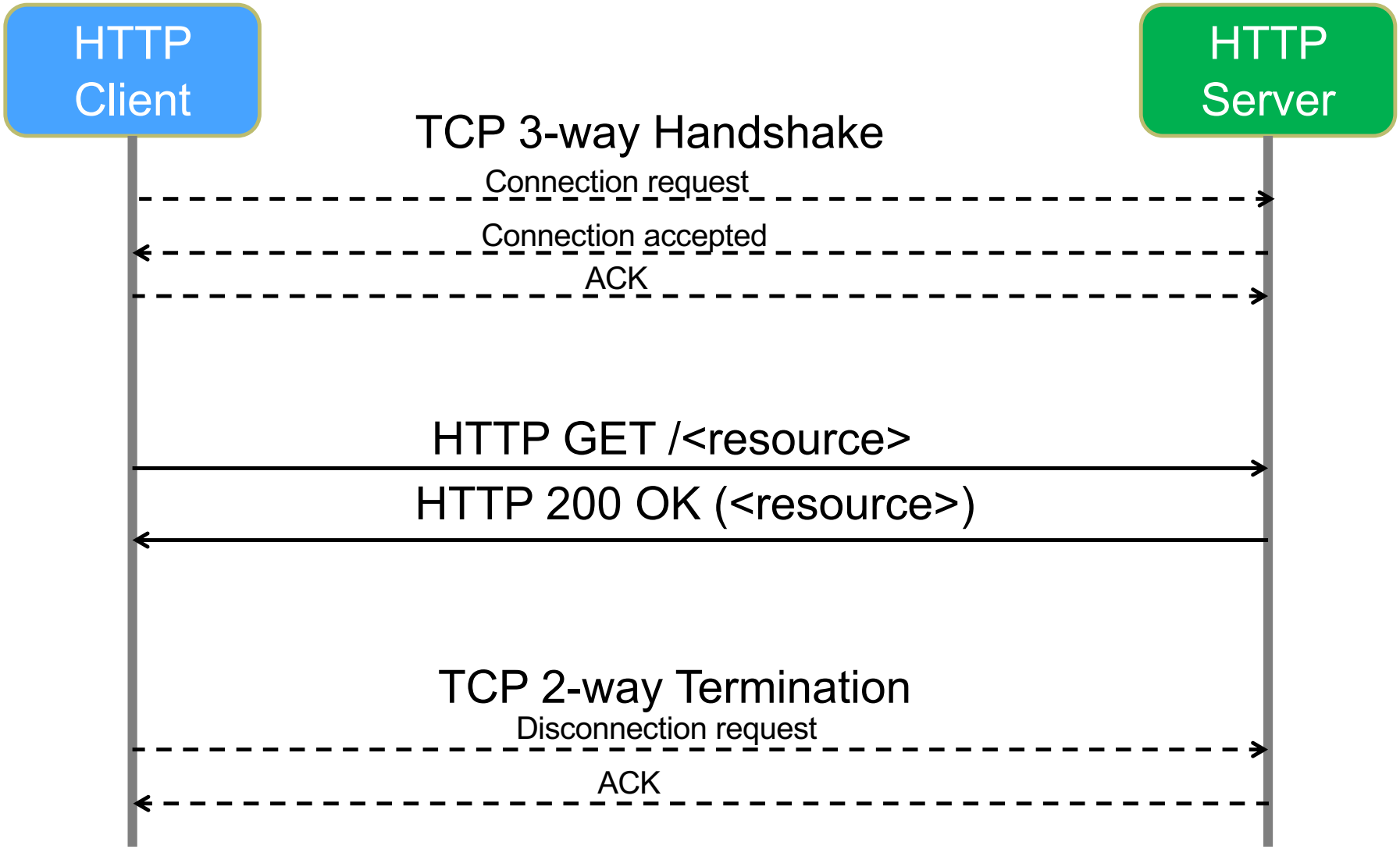
Difference MQTT-COAP

Features	CoAP	MQTT
Full Form	Constrained Application Protocol	Message Queue Telemetry Transport
Model used for communication	Request-Response, Publish-Subscribe	Publish-Subscribe
RESTful	Yes	No
Transport layer	Preferably UDP, TCP can also be used.	Preferably TCP, UDP can also be used (MQTT-S).
Header Size	4 Bytes	2 Bytes
Number of message types used	4	16
Messaging	Asynchronous & Synchronous	Asynchronous
Application Reliability	2 Levels	3 Levels
Security	IPSEC or DTLS	Not defined in the standard
Intermediaries	YES	YES (MQTT-S)
LLN Suitability (thousand nodes)	Excellent	Fair
Application success stories	Utility Field Area Networks	Extending enterprise messaging into IoT applications

