Distributed Smart Things

Giorgio Delzanno

Dipartimento di
Informatica
Bioinformatica
Robotica
Ingegneria dei
Sistemi
University of Genova, Italy

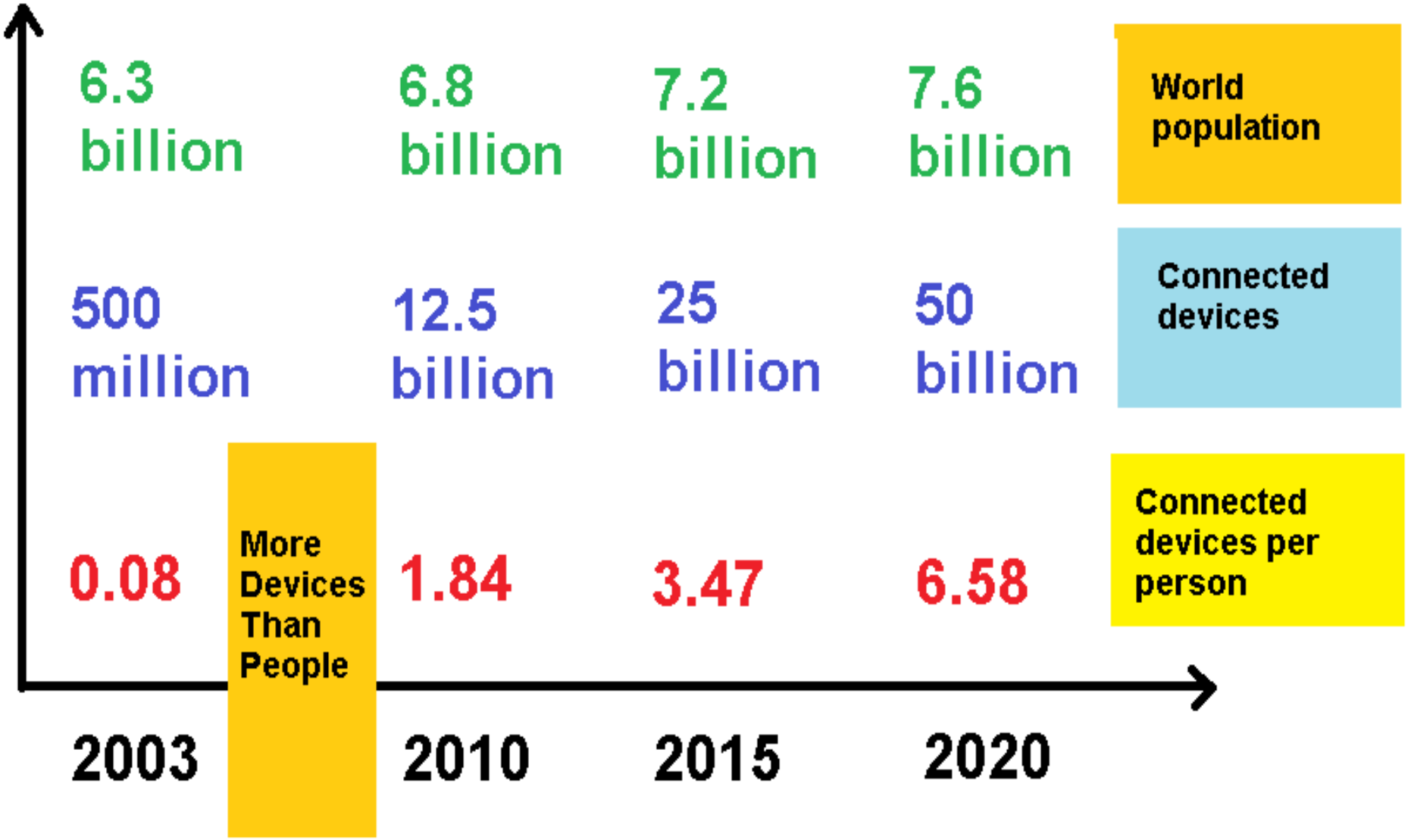
giorgio.delzanno@unige.it

Google/images

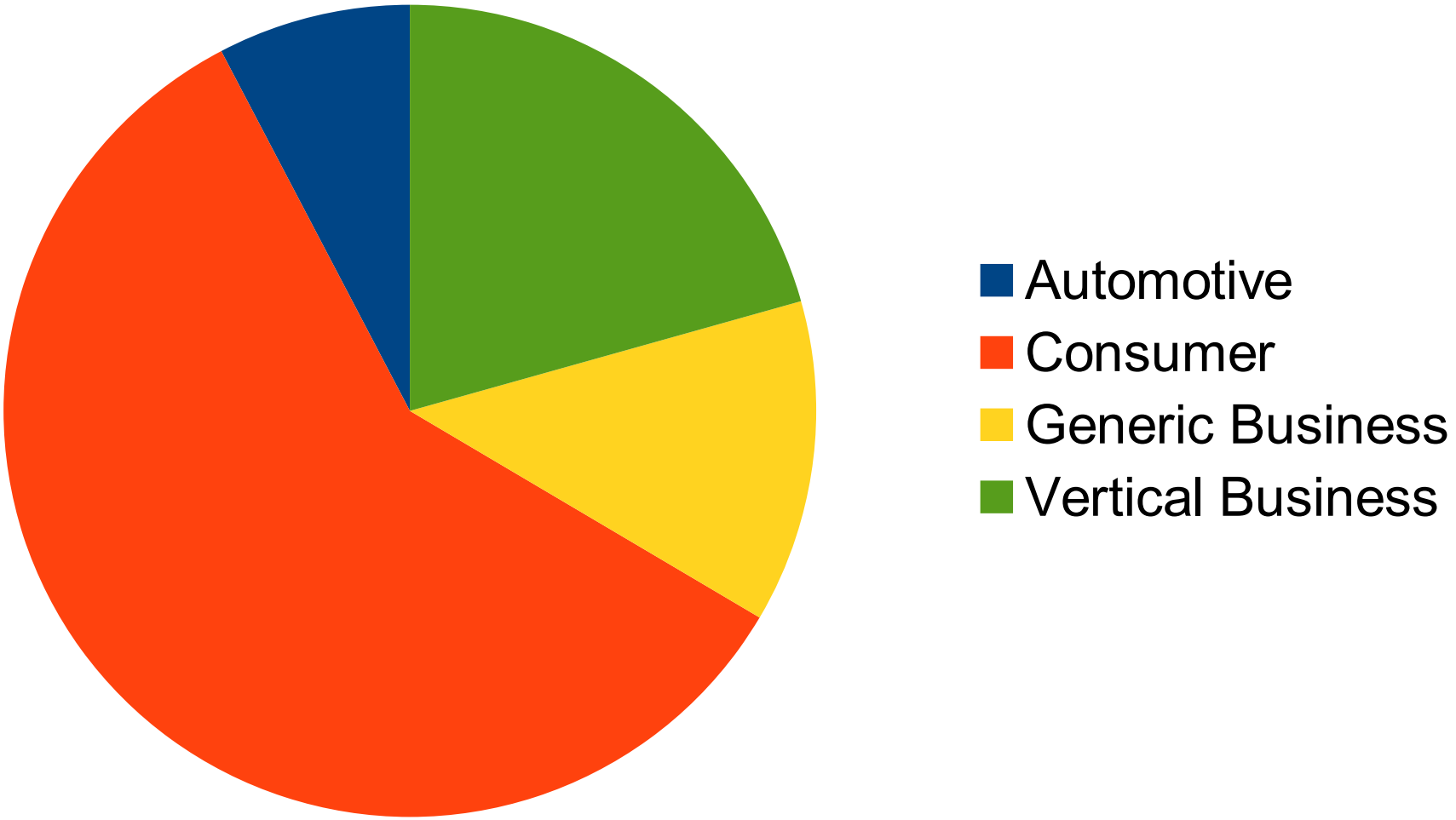
“When almost every object either contains a computer or can have a tab attached to it, obtaining information will be trivial”

Mark Weiser, The Computer for the Twenty-First Century, 1991

A World of Connected Things



A World of Connected Things



Definition

The Internet of Things (IoT) is the network of physical objects or "things" embedded with electronics, software, sensors, and connectivity to enable objects to exchange data with the manufacturer, operator and/or other connected devices

From Wikipedia

Internet of Everything
Machine2Machine (M2M)
Web of Things (WoT)
Ubiquitous/Pervasive Computing

Key Enablers

- Processing power
- Networking infrastructure and connectivity
- Low cost, secure devices
- Storage
- Secure and portable software

- Money!

