UNIVERSITÀ DEGLI STUDI DI PADOVA

MIME

Master's degree ICT Internet Multimedia Engineering

Department of Information Engineering (DEI)
Master degree on ICT for Internet and Multimedia Engineering (MIME)

Internet of Things and Smart Cities
03 – System architecture

Marco Giordani (marco.giordani@unipd.it)
Department of Information Engineering (DEI) – SIGNET Research Group
University of Padova – Via Gradenigo 6/B, 35131, Padova (Italy)

UNIVERSITÀ DEGLI STUDI DI PADOVA

MIME

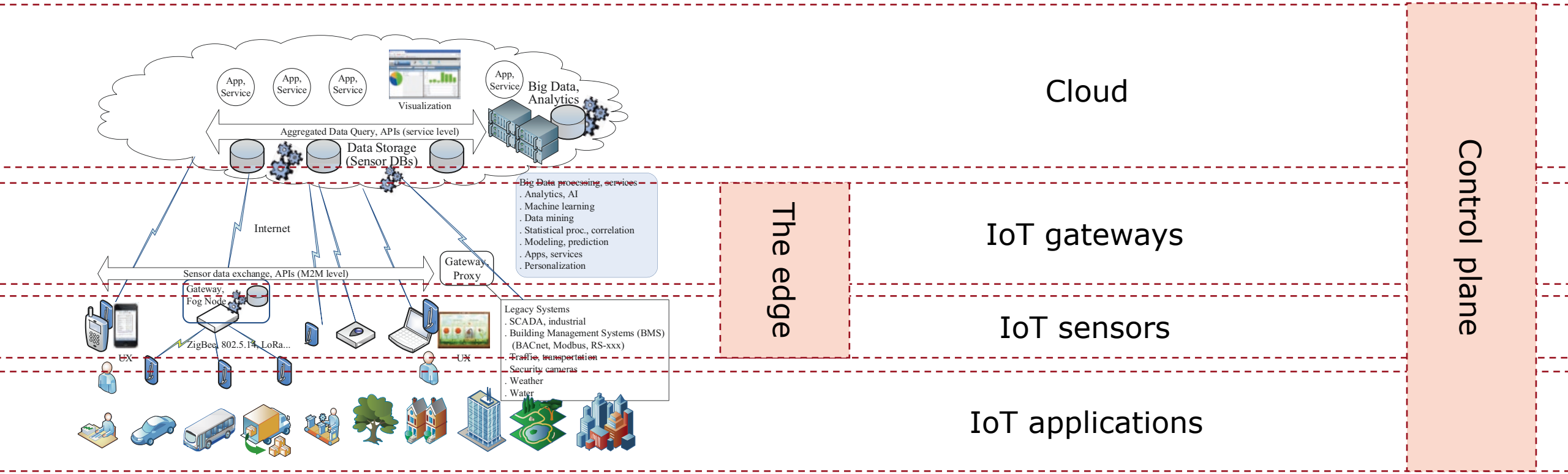
Master's degree ICT Internet Multimedia Engineering

03 – System architecture
Structure and definitions

Marco Giordani (marco.giordani@unipd.it)
Department of Information Engineering (DEI) – SIGNET Research Group
University of Padova – Via Gradenigo 6/B, 35131, Padova (Italy)

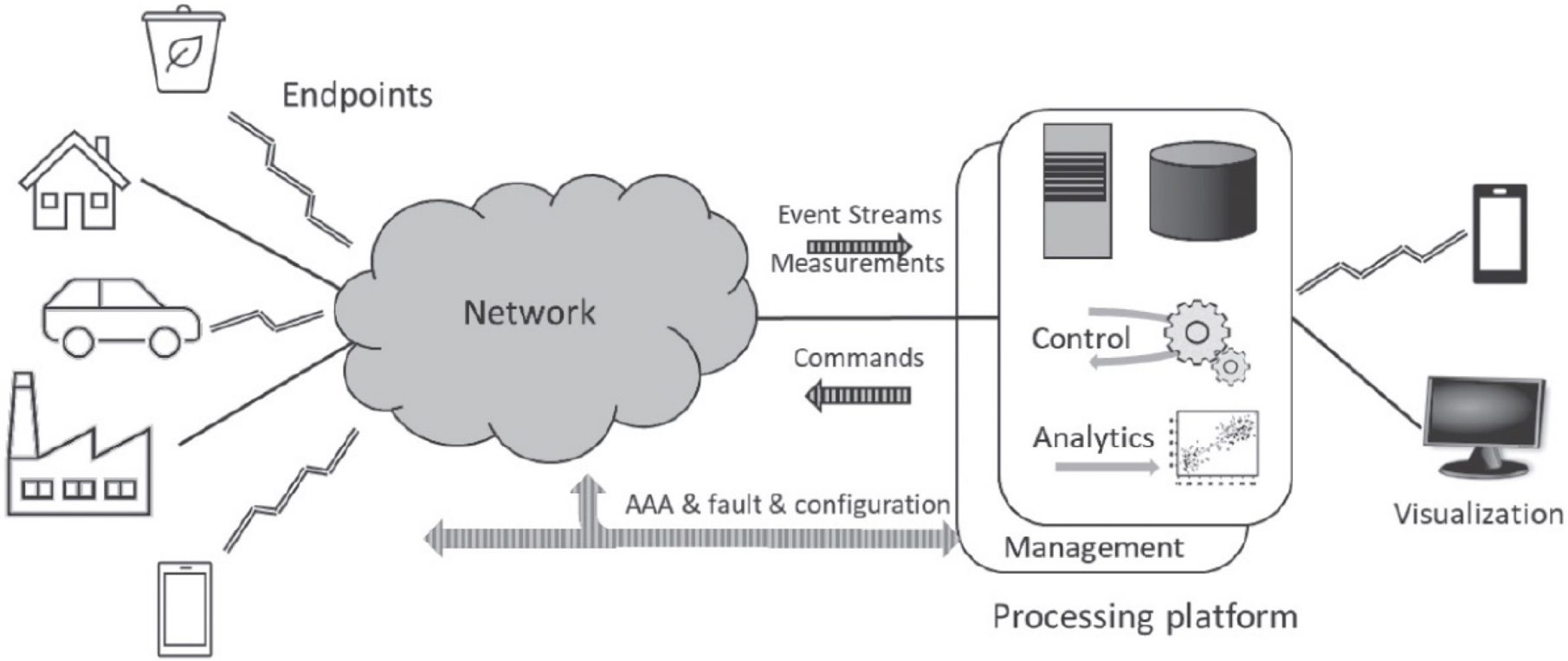
Do you remember?

System architecture



System architecture

System architecture



System architecture

System architecture

- **Client/server**: the information sources explicitly know the destination about the destination of their messages.
- **Publish/subscribe**: the sources send (*publish*) their messages on a logical distribution channel, to which zero, one or more interested destinations may *subscribe*, implementing a o**ne-to-many communication pattern**.
  - It is not necessary to create explicit communication sessions.
- Looking at the information carried by the network, we can identify 3 flows:
  - Streams of **asynchronous** events (generated autonomously).
  - Measurements transmitter **periodically** or **on-demand.**
  - **Commands** coming from the processing platform.

System architecture

Standard reference models

- Properties of a generic IoT system:
  - **Trustworthiness**: availability, resilience, confidentiality, integrity, protection of personal information, safety.
  - **Architectural**: composability, functional and management capability separation, heterogeneity, distribution, legacy support, modularity, network connectivity, scalability, shareability, and unique identification.
  - **Functional**: accuracy, auto-configuration, compliance, content-awareness, context-awareness, "big data" management, discoverability, flexibility, manageability, network communication, network management and operation, real-time capability, self-description, and service subscription.