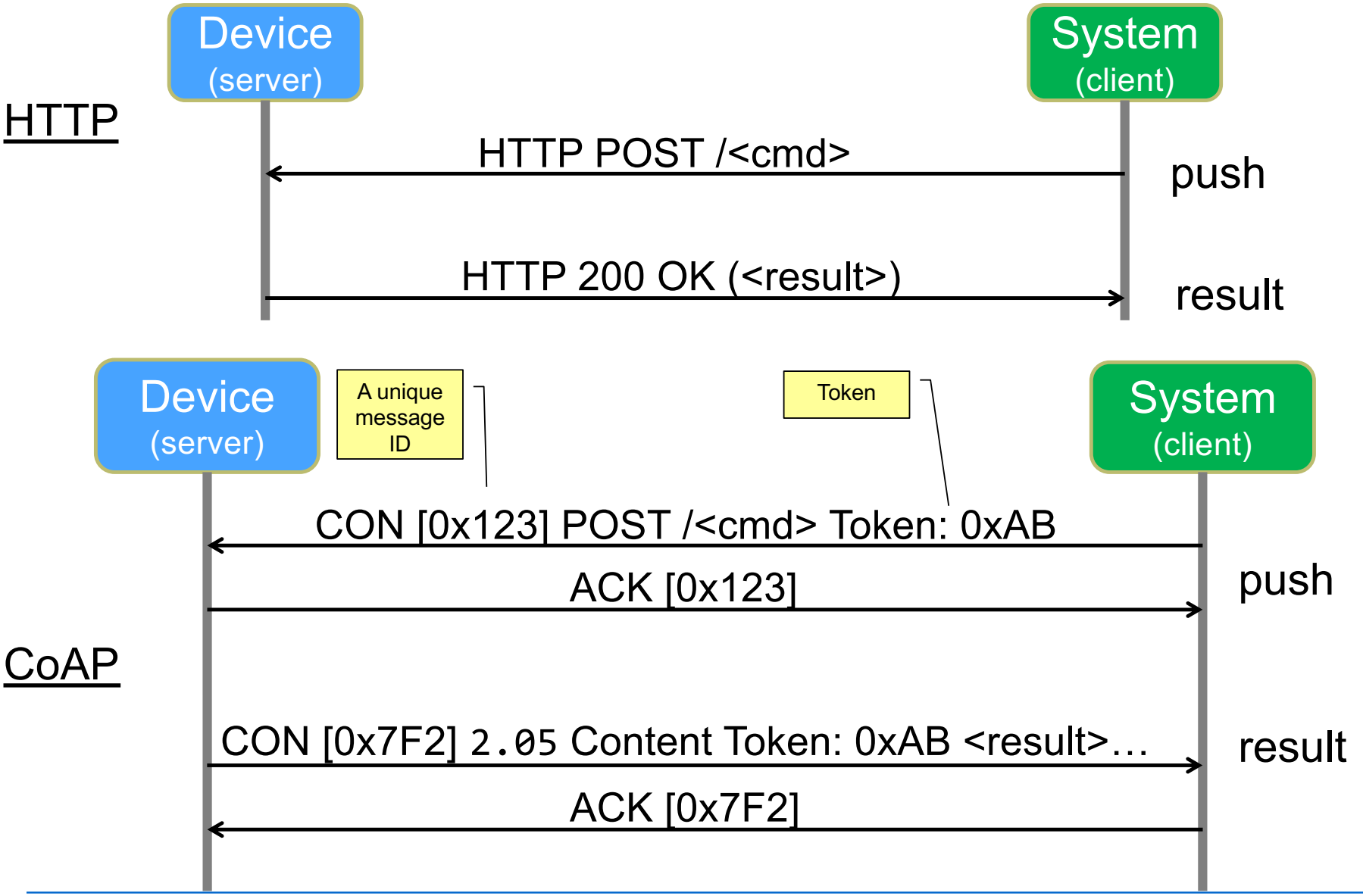
LLN = low power and lossy network

Source: RF Wireless World Consortium

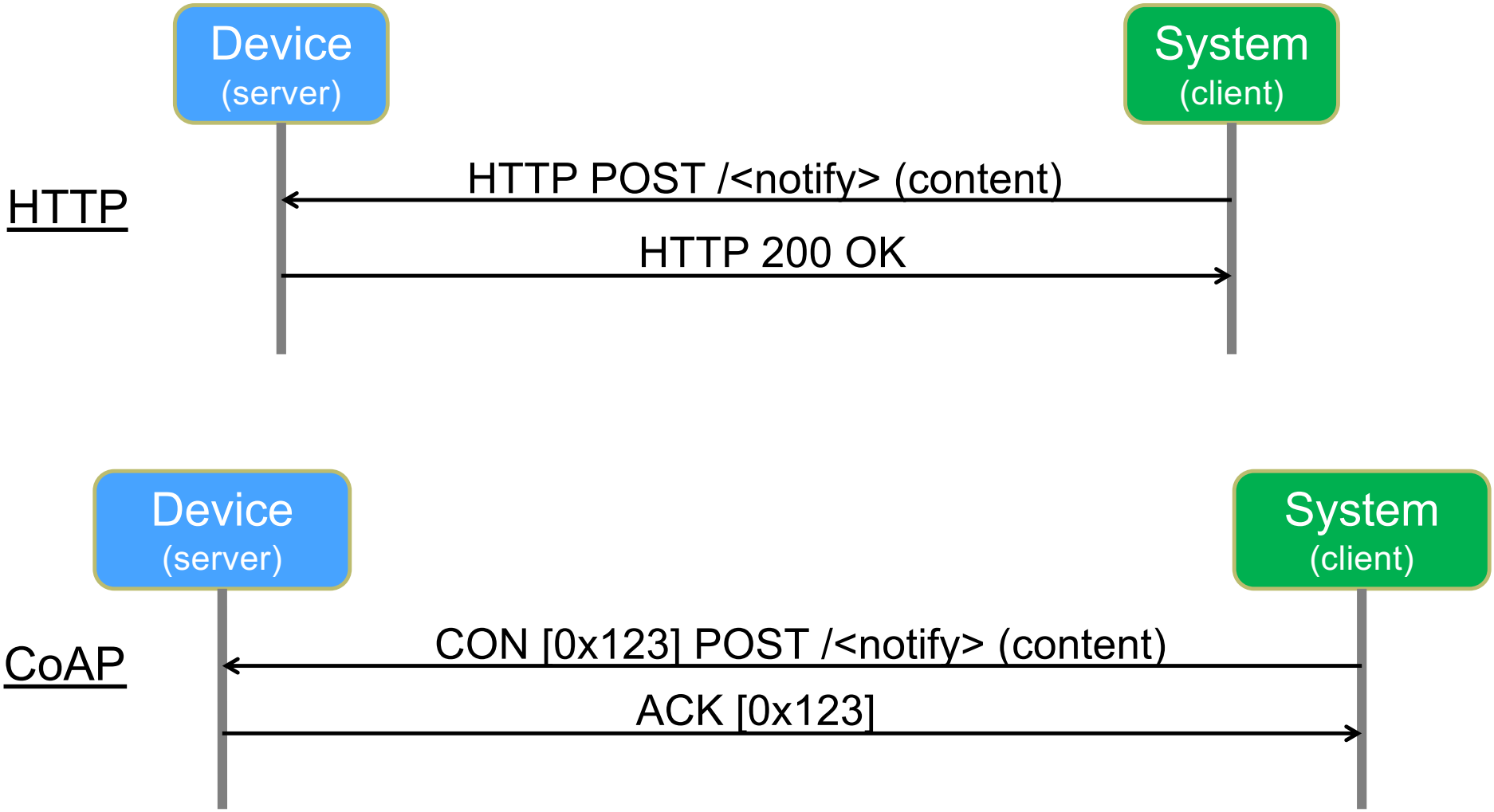
An HTTP Request



HTTP/CoAP: Command