Key Domains

- Smart Cities
- Smart Home
- Data Centers
- Manufacturing/Industry
- Health/Independent Living
- Environment/Agriculture
- Automotive
- ...

Standardization

- Connectivity/interoperability standards
- Operating Systems
- Topologies
- Security
- Software abstractions (API)
- Multi-tenant environments

- Open Source (hw/sw) to avoid fragmentation due to in-house hw/sw/protocols

Processing

Need of scalable processor architectures

- Microcontrollers
- Networking & storage
- Mobile and home solutions
- Data center infrastructures

Technology

Things

- Connected security system
- Thermostats
- Cars
- Electronic appliances
- Lights in household and commercial environments
- Alarm clocks
- Speaker systems

But how can we make things smart and distributed?

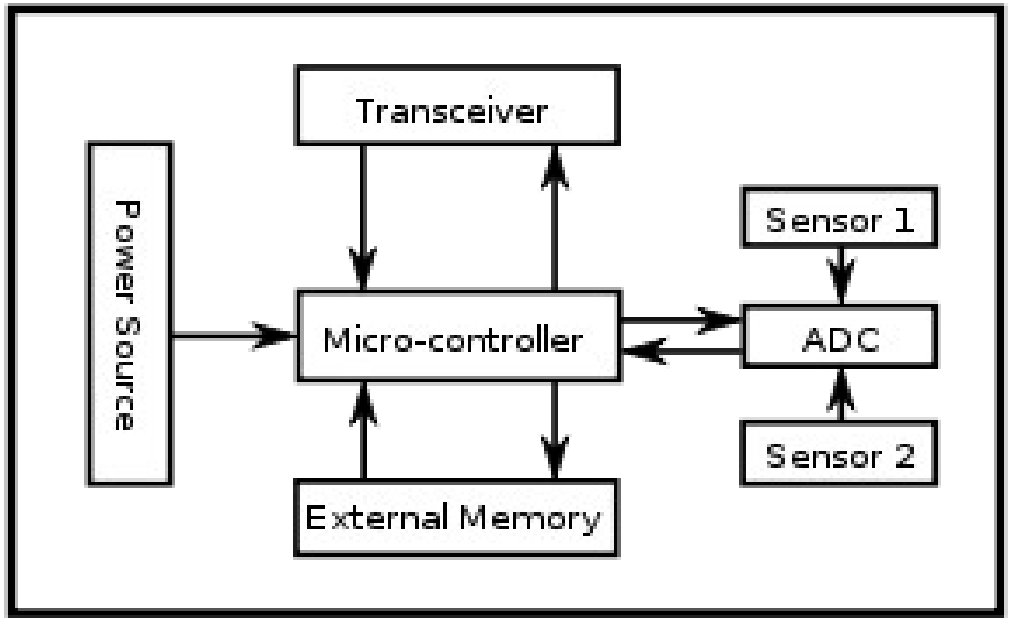
Wireless Sensor Networks (WSN)

- Large number of nodes (motes)
- Limited power and cost
- Prone to failures
- Nodes may not have a global ID
  (like IP addresses)
- Tight integration with sensing tasks
- Communication among motes and other devices

Mote (Sensor)

- Sensing external phenomena
- Processing information
- Storing information
- Communicating with other motes or devices
- Energy harvesting (solar, vibration, …)

Mote: Typical architecture



- Storage (Flash)
- Microcontroller (EPROM, CPU)
- Lowpower standby/wakeup
- Sensor interface (ADC)
- RF Transceiver (Wireless communication)
- Wired net interface (USB, ...)

Power managemet

- Sensors can be data originator or data router
- Power conservation and power management at different levels are mandatory:
  - Power-aware communication
  - Low-power processing (and processors)
  - Low-power sensing
- The “Idle listening problem” of radio communication: Power consumption is the same when transmitting, receiving, or listening for potential receptions

Power aware computing

- Ultra low power controllers
- Dynamic power management of devices
- Components switch off after some idle time
- Energy aware OS
- Energy aware packet forwarding
- Energy aware wireless communication

Example: TI LP MCU Family

Texas Instrument MSP430 (more recently 32bits ARM based MSP432) ultra-low-power microcontroller (MCU) family

- SW managed flexible clock system with von-Neumann architecture
- Interrupt handling
- SRAM/FRAM for real-time data capture
- Analog and digital peripherals
- Real-time clock that can operate without the CPU for simpler data capture and manipulation.

FRAM

Ferroelectric Random Access Memory is a memory technology that combines Flash and SRAM

- Non-volatile like Flash
- Offers fast and low power writes (100,000x read/write without consuming extra battery power)
- Code and data security that is less vulnerable to attackers than Flash/EEPROM
- Resistance to radiation and electromagnetic fields

Built from the ground up to preserve battery life

IoT Operating Systems

TinyOS

- An open source OS for low-power wireless devices, written in a dialect of the C language optimized for the memory limits of sensor networks

- Component-based, components are connected to each other using interfaces (for packet communication, routing, sensing, actuation and storage)

- TinyOS is completely non-blocking:
  - It has only one stack
  - all I/O operations that last longer than a few hundred microseconds are asynchronous and have a callback.

- A component can post a task, which the OS will schedule to run later. Tasks are non-preemptive and run in FIFO order

ContikiOS

- Open Source OS for low power/IoT devices
- Multitasking and a built-in TCP/IP stack
- Contiki only needs about 10 kilobytes of RAM and 30 kilobytes of ROM
- Based on **protothread**, a low-overhead mechanism for concurrent programming
- Protothreads function are stackless, lightweight threads providing a blocking context using minimal memory per protothread (order of bytes)

M2M Communication

Bluetooth (10 meters)

- Bluetooth technology is a standard for wireless communication between phones and computers
- The Bluetooth link layer, operating in the 2.4-GHz ISM band, was previously standardized as IEEE 802.15.1
- Bluetooth low energy: reduce power consumption, allows any number of connected devices

WI-FI (100 meters)

- Wi-Fi technology, based on the IEEE 802.11 standard, created for Internet connectivity and natively integrated with the TCP/IP stack

- Wi-Fi networks operate in the ISM 2.4-GHz band and in the 5-GHz band where more channels exist and higher data rates are available.