System architecture

Standard reference models

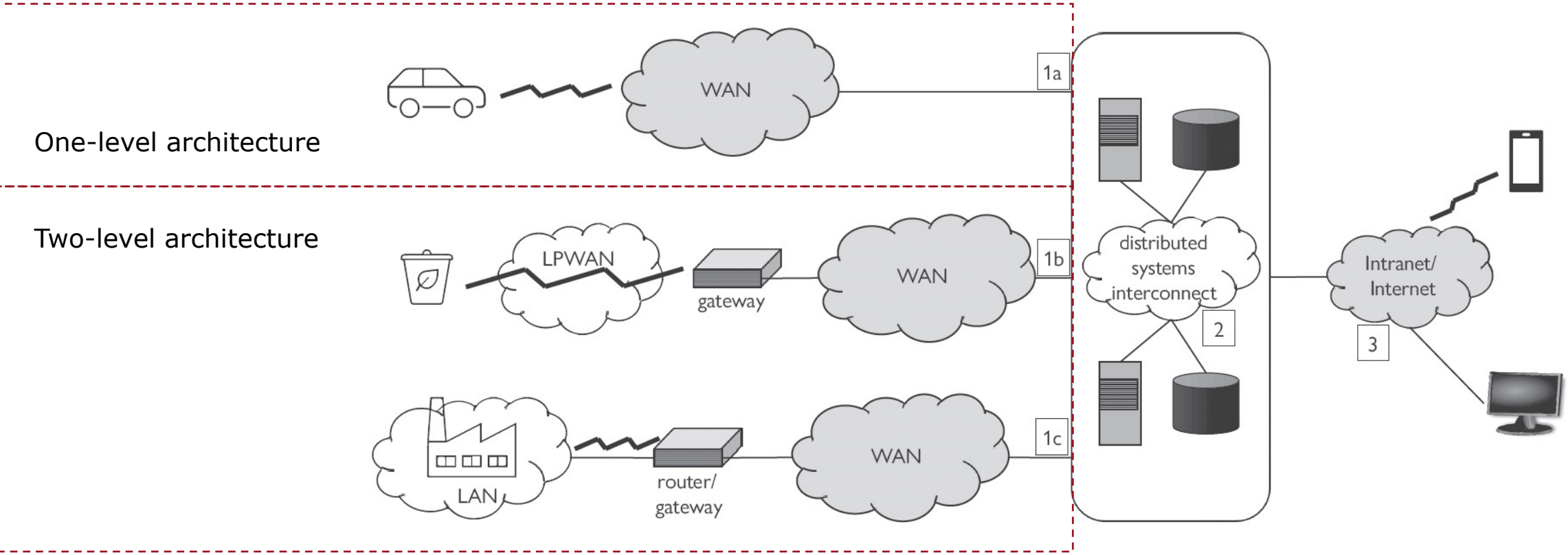
- Six functional domains are created:
  - **Physical Entity Domain (PED):** physically monitored and controlled objects.
  - **Sensing & Controlling Domain (SCD):** sensors, actuators, and adaptation devices bridging the physical and the cyber world.
  - **Operations & Management Domain (OMD):** Operation Support Systems (OSS) + Business Support Systems (BSS); it covers the functions for provisioning, monitoring, assurance, and optimization of the operational performance of the system.
  - **Resource Access & Interchange Domain (RAID):** interfaces through which the services are offered, under suitable access policies, to external entities.
  - **Application & Service Domain (ASD):** services to the end-users interacting with applications, sensors, actuators and externally via the RAID. Use cloud platforms through a portal or Application Programming Interfaces (APIs).
  - **User Domain (UD):** human and digital users, accessing services via general purpose (PCs, smartphones) or specialized (control panels, smart glasses) end-user devices.

System architecture

Network architectures

One-level architecture

Two-level architecture



System architecture

Network architectures

- Three network architectures:
  - **One-level architecture**: the endpoints know about the addressing scheme.
    - Direct IoT connectivity to the Internet protocol stack.
    - Typically implemented with cellular network technology.
  - **Two-level architecture**: the last segment is unaware of addressing aspects (endpoints send messages which are not geographically routable) and a **gateway** handles the communication protocol.
    - End nodes are generally resource and/or energy constrained (e.g., *sensors*).
    - Typically implemented with Low Power Wide Area (LPWA) technologies.
    - Example: LoRa, SigFox, NB-IoT, etc.
  - **Two-level architecture**: the endpoints are not resource-constrained and implement the TCP/IP protocol stack. The traffic is managed by an intermediate **router**.
    - Typically implemented with Ethernet LANs or Wi-Fi, etc.

System architecture

Wide area networks (WANs)

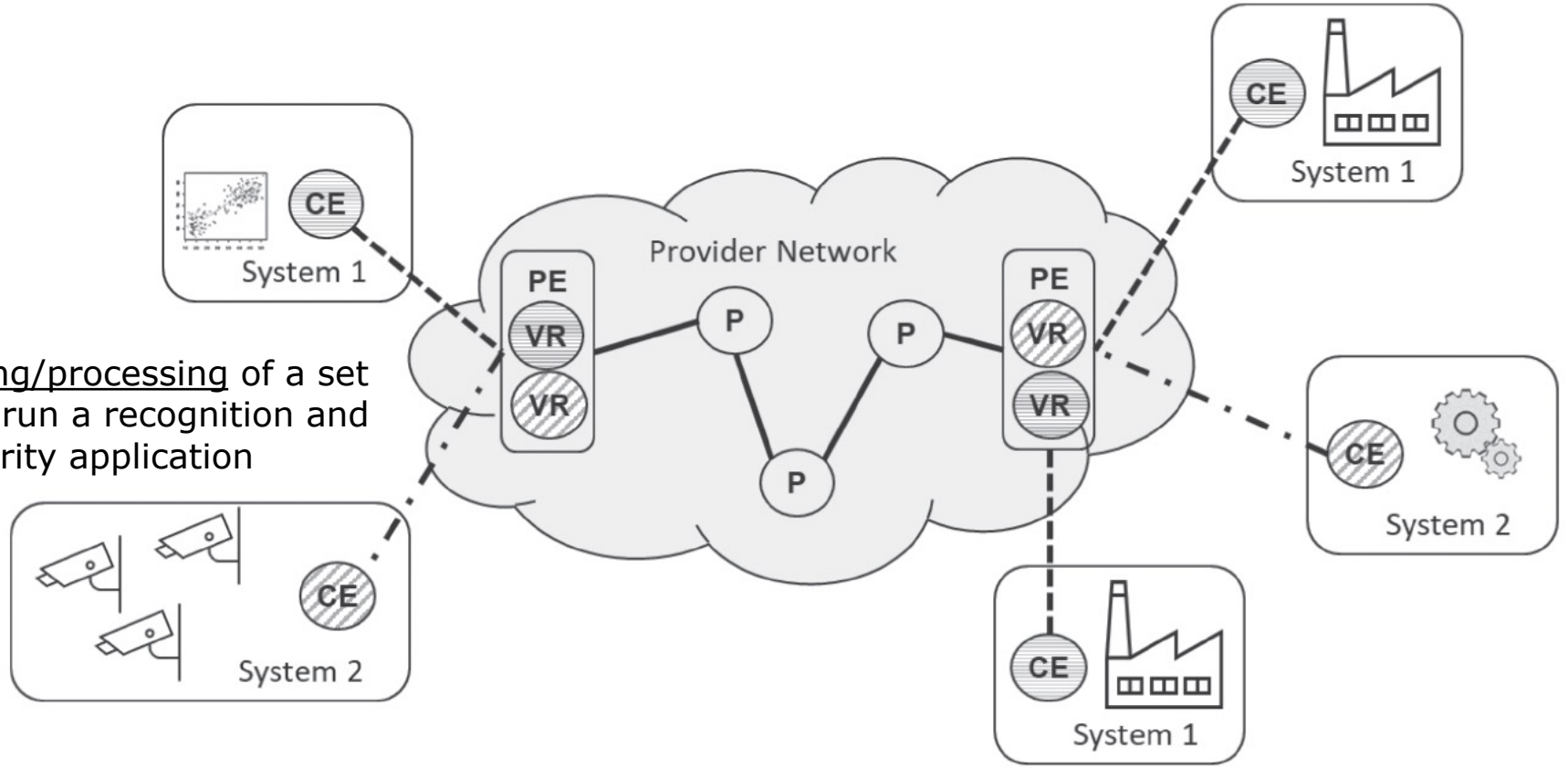
- WANs are wired TCP/IP networks.
- The WAN must be shared among many users and applications (economic reasons).
  - Maintain integrity and data protection: mechanisms to **segregate** and **reserve** the traffic.
- **Virtual Private Network (VPN)**: network that takes resources (*bandwidth capacity, switching and routing functions, firewalls and other security appliances etc.*) and reserves them to a defined group of users/endpoints.
  - Users external to this group cannot communicate directly with the users of the group.
- Users outside of a group cannot communicate directly with users of the group.
- Use **private IP addressing** and **private routing protocol** instances.