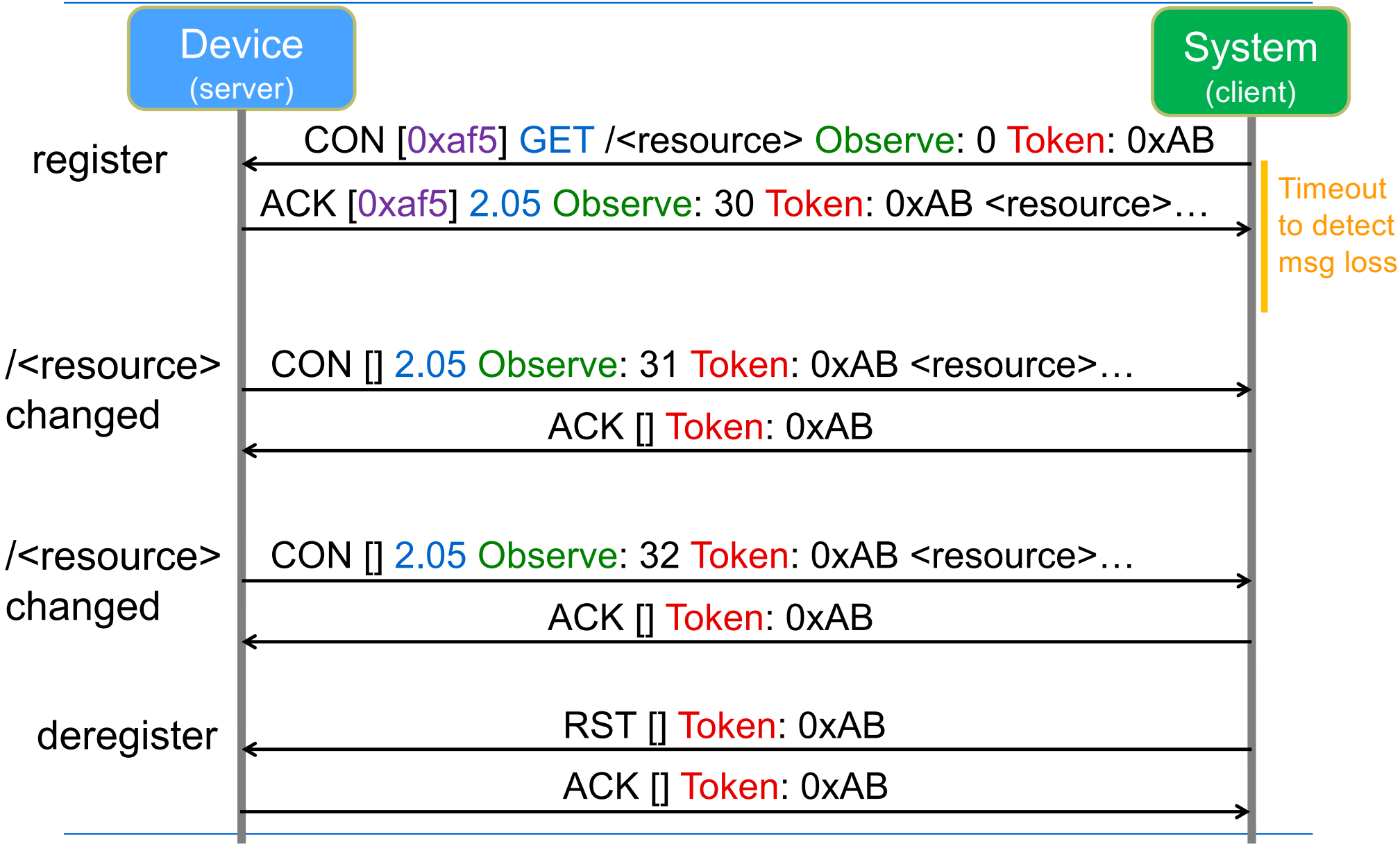
HTTP/CoAP: Notification



CoAP Observation

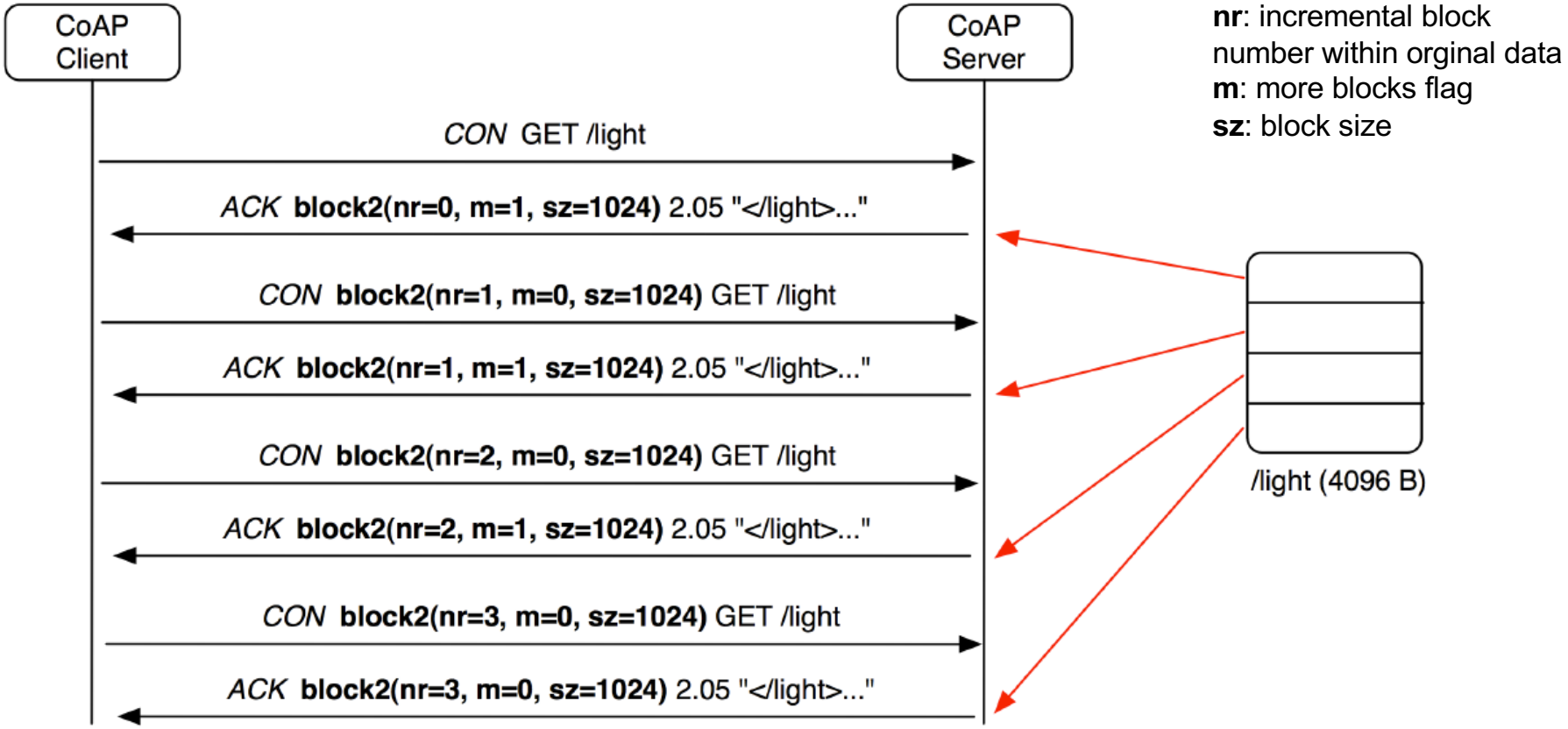
- PROBLEM:
  - REST paradigm is often „PULL“ type, i.e., data is obtained by issuing an explicit request
  - Information/data in IoT is often periodic/triggered (e.g., get temperature measurment every 2 sec, or get a notification if temperature goes higher than 75°C)
- SOLUTION:
  - Use Observation on CoAP resources.