- Range of 5-GHz radios inside buildings is shorter compared to 2.4 GHz, 5 GHz is mainly used in enterprise applications

- Wi-fi/Bluetooth:
  Integrated in laptop, smartphones, tablet

IoT and WI-FI/Bluetooth

WI-FI/Bluetooth can be used to connect a smart object to the Internet via a cellular phone or a laptop

Low Power M2M Communication

IEEE 802.15.4

- Low power wireless data communications use the 2.4 GHz radio frequency spectrum under the IEEE 802.15.4 standard, suitable for short range and low data rates

- Unlike Wi-Fi and Bluetooth, low power consumption operate for several years from a single battery without any maintenance

- Updates to the standard have expanded the number of **sub-1 Ghz** available channels

- Zigbee is an implementation of sub-1 Ghz protocols

ZigBee

A low-cost, low-power, wireless network standard based on the IEEE standard 802.15.4

Targeted at development of long battery life devices in wireless control and monitoring applications

ZigBee minimizes the time the radio is on, so as to reduce power use

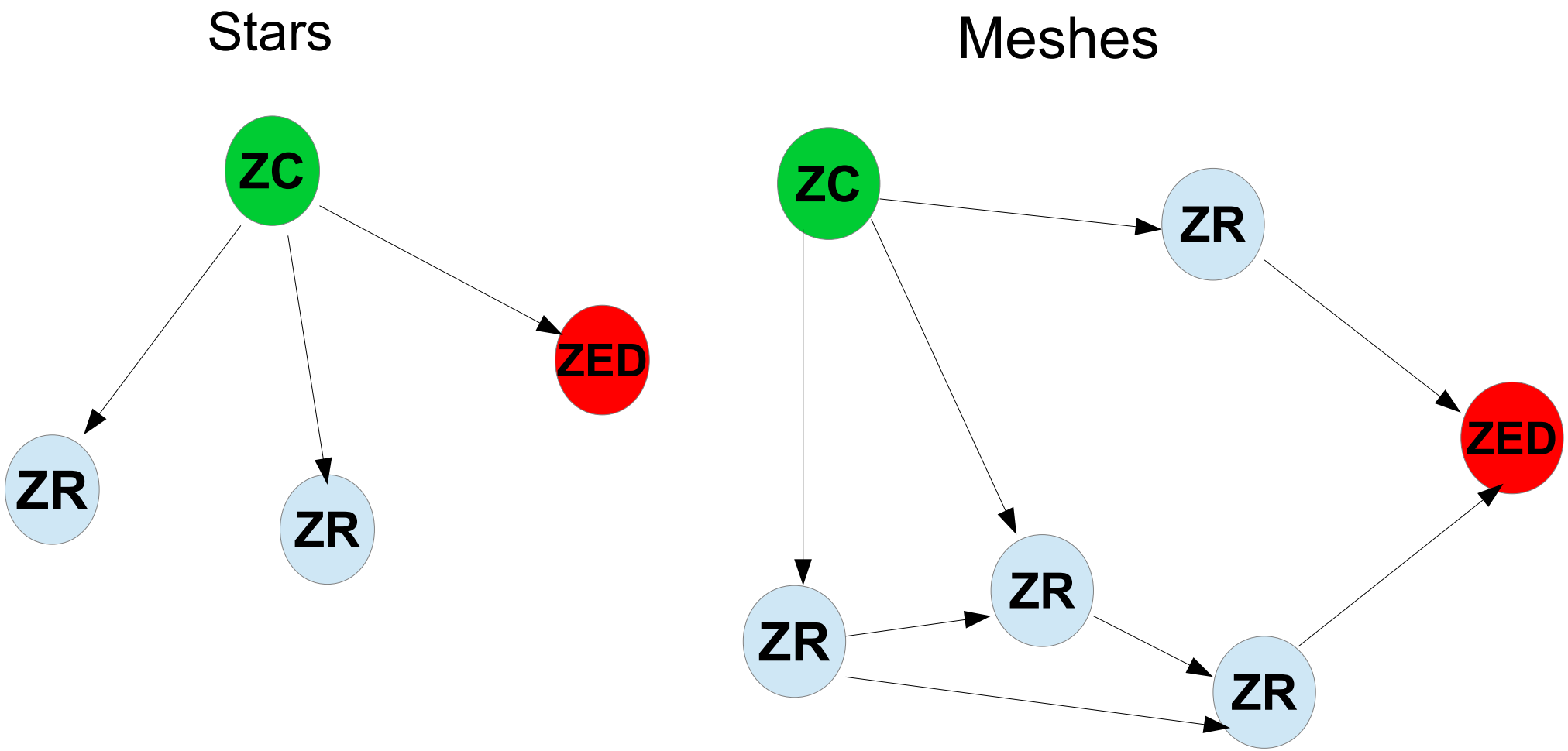
It works with mesh/star/tree network topologies

ZigBee Devices

ZigBee devices are of three types:

- ZigBee Coordinator (ZC): forms the root of the network and bridge to other networks
- ZigBee Router (ZR): acts as an intermediate router, passing on data from other devices
- ZigBee End Device (ZED): contains just enough functionality to talk to the parent node (either the Coordinator or a Router); it cannot relay data from other devices.
- SED require less memory, less power consumption, less expensive hardware than ZR/ZC

ZigBee Topologies



ZigBee Networks

In non-beacon-enabled networks ZRs have their receivers continuously active, requiring more power supply

Ok for heterogeneous networks in which some devices receive continuously, while others only transmit when an external stimulus is detected

Example wireless light switch:

- ZC node at the lamp (connected to supply) receive constantly
- ZED at a battery-powered light switch that remains asleep until the switch is thrown, then wakes up, sends a command to the lamp, receives an acknowledgment, and returns to sleep.

ZigBee Networks

In beacon-enabled networks, the special network nodes called ZRs transmit periodic beacons to confirm their presence to other network nodes.

Nodes only need to be active while a beacon is being transmitted

Nodes may sleep between beacons, thus lowering their duty cycle and extending their battery life

6LoWPAN

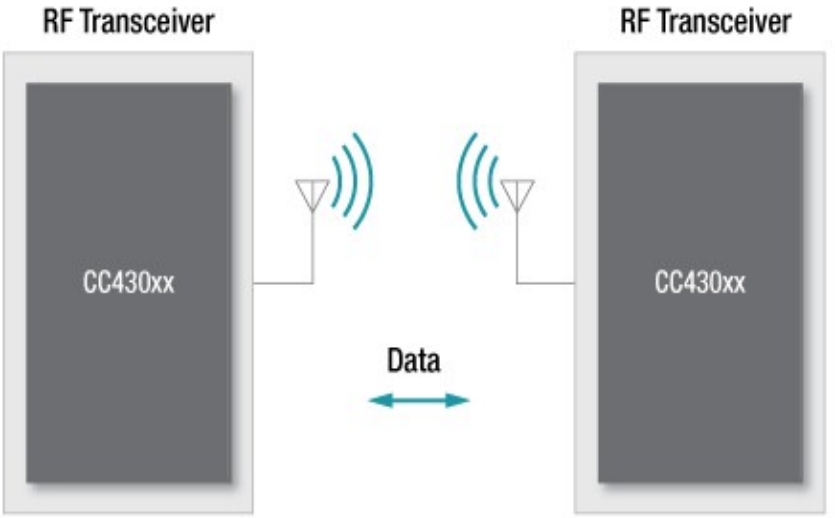
"The Internet Protocol could and should be applied even to the smallest devices"

6LoWPAN

"The Internet Protocol could and should be applied even to the smallest devices"

- 6LoWPAN = IPv6 over Low power Wireless Personal Area Networks
- Encapsulation and header compression mechanisms that allow IPv6 packets to be sent and received over IEEE 802.15.4 based networks
- Based on Edge Routers that identify classes subnets
- Other protocols: ZigBee IP, Thread, Z-Wave