System architecture

Wide area networks (WANs)

System 1: data collection from two production sites to feed a proactive maintenance application

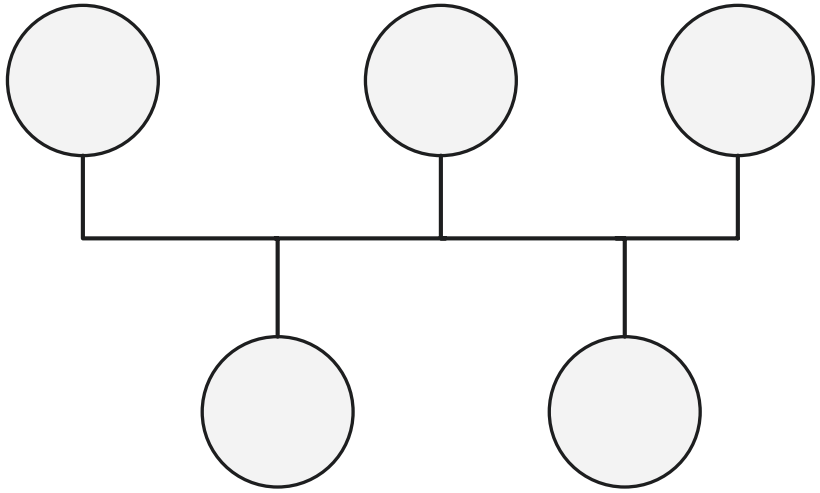
System 2: managing/processing of a set of video camera, to run a recognition and tracking security application



System architecture

Topologies: bus

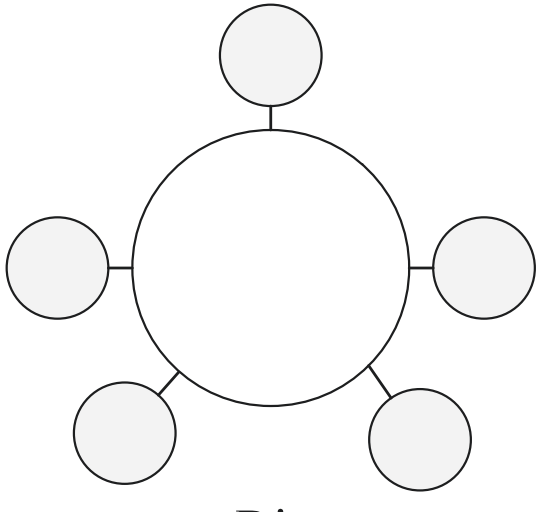
- Every nodes taps into a common medium, signals may **collide with each other**.
- Need to arbitrate who will get the bus.
- The common medium is the bottleneck (to be shared) → **single point of failure.**
- Use CSMA/CD
- Example: (old, 10BASE2, 10BASE5) Ethernet.



System architecture

Topologies: (token) ring

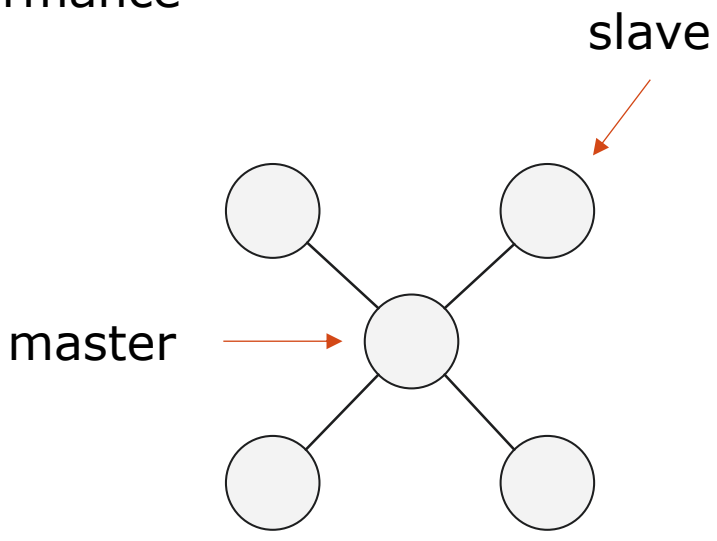
- Nodes are in a ring: receive from predecessor and send to successor.
- Need to arbitrate which node can access the ring (**repeaters** at every link).
- The common medium is the bottleneck (to be shared) → **single point of failure.**
- **Token-ring topology:**
  - A node wishing to transmit waits for the receipt of a token, removes it from the ring, and places its message.
  - Then, the node holding the token passes it along.
  - Eliminates collisions and can increase throughput.
  - Need a mechanism to preserve token integrity and to regenerate it if necessary (e.g., when nodes are powered off).



System architecture

Topologies: star

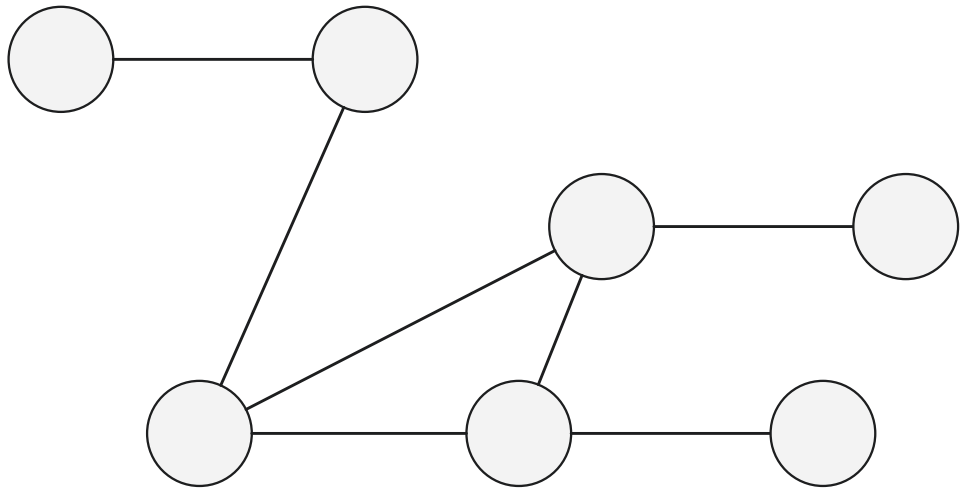
- The **master** is at the center, other nodes are **slaves** linked to the master.
- Slaves communicate via master.
  - Easy to arbitrate among slaves (master decides)
  - Not scalable (the master is the bottleneck)
  - Normally for small networks or that requires predictable performance
  - Master failure shutdowns → **single point of failure.**



System architecture

Topologies: (partial) mesh

- Nodes are arranged according to **connectivity graph**.
  - Each node has a dedicated point-to-point connection to every other node.
  - Number of physical links/connections is n(n-1)/2 (if full-duplex).
  - Reduces traffic forwarding issues (+ no shared medium collisions).
  - Ensures robust and private/secure network.
- Partial mesh: a subset of links representing direct connectivity between a subset of nodes.



Endpoints

Sensors

- A sensor is a device that detects some measurable aspect of the physical world state (stimulus) and converts it to a **processable output**, e.g., an electrical signal.
- There are thousands of sensor types, that can measure almost anything:
  - Pressure, humidity, air quality
  - Fingerprints, facial features, iris patterns
  - Microphones, touch-sensitive surfaces, radars
  - Light, smoke, chemical sensors
  - Temperature
  - Force
  - Positioning
  - …

Endpoints

Types of sensors

- **Fixed**: stable position wrt the network topology (wireless vs. wired).
  - No need to re-authenticate, stable address and routing, simple management, etc.
  - Can be mains powered.
  - Can have permanement bad wireless connectivity conditions, if bad deployment.
- **Mobile**: must be connected to a cellular or LPWA network.
  - May experience a degraded service only occasionally.
  - Limited mobility constraints (e.g., handover).
  - Cannot be mains powered.
- **Nomadic**: can change their physical position, but stay in the same place for the whole duration of a communication session.
  - Must re-authenticate, but do not require tracking functions.

Endpoints

Types of sensors

- **Active**: need to run a current or send a signal to get a measurement (e.g., clocks, strain gauges, radar).
- **Passive**: use the physical phenomenon itself (e.g., piezoaccelerometers, GPS, thermocouples).
  - This does not mean that passive sensors consume zero energy!

Endpoints

Actuators

- Actuators are an output part of the IoT system that **performs direct actions**.
- Actuation provides the means to implement control actions as and when determined by the algorithms or system operators.
  - **Digital actuators**: On-off action.
    - Past the appropriate signal conditioning can trigger relays or change the state of a thing, such as a power switch which can result in turning a light on or off.
  - **Analog actuators**: Produce continuous signals that can be used to drive devices.
    - For example, controlling the speed of a motor or producing sound in headphones.

Endpoints

Gateways

- It **links** sensors and things at the edge with higher levels of system processing hierarchy and the cloud.
- It performs or assists **data collection**.
- It provides **Internet connectivity** to transfer of sensor data to other components.
- **Security** boundary between things with varying levels of security.
- Optional: **data storage**, **event and alert processing**, and **control** (automation).
- Advanced: **analytics**.