CoAP: Telemetry



Blockwise Transfer

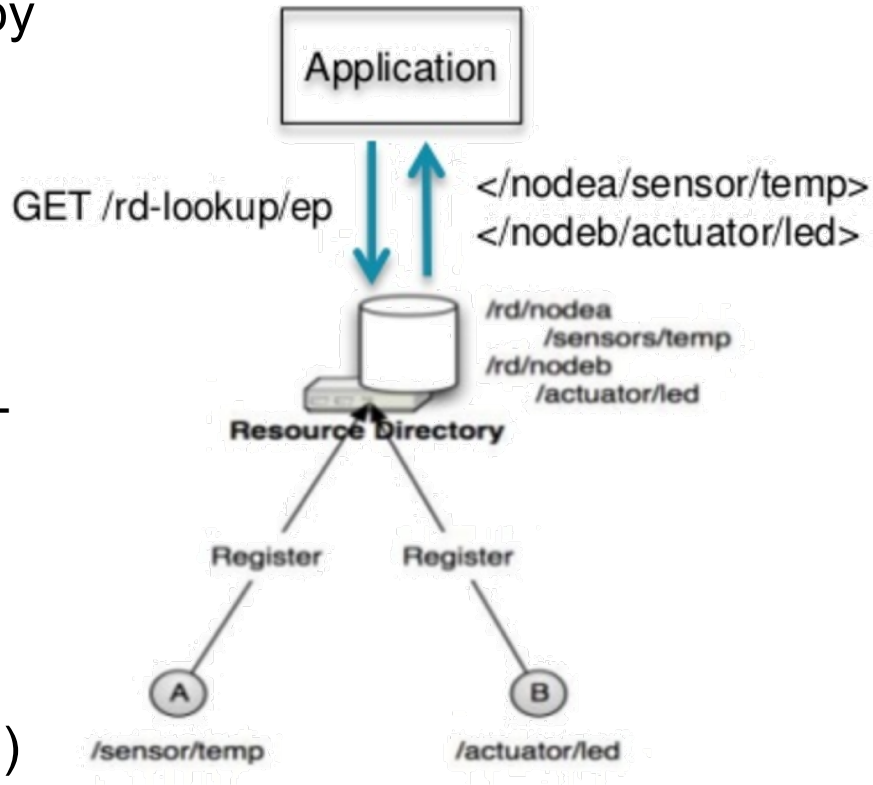
- PROBLEM: avoid segmentation in the lower layers (IPv6)
- SOLUTION: CoAP Block Transfer Mode
  - brings up fragmentation at the application layer



nr: incremental block number within orginal data
m: more blocks flag
sz: block size