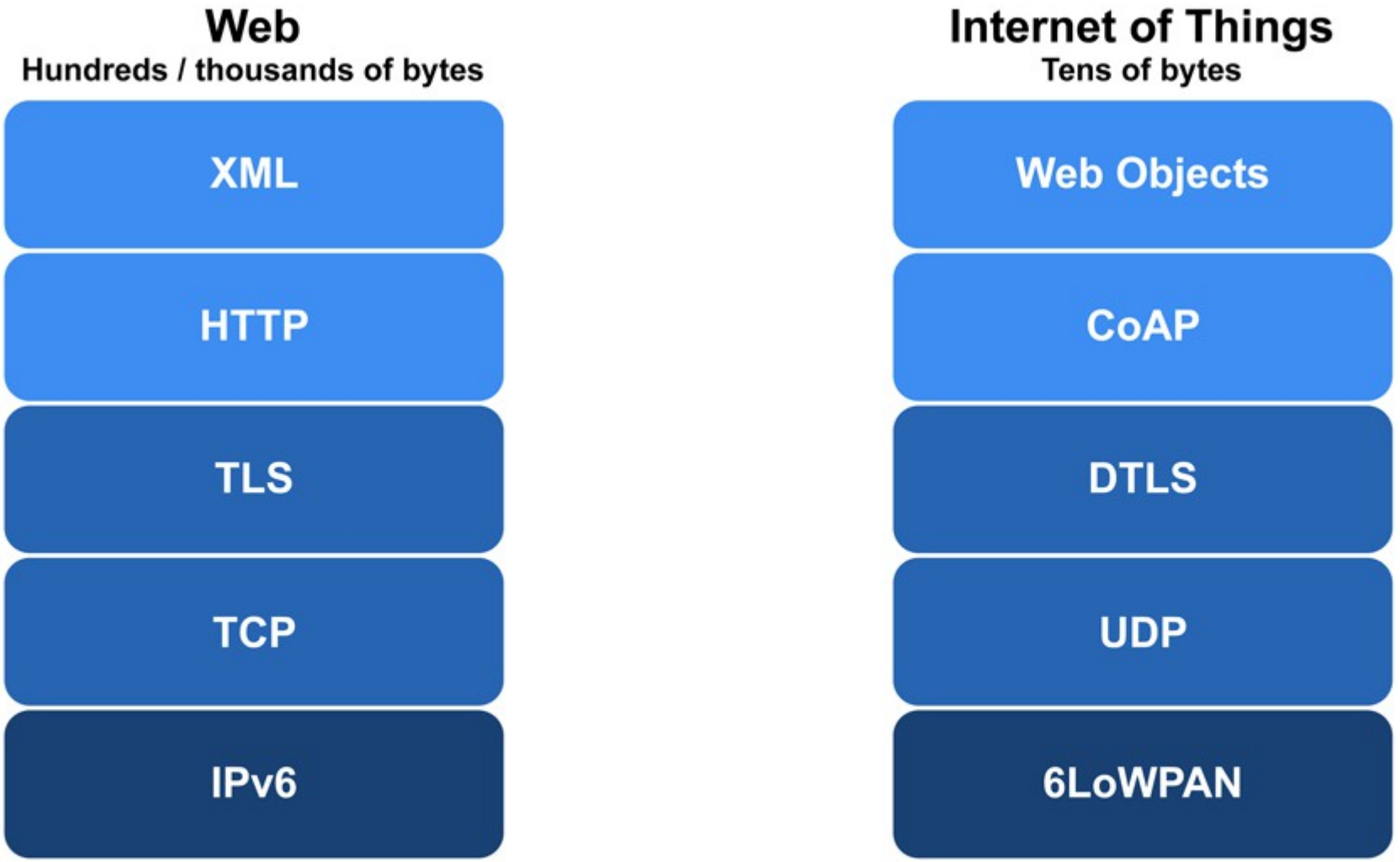
Example: TI LP RF-MCU Family



- TI CC430 SoCs = high performance RF transceiver and MSP430 ULP MCU on one chip.
- Sub 1GHz radio communication between two or more participants in a network

Application Layer

New Protocols



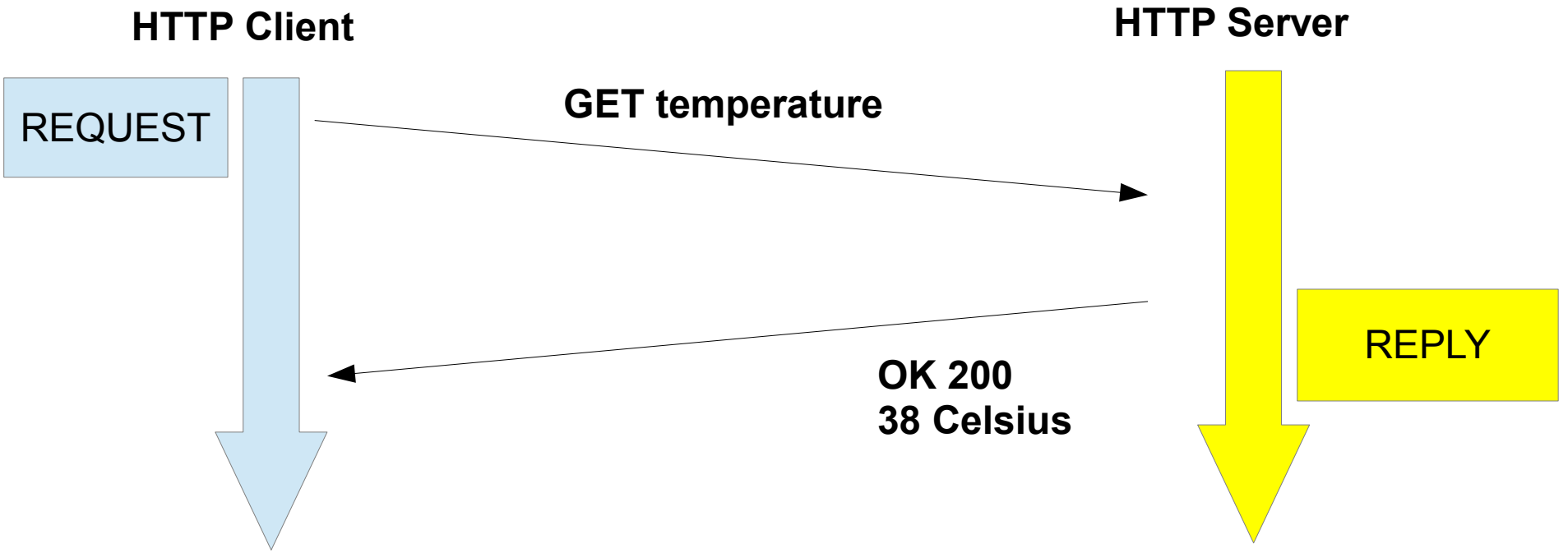
- Inefficient content encoding
- Huge overhead, difficult parsing
- Requires full Internet devices

- Efficient objects
- Efficient Web
- Optimized IP access

REST Protocol

- Recources are identified by URIs like
  http://www.example.it:8080/sensors/?id=light

- They managed by servers and accessed synchronously by clients via the HTTP protocol (request/response)



Constrained Application Protocol

(Coap)