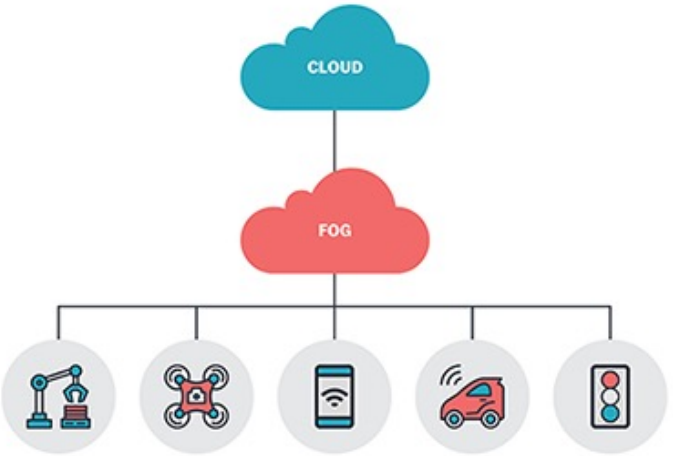
Example: https://www.lantronix.com/products-class/iot-gateways/

Endpoints

Fog nodes

- Etimology: *bring the cloud to the ground*.
- Continuum between the edge and the cloud by placing **additional computation and storage capability closer to the edge.**
- Perform *local processing* (e.g., data reduction, filtering, and front-end analytics) with potentially **lower latency than in the cloud**.
- Fog nodes tend to be more powerful but are otherwise not architecturally or functionally different from the edge gateways.

https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf



UNIVERSITÀ DEGLI STUDI DI PADOVA

MIME

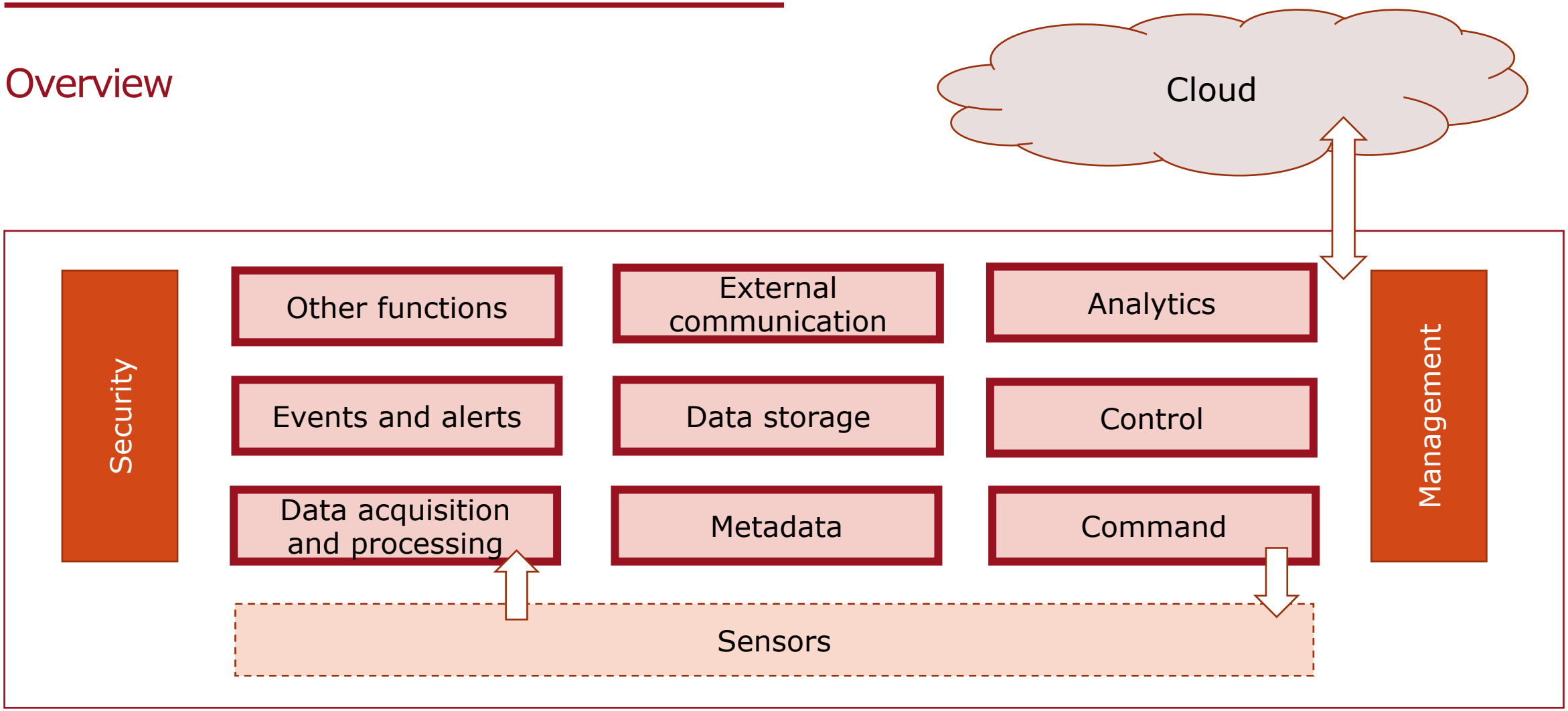
Master's degree ICT Internet Multimedia Engineering

03 – System architecture
Data plane functions

Marco Giordani (marco.giordani@unipd.it)
Department of Information Engineering (DEI) – SIGNET Research Group
University of Padova – Via Gradenigo 6/B, 35131, Padova (Italy)

Data plane functions

Overview



Data plane functions

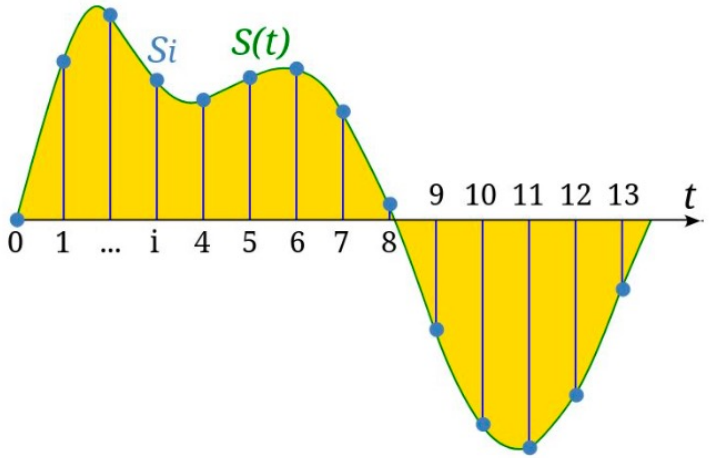
Data acquisition and processing

- In order to get a digital signal from a sensor, we need to account for:
  - Sampling
  - Aliasing
  - Quantization
  - Saturation
  - Hysteresis and non-linearities
  - Calibration
  - Error propagation

Data plane functions

Data acquisition and processing: Sampling

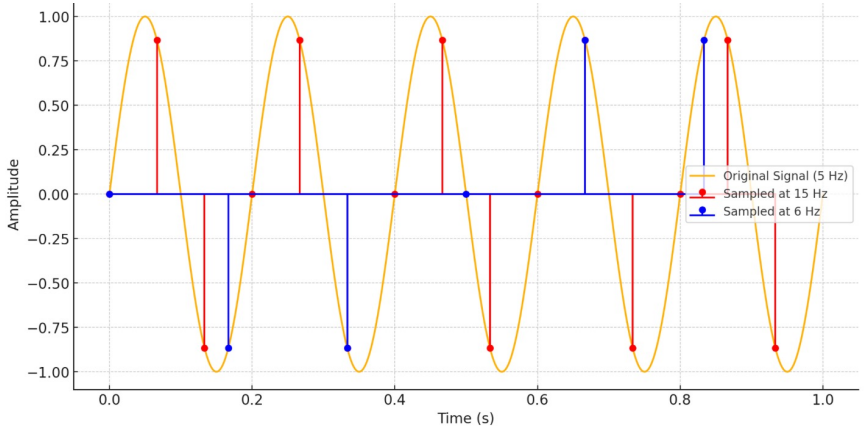
- First, we sample: we take (regular) measurements over time, and represent the signal with those measurements.
  - **Nyquist theorem**: if the sampling is twice as fast as the signal, sampling has no errors.
- Example: If a signal is thought to have a maximum frequency between 1000 Hz and 4000 Hz, which of the following would be the most appropriate sample rate?
  - a. 500 Hz
  - b. 8000 Hz
  - c. 9000 Hz
  - d. 24000 Hz



Data plane functions

Data acquisition and processing: Aliasing

- If we do not sample "enough," we have **aliasing**: patterns in the sampled version.
  - A high frequency component in the frequency spectrum of a signal takes the identity of a lower frequency component in the same spectrum of the sampled signal.
  - This falsely estimated signal will be indistinguishable from (i.e., be an "**alias**" to) another signal having that true lower frequency.
- There are tools to avoid this, but we need to know it is an issue.
  - Example: 15 KHz vs. 6 KHz sampling.