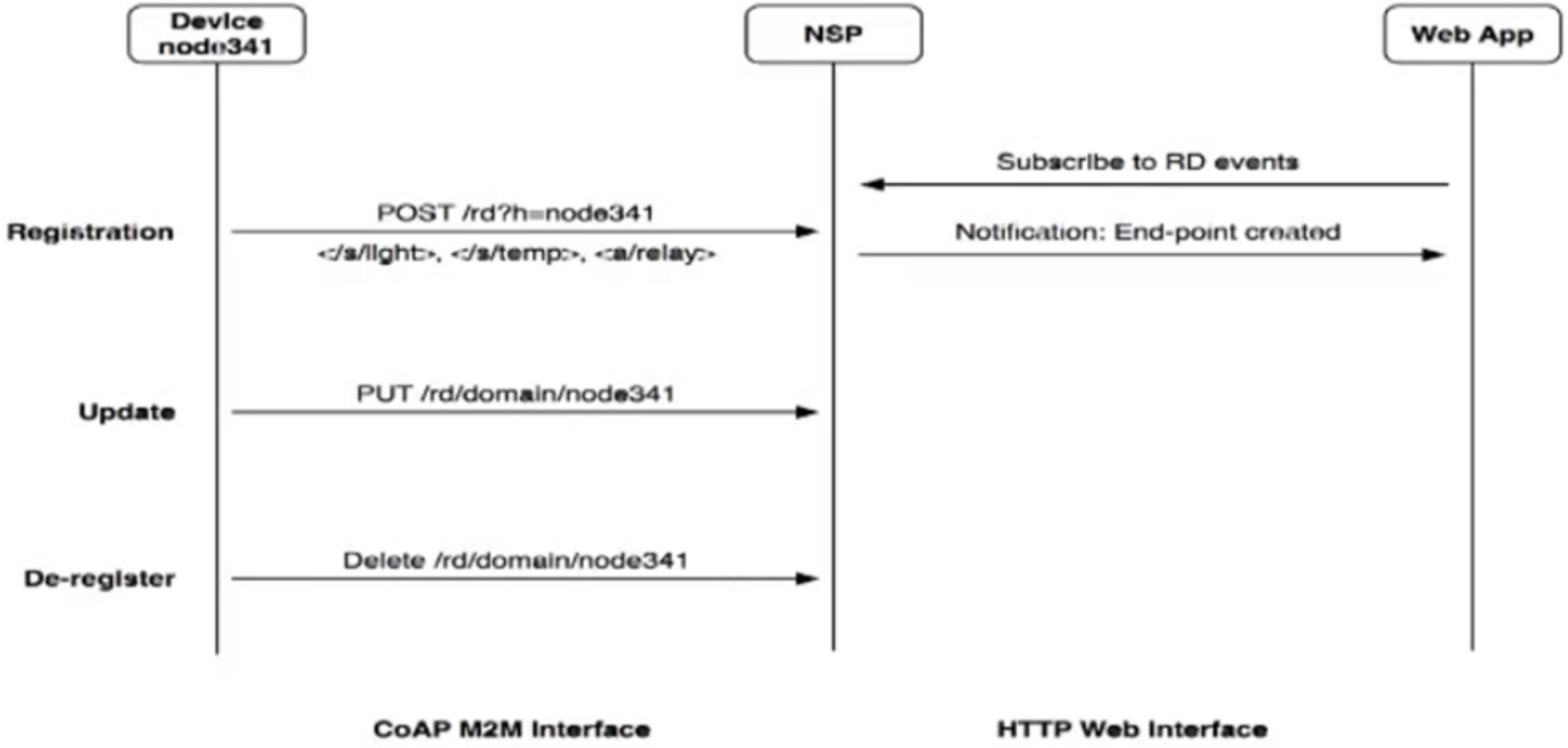
Discovery

- Resource Discovery
  - GOAL: Discovering the links hosted by CoAP servers
    - GET /.well-known/core?optional_query_string
  - Returns a link-header style format
    - URL, relation, type, interface, content-type etc.
- A Directory Approach
  - Supports sleeping nodes
  - No multicast traffic (longer battery life)
  - Remote lookup, hierarchical and federated distribution



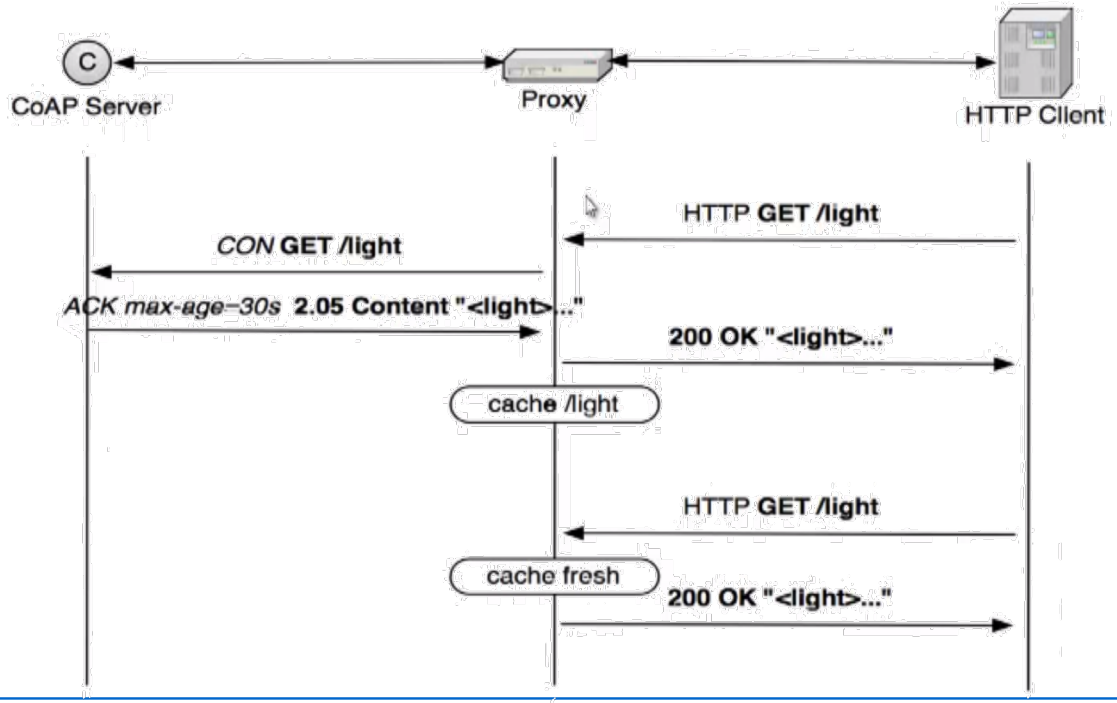
Building Resource Directory (RD)

- Nodes POST (register) their link-format to an RD
- Nodes PUT (refresh) to the RD periodically
- Node may DELETE (de-register) their link-format
- Nodes may GET (lookup) the RD or resource of other nodes



Proxying and Caching

- CoAP includes a simple caching model
  - Cacheability determined by response code
  - An option number mask determines if it is a cache key
- Freshness model
  - `Max-Age` option indicates cache lifetime
- Validation model
  - Checked using `Etag` option (version of resource representation)
- A proxy often supports caching
  - Usually on behalf of a constrained node, or a sleeping node
  - Or to reduce network traffic



CoAP: Implementations

CoAP Tools

- There are many open source implementations:
  - Java CoAP library `Californium` (Eclipse)
    - `Scandium`: Security for Californium
    - `Actinium`: App-server for Californium
  - `Erbium` C library
  - `libCoAP` C library
  - `openCoAP` C library
  - `jCoAP` Java library
  - Firefox has a CoAP plugin: `Copper`
- Commercial implementations

Eclipse CoAP Support

- See getting started under: https://iot.eclipse.org/getting-started

CoAP

A CoAP server exposing test resources is available at:
`coap://californium.eclipse.org:5683/` , and DTLS-enabled version at
`coaps://californium.eclipse.org:5684` .