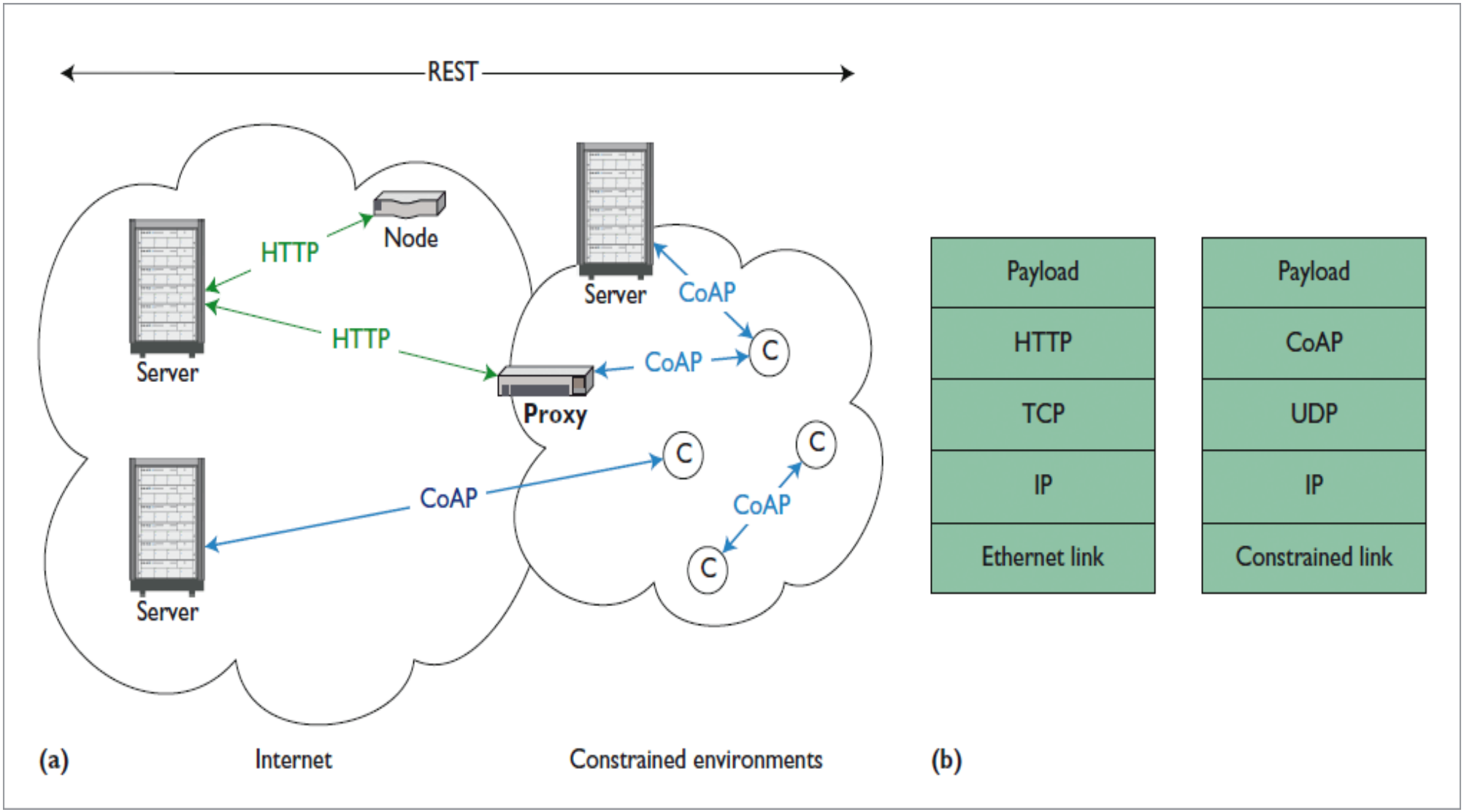
Coap

Constrained Application Protocol (Coap), a REST-based, UDP over IP protocol for constrained networks

- URI and content-type support
- Simple caching based on max-age
- Translates to HTTP via proxies
- Asynchronous transaction model
- UDP binding with reliability and multicast support
- Confirmable and non-confirmable messages

Additional features:
stop&wait, discovery, observation, block transfer mode

Coap

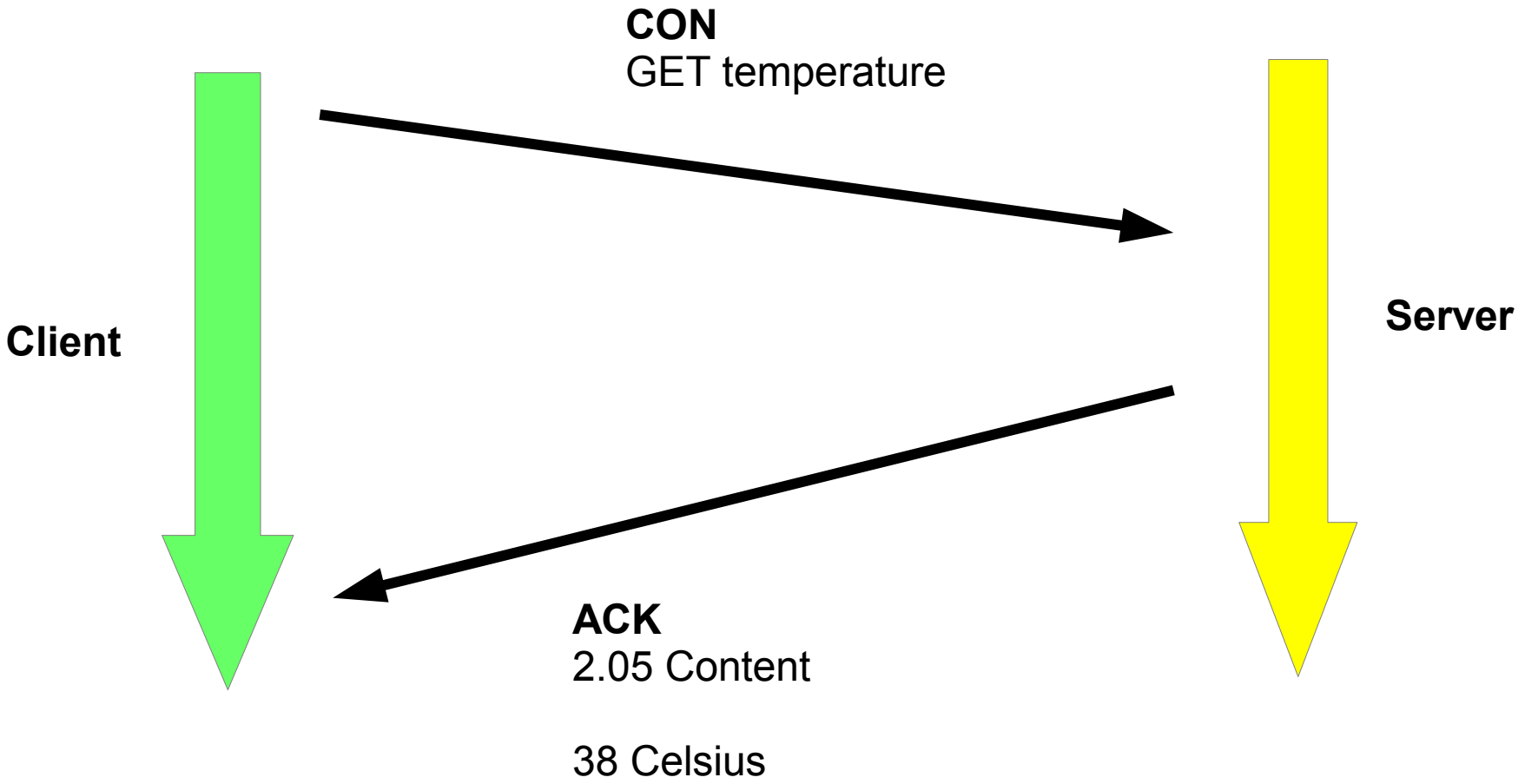

REST
HTTP
Node
Server
HTTP
Server
CoAP
C
CoAP
Proxy
C
C
CoAP
CoAP
C
Server
(a)
Internet
Constrained environments
(b)
Payload
HTTP
TCP
IP
Ethernet link
Payload
CoAP
UDP
IP
Constrained link