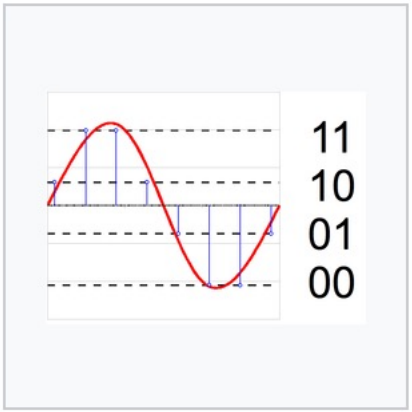


Data plane functions

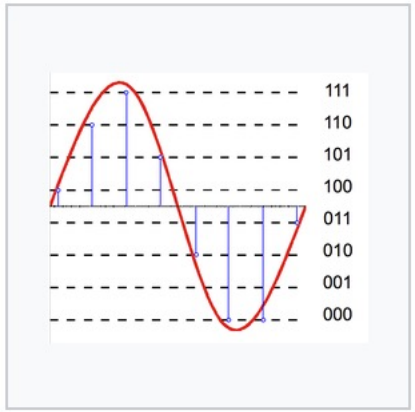
Data acquisition and processing: Quantization

- Digital values are finite: we cannot represent continuous values precisely!
- Quantization always introduces an error, which is the **sensor's natural error**.

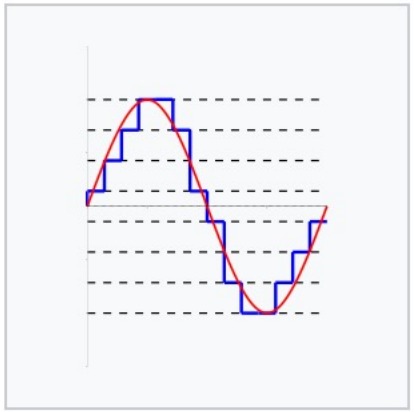
2-bit

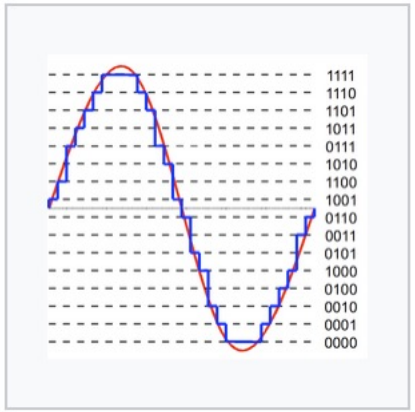


3-bit

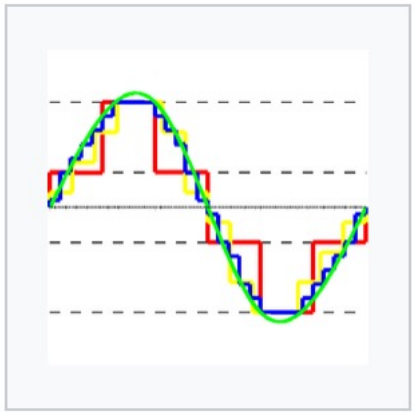


4-bit





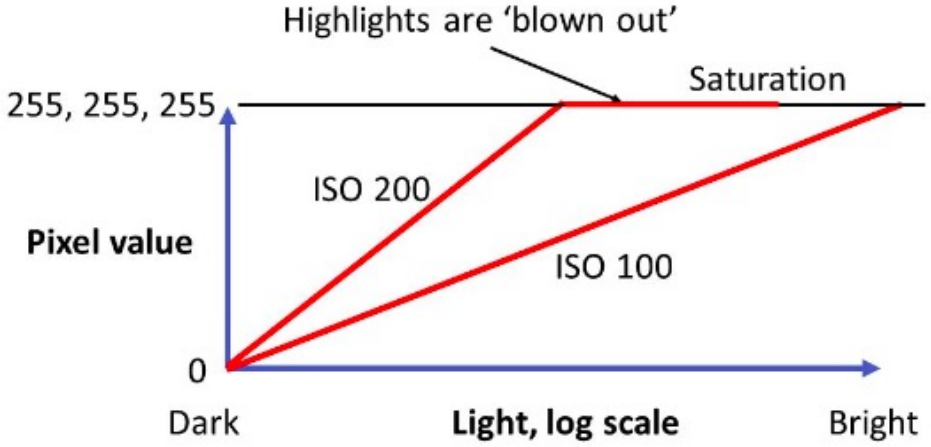
comparison



Data plane functions

Data acquisition and processing: Saturation

- Sensors often have an **upper limit**, which is either physical or dictated by the maximum quantization level: in this case, we have saturation (all higher values are mapped to the same output)
- This is what happens when you take a picture with high exposure: all lighter points are bright white!



Data plane functions

Data acquisition and processing: Hysteresis

- Sometimes, a sensor (e.g., a thermometer) takes a while to track the process because the **physical nature of the measurement is slow** → Hysteresis error when the delay of sensor measurements depends on the magnitude of the change.
- Example: When you apply pressure to a sensor and then release that pressure, the sensor's output should in theory return you to the same value it started at before that specific pressure cycle. However, in reality there are often small differences between the initial value and the final value after a pressure cycle.
  - From 0 to 100 psi
  - 0 psi → 0 V
  - 100 psi → 2500 V
  - 0 psi → 0.05 V
    - **0.02% / 1 psi hysteresis**

Data plane functions

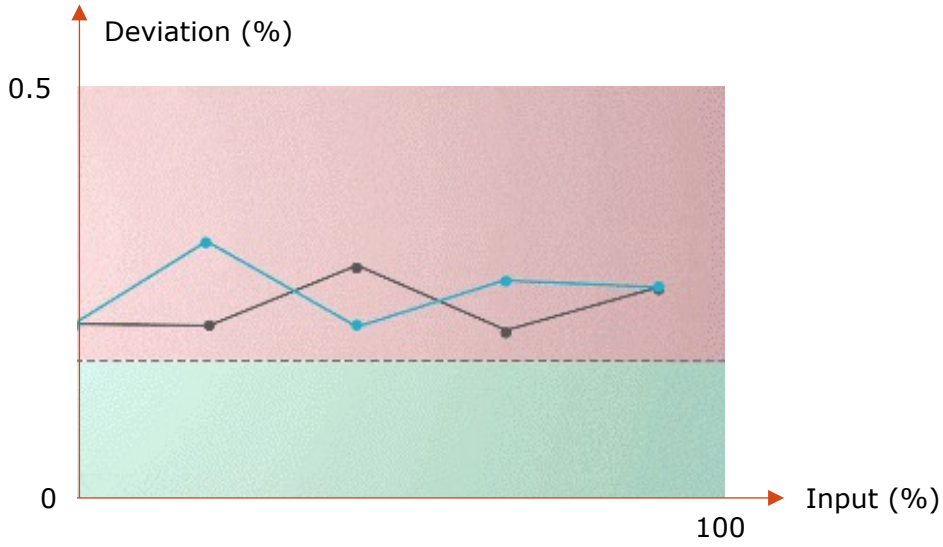
Data acquisition and processing: Non-linearities

- Ideally, sensors should also have linear responses: in reality, there are often **non-linear effects** that we need to adjust for.
- Example: A linear sensor's output will change by equal amounts for equal pressure changes across its entire range.
  - From 0 to 10 bar
  - 0 V at 0 bar pressure
  - 5 V at 10 bar pressure
  - But at 3 bar pressure, it outputs 4.2 V instead of 3.75 V.
    - **≤2% non-linearity of full scale**

Data plane functions

Data acquisition and processing: Calibration

- Sensors often require **calibration** to give proper results:
  - Bias is a constant error term in one direction.
  - Distortion is due to non-linearity.
- By measuring a series of known quantities, we can compensate for these issues (however, external factors such as temperature can throw off calibration!)
- Example: Using the data from the **calibration sheet**, we see that the deviations are all less than the maximum deviation allowed of 0.5%.