It should be used by anyone interested in testing a CoAP client implementation against another endpoint, and more generally by anyone interested in understanding the key concepts of the CoAP protocol.

This server is running Eclipse Californium.

HTTP vs CoAP: Implementation & Weight

- HTTP
  - client more complex (ASCII parser)
  - more bytes to pay on data transfer
  - connection oriented via TCP
- CoAP
  - HTTP-like but binary
  - Connection-less via UDP
  - Client more simple than HTTP
  - “options” like HTTP headers (binary)
- HTTP & CoAP
  - Content-Type based on MIME

HTTP vs CoAP

- HTTP
  - More verbose (ASCII, headers, ...) for few data
  - Addressing problem (mobile roaming, NAT, ...)
  - No QoS (but based on TCP)
- CoAP
  - “observer” pattern
  - Addressing problem (mobile roaming, NAT, ...)
  - QoS with “confirmable” message or not

Conclusions

- CoAP supports HTTP-style RESTful applications
- CoAP reduces TCP carrier overhead and brings it under control of the application
- CoAP reduces the data-size overhead of HTTP/TCP significantly
  - typical GET and response just a few bytes in size
- CoAP support a variety of in-network behaviors/mechanisms that improve performance of low-resource devices
  - proxy, caching
- CoAP deals effectively with peculiarities of low resource nodes

Pair exercise: What are COAP & MQTTs strengths

- Read the article “MQTT and COAP: Underlying Protocols for the IoT” (ca 10 minutes)
- Turn 1: group ”blue”: explain MQTTs strengths and weaknesses to your partner (2-3 minutes)
- Turn 2: group “red” explain COPAs strengths and weaknesses to your partner (2-3 minutes)

MQTT vs CoAP

Connectivity: Device to Cloud

Internet of Things

- **CoAP** on UDP/IP (1-1) (GET, POST, PUT, DELETE)
  - coap://host/lamps/12/status
  - PubSub (m-n) **MQ Telemetry Transport** (on TCP/IP)

Classical Internet

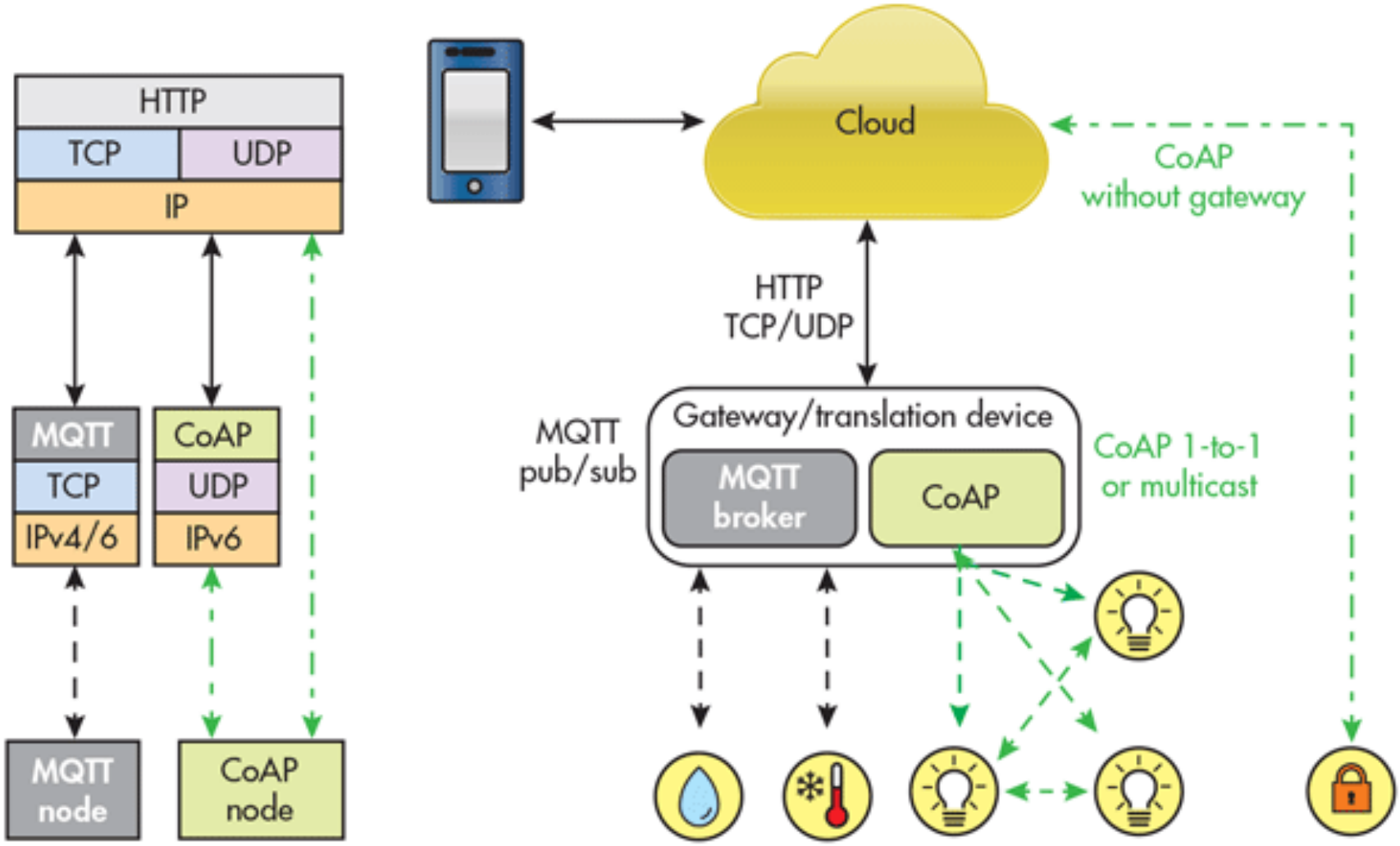
- HTTP (on TCP/IP)
  - http://host/docs/doc12
  - Web-Socket, RabbitMQ