Coap Protocol

Messages with 4 Byte Header
Version : Type : Token_length : Code : Message_ID

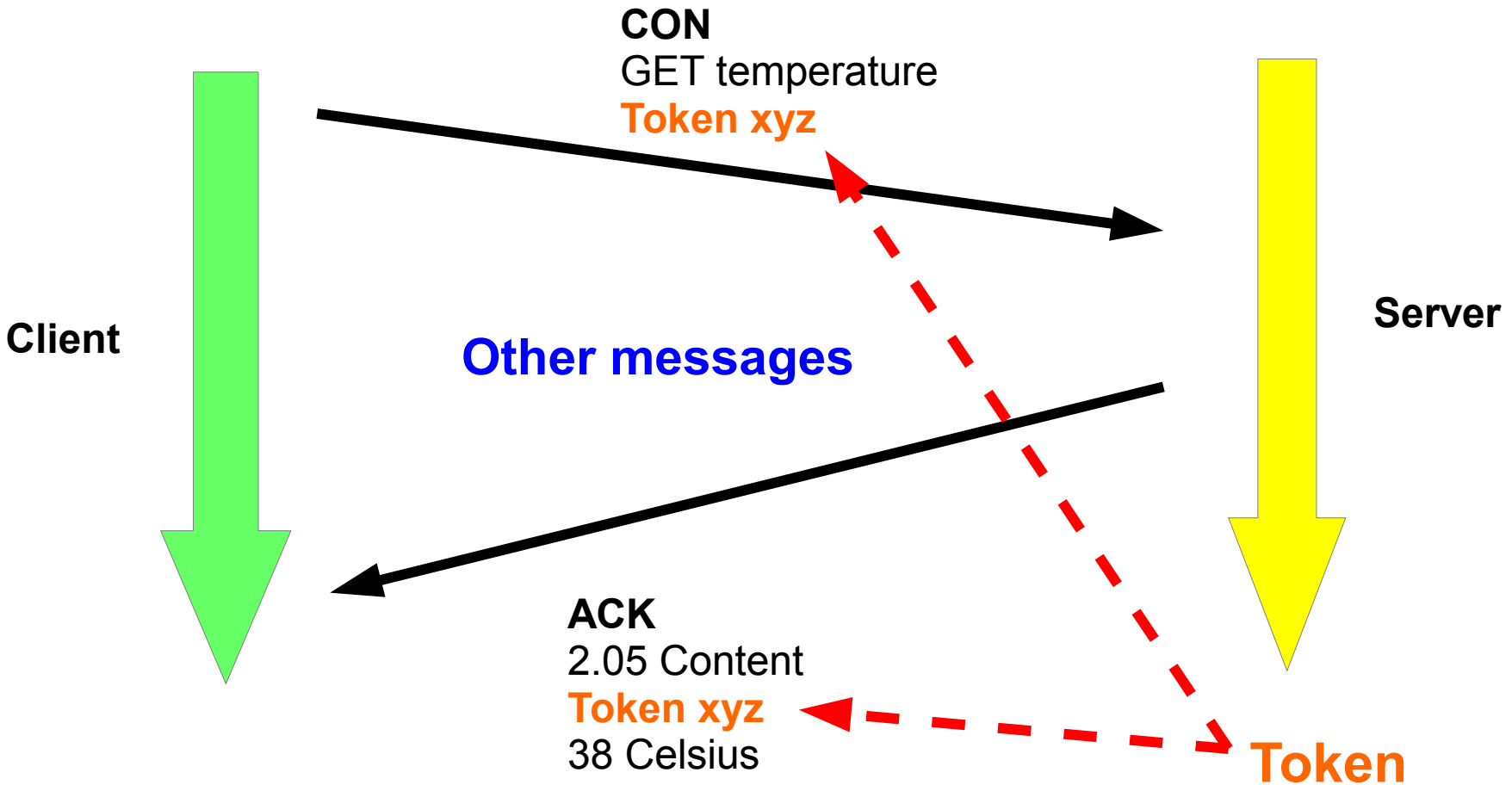
Coap Protocol

Messages with 4 Byte Header
Version : Type : Token_length : Code : Message_ID



Coap Protocol

Messages with 4 Byte Header
Version : Type : Token_length : Code : Message_ID



Other Features

- **Stop and Wait**: repeat a request after a time-out in case ACK is not coming back (max number of attempts, until total duration exceeds max_transmit_time
- **Observation to provide periodic requests**: registration on a resource, observation of current state, notification upon updates
- **Block Transfer Mode**: fragmentation from the transport to the application layer using sequences of confirmable messages associated to blocks of large messages
- **Discovery**: special requests (“wellknown”) to discover resources managed by a server (e.g. light, etc)

Message Queue Telemetry Transport (MQTT)

MQTT

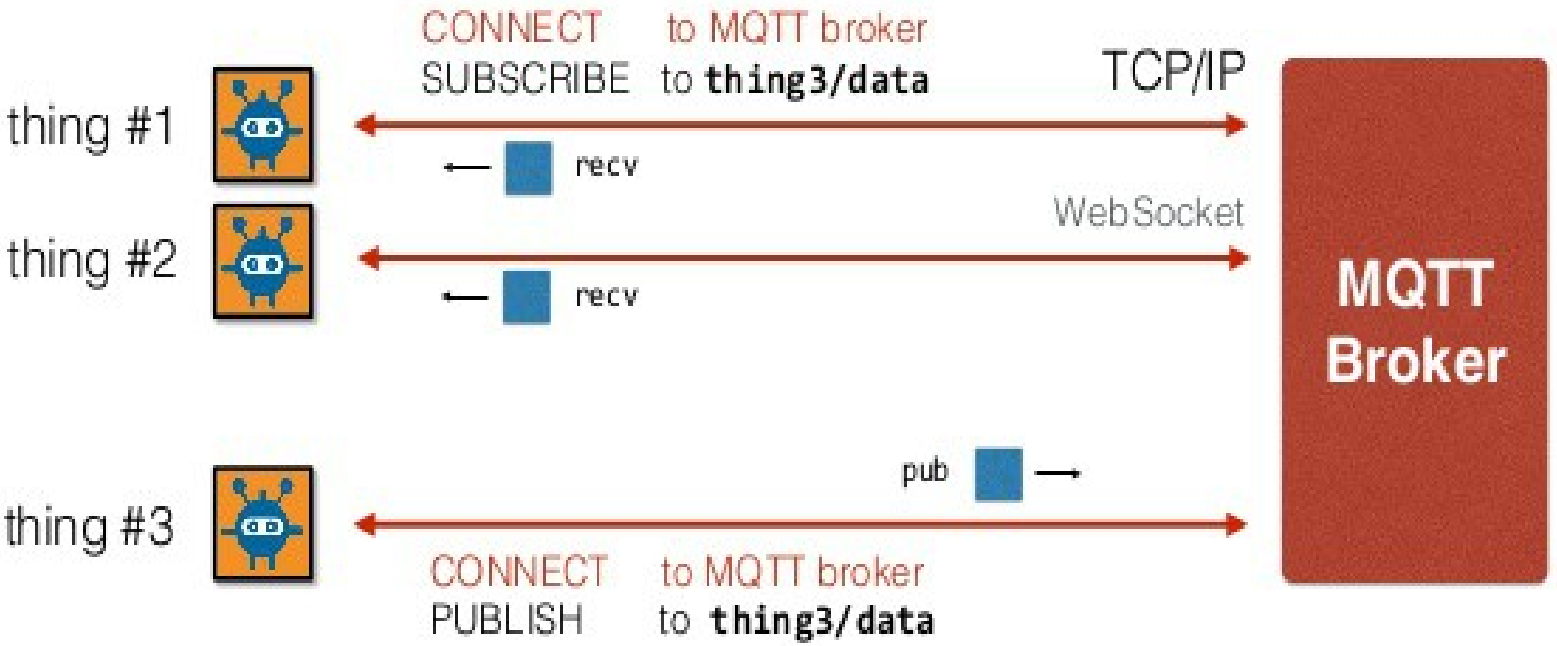
- MQTT (Message Queue Telemetry Transport) is a light weight messaging protocol built on top of the TCP/IP protocol (instead of UDP as Coap)