Data plane functions

Data acquisition and processing: Error propagation

- Calculations can change the error of a measurement, as the errors are also considered in the function you compute.
  - Non-linear functions can have weird effects with measurement errors.
  - Derivation (e.g., computing velocity from position measurements) is brittle: since the measurements are close together, noise can have a huge effect.
- Integration/averaging are robust (rely on many points, and noise is compensated).
- Example: Calculating heat index
  - Temperature sensor: measured (30°) with uncertainty ±0.5
  - Humidity sensor: measured (70%) with uncertainty ±2%
  - Heat index = $T + 0.33 \times RH - 0.7 = 52.4$
    - Uncertainty with error propagation: 0.828

Data plane functions

External data communication

- Sensors data may need to be shared with external parties, such as other peer nodes at the edge, fog, and cloud levels.
  - Messages exchanged between IoT nodes may be delivered using **push** or **pull** modes.
    - Pull: a message is **explicitly requested** from the source (server) by its recipient, an IoT client.
    - Push: the data source and the gateway send data **when the specified conditions are met**, such as arrival of the data sampling time mark, or exceeding of the measurement threshold.
  - **Publish/subscribe system**: data sources "publish" data to a known place, and authorized clients can "subscribe" to data of interest and receive related messages.
    - Publishers do not need to know the identity or number of their receivers (subscribers), and the receivers do not need to know the identity or state of their publishers.

Data plane functions

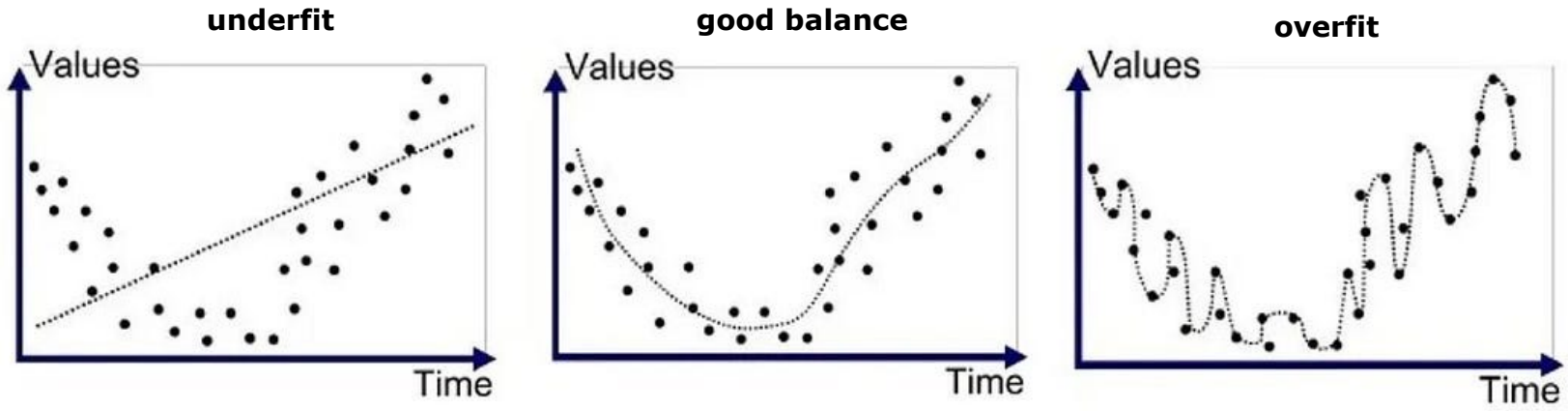
External data communication: how to share sensors' data?

- Some processes (e.g., an electricity meter, a room thermostat) do not need to be sampled very often.
  - If we only care about cumulative measures (how much electricity was consumed in total), we do not need frequent samples!
  - On the other hand, faster processes need frequent samples, as process can change faster.
- **Why don't we transmit all the time?**
  - There are several reasons why we cannot transmit all the time, but we will concentrate on two of them in this first part of the course:
    - Limited bandwidth
    - Limited energy
  - Each transmission (and, in fact, each activity the sensor performs) costs energy, as well as blocking the wireless channel for other transmissions

Data plane functions

External data communication: some solutions to send sensors' data more efficiently

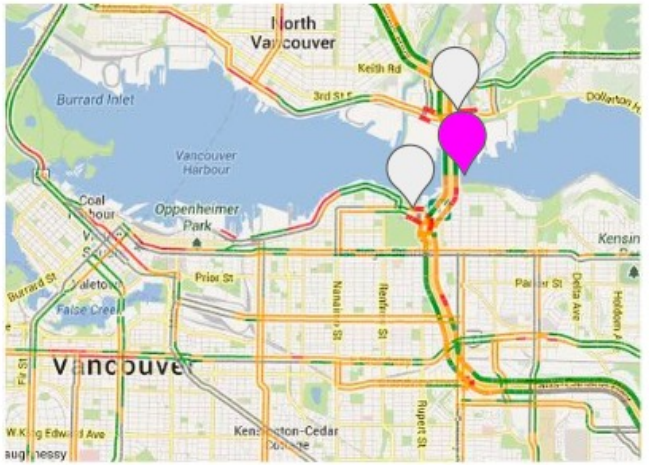
- **Interpolation**
  - Interpolation is the mathematical tool to infer missing data from sparse samples. We have to make some assumptions on the nature of the process to be able to interpolate: if we have the correct model, we can send a lot fewer data samples!
  - Interpolation can also be dangerous: we can **underfit** (select a model that is too simple) or **overfit** (select a model that is too complex and ends up following random noise)



Data plane functions

External data communication: some solutions to send sensors' data more efficiently

- **Spatial correlation**
  - Data are usually correlated in space:
    - Traffic on either side of a bridge
    - Temperatures close together
    - There can be easy natural relations (points close together are similar) or structural relations (connected points are similar)
  - Since measurements have errors, correlations help us **improving the quality of estimates**: we can make statistical calculations and estimates to figure out what is happening in points (like the purple one on the bridge) where we have no sensors, or refine our estimates.



Data plane functions

External data communication: some solutions to send sensors' data more efficiently

- **Temporal correlation**
  - Data are also correlated in time:
    - Traffic in one spot
    - Temperature measured by one sensor
  - The relation here can be more or less complex, and depend on space as well (e.g., line at a traffic light), but we can also exploit this to get better data with fewer transmissions.
- **Correlations can be misleading and dangerous!** When relying on statistical information to reconstruct missing data or make decisions, we need to be aware that random chance is always a possibility.

Data plane functions

External data communication: some solutions to send sensors' data more efficiently

Data plane functions

Local control

- Provided in systems where latency is critical and/or edge autonomy is required.
- Local control enables the edge node to **execute pre-defined control sequences** when a specified condition occurs on some combination of local data.
  - **If This Then That**: scenes where lights and heating/cooling are adjusted accordingly when users are leaving or approaching their homes.
- **Node-RED**: provides a graphical user interface to construct data flows with the functional processing modules and actuation output stages.
  - In the extreme case of temperature, the script triggers the device shutdown.

Data plane functions

Data storage

- Data storage for the acquired data may be provided by the **gateway**.
- The existence of local storage provides the flexibility to conserve network bandwidth and to reduce the load on cloud resources by **selectively reporting data that meet certain conditions**.
  - Depending on the nature of the physical thing being measured, only some of the data points may be of interest. E.g., successive minor changes may be of little interest, but those that exceed a comfort guard band or change by a significant percentage or amount from prior readings are.
  - Qualifying readings may be defined as events and reported only when they occur, with raw readings stored in the local database should they be required for auditing or subsequent analysis.

Data plane functions

Edge analytics

- Edge analytics for the data may be provided by the **gateway** or the **fog node**.
  - Proximity to data sources: low latency
  - Ability to process high-frequency data samples (at the rate of the sensor)
  - Conservation of network bandwidth by not sending data to the cloud.
  - Can operate even in disconnected mode.
  - Analytics generally require AI/ML models:
    - Insufficient computational resources at the edge.
    - No data aggregation of edge/local data.
- IoT systems should be implemented using design tools and practices that facilitate flexible allocation of functions (in tandem between edge and cloud).