- **MQTT-SN** (on ZigBee, Bluetooth/BLE, Z-Wave)
  - Connectionless
  - Gateways/proxy to translate MQTT-SN to MQTT

- MQTT/CoAP @ Eclipse IoT
  - Paho (MQTT und MQTT-SN) for devices
  - Mosquitto MQTT broker
  - Kura (MQTT Application framework)
  - Ponte (bridge HTTP, CoAP, MQTT)
  - Californium (CoAP)

Mosquitto on Raspberry Pi

MQTT and CoAP



Src: MQTT and CoAP: Underlying Protocols for the IoT, Electronic design, July 2015

CoAP vs HTTP vs MQTT (1)

Protocol	**CoAP**	REST/HTTP	**MQTT**
Transport	UDP	TCP	TCP
Routing	IPv6 with 6LoWPAN	IP	IP
RESTfullness	Yes	Yes	No
Messaging	Request/Response	Request/Response	Publish/Subscribe
2G, 3G, 4G Suitability (1000s nodes)	Excellent	Excellent	Excellent
LLN *) Suitability (1000s nodes)	Excellent	Fair	Fair ++
Simplicity	CoAP is less complex and has lower overhead compared to HTTP. CoAP allows only 4 types of messages.	Can be complex (but often well understood) – multitude of return codes and methods. REST is a great principle but not always the best for SIMPLE data applications.	Has few methods (publish/subscribe/ unsubscribe), quick to learn. MQTT allows 16 different types of messages.

*) Low-Power and Lossy Networks (LLN)

CoAP vs HTTP vs MQTT (2)

Protocol	CoAP	REST/HTTP	MQTT
Lightweight Stack (CPU & MEM)	CoAP is simpler than MQTT and has lower consumption of CPU and memory.	HTTP (often with associated XML or JSON libraries for SOAP or REST etc) can be relatively large on top of OS network libraries. Plus … even if the client is small, consider whether it is really necessary to run an HTTP server on every device.	MQTT has been trivially implemented on tiny to larger platforms in very small libraries (IBM ref implementation = ~80 kB for full broker).
Light on the network	CoAP's packet size is smaller than that of MQTT. CoAP header is of 4 bytes. The protocol was optimized for LLN.	HTTP is relatively verbose – lots of „chatter" in a POST.	The smallest possible packet size for an MQTT message is 2 bytes (MQTT header is of 2 bytes). The protocol was optimized from start for unreliable, low-bandwidth, expensive, high-latency networks.

CoAP vs HTTP vs MQTT (3)

Protocol	CoAP	REST/HTTP	MQTT
Communication model	CoAP supports both synchronous and asynchronous messaging.		MQTT supports asynchronous messaging: MQTT has a highly decoupled publisher and subscriber model.
Easy distribution of data	CoAP network is inherently **one-to-one**; CoAP supports multicast (because of UDP, IP multicast can be used for **one-to-many**). CoAP has a simplified "**observe**" mechanism similar to MQTT's pub/sub.	HTTP is **one-to-one** (can be mediated/ clustered but no distribution mechanism). To distribute to multiple receivers a large number of POSTs may be required.	MQTT distributes **1-to-none, 1-to-1, 1-to-many** via the pub/sub mechanism → very efficient
Message reliability (QoS levels)	CoAP has a **2 level application reliability**. It provides a very simple method of providing a "confirmable" message and a "non-confirmable" message.	HTTP has no retry/confirmation/attempt at once-only delivery. Retry needs to be written in the application level. Application must also handle timeouts.	MQTT has a **3 level application reliability**, whereas: It supports fire-and-forget or fire-and-confirm or exactly-once (aka QoS 0/1/2)

CoAP vs HTTP vs MQTT (4)

Protocol	CoAP	RESTful HTTP	MQTT
Energy-efficiency	Higher than MQTT due to the use of UDP (less messages, asynchrony).	low	high
Dynamic discovery	Yes	No	No
Encoding	Binary	Plain text	Binary
Real-time	No	No	No
Security	DTLS	TLS/SSL	Username-password authentication + SSL encryption
Standard maturity	Still evolving (IETF)	Mature (IETF)	Mature (OASIS)