- Designed for connections with remote locations where a “small code footprint” is required and/or “network bandwidth is limited”

- Publish-subscribe architecture based on WebSockets:
  A message broker is responsible for distributing messages to interested clients based on the topic of a message.

MQTT

MQTT bi-directional, async "push" communication



WebSocket (RFC 6455)

- Full-duplex communication over a single TCP connection
- Standardized way for the server to send content to the browser without being solicited by the client, and allowing for messages to be passed back and forth while keeping the connection open.
- In this way a two-way (bi-directional) ongoing conversation can take place between a browser and the server.
- WebSocket can be used for communication between drivers of smart objects and applications that collect real-time data (e.g. IoT platforms, data centers, data analytics tools)

Network Infrastructure
SDN for IoT

SDN

- Software Defined Networks (SDN) technology can be applied to improve the efficiency and robustness of network technology supporting the IoT stack
- SDN is based on programmable networks in which data and control plane are separated: control is moved from switches to a controller node, data are handled by every switch
- SDN is based on protocols like Openflow that regulate the communication between controller and switch nodes

IoT/SDN

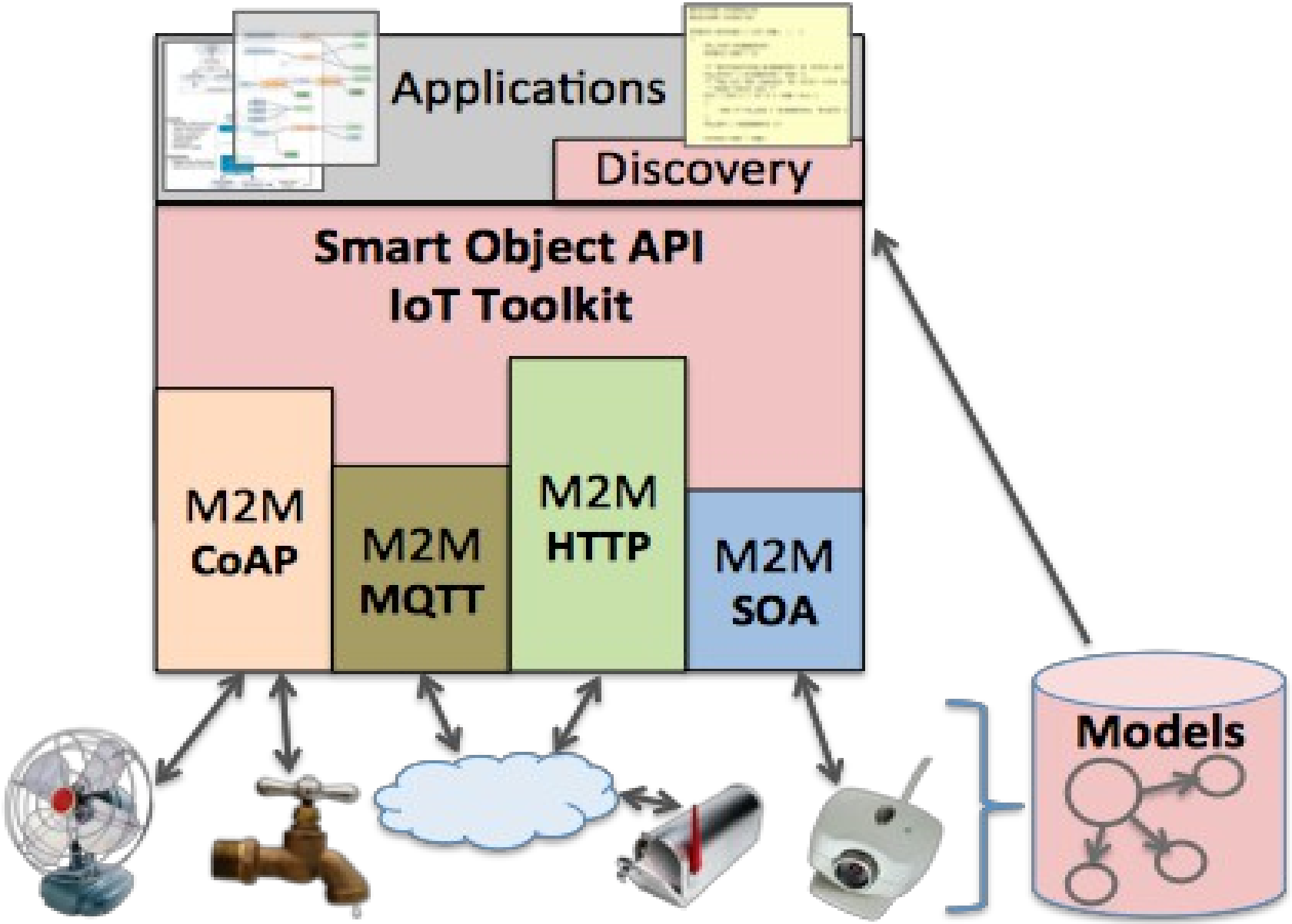
- Openflow controls packet traffic by using flow tables that can be dynamically adapted to the current traffic state
- Flows are defined by informations of packets like IP/MAC addresses, logic/physical port numbers, etc.
- Packets are managed according to the flow table entries (forwarded, dropped, etc), a miss in a flow table is redirected to the controller
- Controller can dynamically update the flow table entries, i.e., update the routing strategies
- Currently used in Data Center, often at the core of data analytics of IoT technology

Software Abstractions in the Cloud

Cloud-based IoT Platforms

- Sensed data are transmitted by smart objects
- To handle massive amounts of data we need to move to the Cloud
  - Connect devices to the cloud
  - Store/retrieve data to the cloud
  - Take decisions
  - Visualize data
  - Actuate from the cloud

IoT Platforms



IoT/ES Platforms

- Oracle IoT
- IBM Blue Mix
- Samsung Artik
- Google Brillo/Weave
- Xively
- Thingworkx
- Fiware
- NodeJs