Data plane functions

Analytics placement

- Optimal placement of data and processing functions is an age-old tradeoff in distributed systems.
- It is based on whether it is more cost-effective to move the data to the computation or to move the computation to the data, depending on:
  - Availability and cost of bandwidth
  - Latency requirements for time-critical operations
  - Local autonomy of operations, including disconnected mode
  - Security and data control or privacy concerns
  - Cost and complexity of managing distributed nodes with computing and storage
  - Available energy resources
  - Available computational capacity
  - …

UNIVERSITÀ DEGLI STUDI DI PADOVA

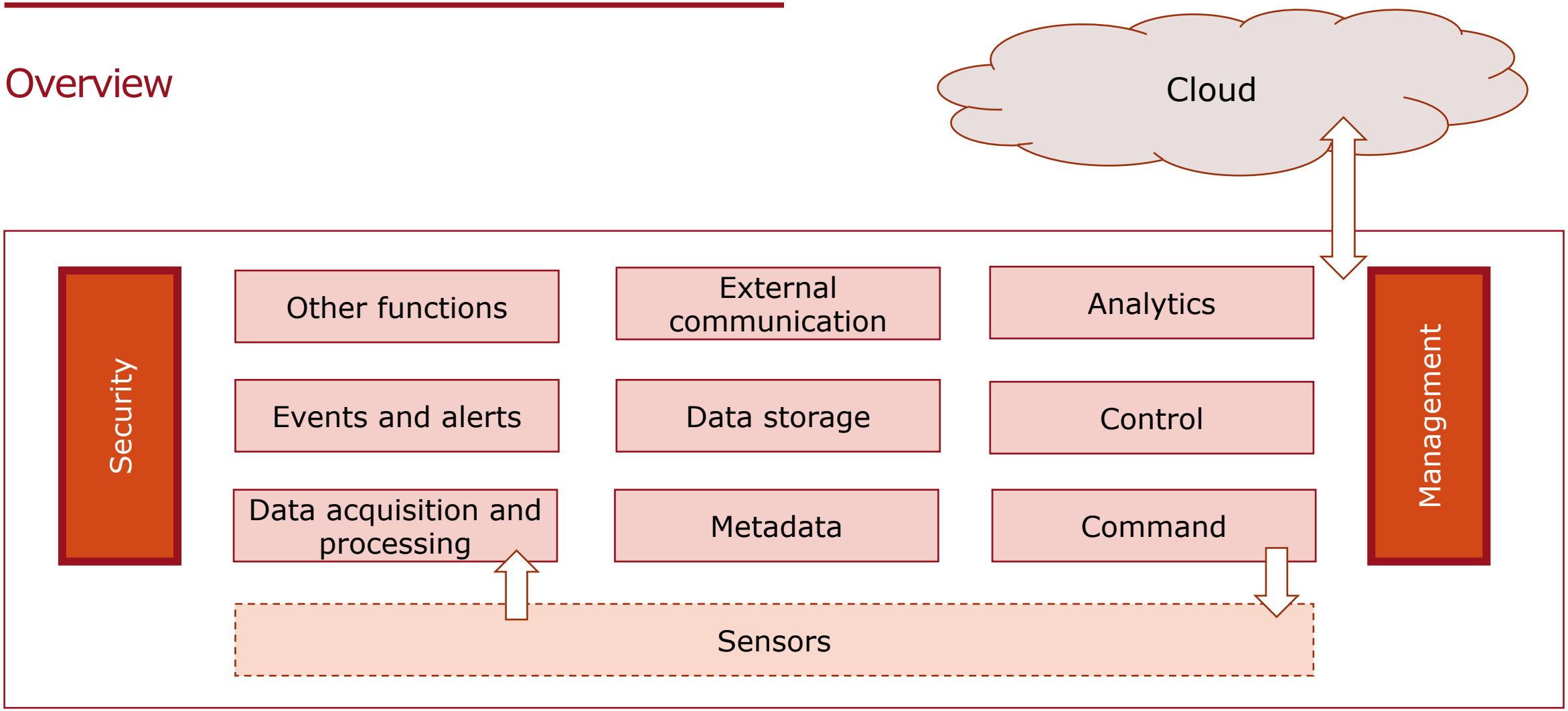
Master's degree ICT Internet Multimedia Engineering

03 – System architecture
Control plane functions

Marco Giordani (marco.giordani@unipd.it)
Department of Information Engineering (DEI) – SIGNET Research Group
University of Padova – Via Gradenigo 6/B, 35131, Padova (Italy)

Control plane functions

Overview



Control plane functions

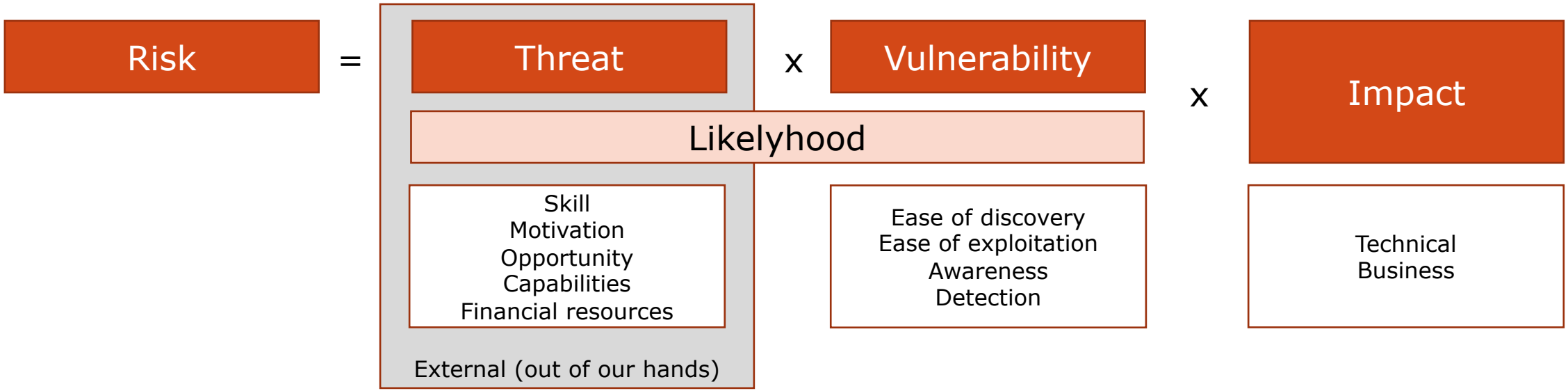
Overview

- Fully functional gateways may be fairly sophisticated computational and communications nodes that host and execute a variety of services.
  - They need to be properly **secured** and **managed**.
- Main control functionalities:
  - Fault management (troubleshooting, error logging, and recovery)
  - Remote monitoring, control, administration, and diagnostics
  - Remote firmware and software updates
  - Security updates
  - Metering (network bandwidth and software usage) and supervision
  - Provisioning and authentication

Control plane functions

(Cyber)security

- The first step is to determine the security risk.
  - Identify, quantify, and prioritize the risks for the system, and find a balance between costs and potential losses from residual risks.
- Model to quantify risk: **vulnerability**, **threats**, and **impacts**.



Control plane functions

(Cyber)security

- **Vulnerabilities**: can originate from the design of the system architecture, technology, and protocols, but more often emerge in the implementation (e.g., conding flaws) and in the **operation phases** (e.g., configuration errors, use of weak credential, unparched devices, unexpert use, etc.).
- **Security measures** for the diligent user:
  - Partitioning the system into zones (characterized in terms of security levels).
  - Adotp effective authentication protocols, tailored to the different zones
  - Provide cryptographic protection of data
  - Install equipment from truster providers.
  - Make and install frequent patches and workarounds.