….an many others

Xively

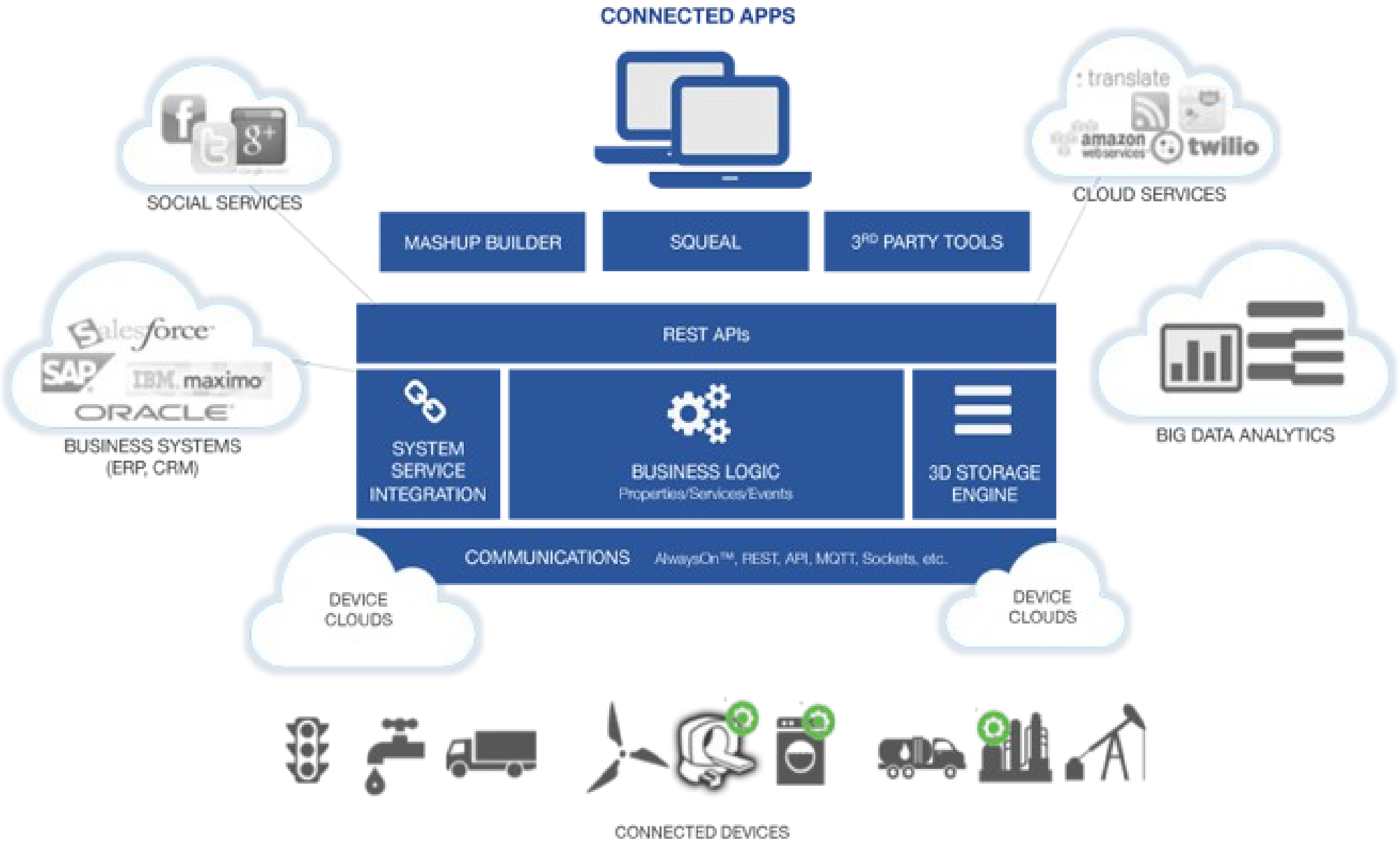
API services for IoT

- Stores data from devices and sensors
- Serve data to other entities
- Supports interoperability
- Security via HTTP and SSL
- Provides RESTful API for different languages (Java, C++, …)

Thingworx

- Graphical IDE for developing IoT applications
- Selection of data streams
- Storage/data query
- Graphical widgets to visualize data
- Data analytics tools
- Customized event handlers (in Java)
- Marketplace for device SDK based on WebSocket

Thingworx Architecture



Google Brillo/Weave (coming soon)

- Interoperability through certification program for developers
- Seamless and secure communication schemas between devices both locally and through the cloud
- Built-in into Android
- Integrated into Google Play services
  - Google Analytics
  - App Engine for developing on the Cloud
  - Google Maps API
  - Android and Wearable API
  - ...

Node.js

- Node.js is a platform based on the JavaScript runtime for building fast, scalable network applications.
- Node.js uses an event-driven, non-blocking I/O model suitable for data-intensive real-time applications that run across distributed devices.

Node.js

- Node.js is a platform based on the JavaScript runtime for building fast, scalable network applications.
- Node.js uses an event-driven, non-blocking I/O model suitable for data-intensive real-time applications that run across distributed devices.

```
var sys = require("sys"),
my_http = require("http");
my_http.createServer(function(request,response){
    sys.puts("I got kicked");
    response.writeHeader(200, {"Content-Type": "text/plain"});
    response.write("Hello World");
    response.end();
}).listen(8080);
sys.puts("Server Running on 8080");
```

Web Server in Node.js

Open Problems

Security and Privacy

- The increase of “smart object dependability” augments the Risk of critical attacks at all levels of the IoT stack (e.g. to smart meters, pacemakers, automation controllers, OS, network infrastructure, software, etc)
- Everything connected in the Internet means potential loss of private data (e.g. electronic devices may reveal custom habits or expose banking information, etc)

Ensure Security Bottom-up

The IoT stack must be tightly integrated with all available network security tools

- Secure booting to authenticate software installed on devices, e.g., TTEs used in smartphone OS
- Device authentication
- SSH/DTLS for securing communication
- Firewalls
- Access control policies for ensuring data privacy

Security and privacy are still open problems

IoT Applications

Top IoT Applications

Smart Parking: Monitoring of parking spaces availability in a city

Structural health: Monitoring of vibrations and material conditions in buildings, bridges and historical monuments.

Noise Urban Maps: Sound monitoring in bar areas and centric zones in real time.

Smartphone Detection: Detect iPhone and Android devices and in general any device which works with WiFi or Bluetooth interfaces.

Eletromagnetic Field Levels: Measurement of the energy radiated by cell stations and and WiFi routers.

Top IoT Applications

Traffic Congestion: Monitoring of vehicles and pedestrian levels to optimize driving and walking routes.

Smart Lighting: Intelligent and weather adaptive lighting in street lights.

Waste Management: Detection of rubbish levels in containers to optimize the trash collection routes.

Smart Roads: Intelligent Highways with warning messages and diversions according to climate conditions and unexpected events like accidents or traffic jams.

Some of our projects

- Android Applications that exploits iBeacon to send notifications to, acquire data from users, to profile clients/visitors (stores, museums, etc)
- Indoor positioning and navigation inside museums/hospitals
- Social networks (whatsapp-like) based on Wi-Fi Direct P2P networks
- Environmental Monitoring sensor network based on low cost optical rain gauges (on going work)

iBeacon

iBeacon uses Bluetooth low energy proximity sensing to transmit a universally unique identifier

The identifier can be used to determine the device's physical location or trigger a location-based action on the device such as a push notification.

One application is distributing messages at a specific Point of Interest, for example a store

Similar to geopush technology based on GPS, but with a much reduced impact on battery life and much extended precision (from a few metres down to a few centimetres).

Flood Monitoring

Networks of Rain gauges for monitoring rainfall intensity in the Monterosso municipality, an area subjected to flash floods

In preparation, short presentation tomorrow afternoon

Conclusions

- Internet of Things:
  a collection of hardware, network technology, software application to build innovative applications by using connected devices
- Challenges:
  - Standardization and Interoperability
  - Programming platforms
  - Performance and energy saving/harvesting
  - Security, privacy, trust
  - Socio-technical issues
  - ...