Control plane functions

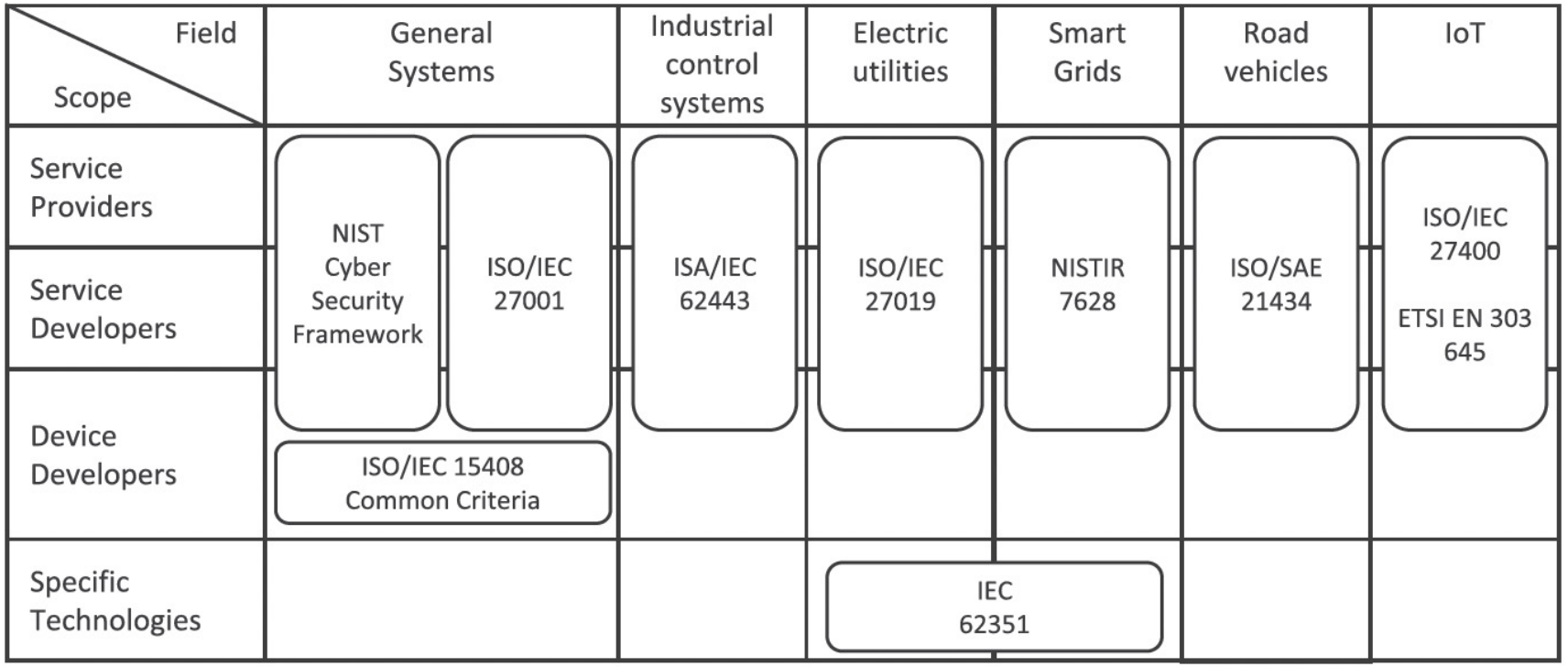
(Cyber)security

- **Threat**: an IoT network is a possible point of attack if the endpoint is unguarded and is positioned in an easily reachable position.
  - **Tampering**: intentional alteration of measurements and event notifications, or intentional installation of malwares on the endpoint.
  - Software **patching** process many be slow (hours or days), making devices vulnerable.
  - **Malwares** can be injected during remote firmware updates.
- End devices must be treated *a priori* as the weak points of the architecture, and constitute **a separate security zone**.
  - Best approach: anti-tampering self-disconnection operation if security risk is detected.

Control plane functions

(Cyber)security

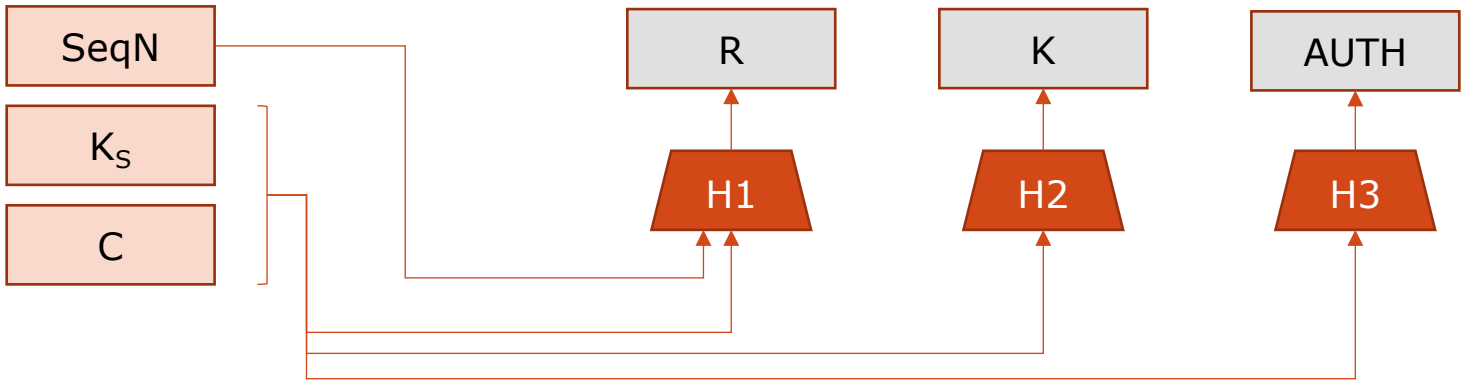
- IoT-oriented and industry specific **standards** have ben developed for security.



Control plane functions

Provisioning and authentication

- The lifecycle of an enpoint begisn with its **provisioning** in the network.
  - The process of giving the endpoint the configuration data and the ability to present itself.
  - **Authorization**: checking the rights of access to the network.
  - **Application-level authentication** (if the network is shared among applications).
  - **Authentication**: verification of the identity ("I know your name, is it you?").
    - **Challenge-response protocol** employing secret keys.



Control plane functions

Provisioning and authentication

- Two parties share a secret key K_S.
- The server generates a randon challenge C, and computes, using the harsh functions H1, H2 and H3, the following elements:
  - R: the response expected by the client
  - K: a key used to encode parts of this exchange
  - AUTH: a code that demonstrates that the server also possesses K_S.
- The server sends $E_K\{SeqN\}$ (i.e., the ciphering algorithm using K, AUTH, and C).
- The client
  - Computes K, decodes SeqN, and understands if this is a replayed authentication request.
  - If this is a fresh request, it computes AUTH and verifies if the server knows K_S.
  - Computes R, and sends it back to the server.
- If the server receives a correct response, it verifies the identity of the client.

Control plane functions

Provisioning and authentication

- Problem: this method is based on the fact that only server and client know K_S.
- However, distributing the secret key is challening.
  - We cannot assume that endpoints hold the key from the beginning (firmware installation).
- Possible solution: build a **chain of thrust** via a **Trusted Third Party (TTP)**.
  - The endpoint stores the public key of the TTP at the firmware installation.
  - With this key, the endpoint receives the secret key from the authenticated TTP.
  - The server uses different credentials from those used to authenticate the endpoint.

Control plane functions

Provisioning and authentication

- Problem: A close device, impersonating the real device, distributes credentials.
- Possible solution: **Physically Unclonable Functions** (PUF).
  - PUFs can generate outputs with the same properties as one-way hash functions, but their input/output function depends on random characteristics of the component occurring in the fabrication process.
  - It works if K_S is embedded in the hardware → it is impossible to be extracted even if the device is tampered with.
  - PUFs require performing an initial enrollment process, where a set of input/output pairs is generated and stored securely for the upcoming authentication procedure.

Control plane functions

Metering and supervision

- **Supervision**: set of activities to guarantee that the performance and availability levels required by the IoT applications are maintained.
  - The last segment (link connecting the endpoints) is the most critical part.
    - The wireless connectivity with the endpoints may be discontinuous due to **environmental factors** (e.g., rain, obstructions, etc.), or **interference.**
    - Use **metering functions** to measure quality of the connectivity (e.g. SNR)
    - Some applications are resource-constrained, and endpoints may be rarely contacted: “keepalive” messages for metering if the state of the node remains unknown for too long.
  - The supervision of gateways is less critical.
    - Continuous connectivity is assumed between the gateways and the core network.
  - In case of cellular acces (direct-to-endpoint connectivity with no gateway) supervision is not needed since this job is carried out by the standard cellualr network.

UNIVERSITÀ DEGLI STUDI DI PADOVA

Master's degree ICT Internet Multimedia Engineering

03 – System architecture
QoS and performance

Marco Giordani (marco.giordani@unipd.it)
Department of Information Engineering (DEI) – SIGNET Research Group
University of Padova – Via Gradenigo 6/B, 35131, Padova (Italy)

QoS and performance

Delay

- **One-way delay (OWD)**: time between the transmission of the first bit of a packet at an interface and the reception of the last bit of the packet at the other interface.
  - Requires perfect time syncrhonization between the endpoints.
  - Time reference: **Network Time Protocol (NTP)**.
- **Two-way delay (TWD)**: time between the transmission of the first bit of a packet at an interface and the reception of the last bit of the packet at the same interface, after being reflected by the system at the other end, net of the End System Delay.
  - No synchronization, but it needs to ESD (processing time at the remote node).
  - We have that **Round Trip Time (RTT)** = TWD + ESD.

Warning: OWD ≠ TWD/2

QoS and performance

Delay

- **Packet-delay variation (PDV)** (~jitter): based on previos OWD/TWD measures.
  - PDV = $|OWD_i - OWD_{i-1}|$
  - PDV = $|OWD_i - mean\{OWD\}|$
  - PDV = $|OWD_i - min\{OWD\}|$
  - PDV = $|OWD_i - OWD_1|$
  - …

QoS and performance

Delay

- Problem: These delay metrics do not consider the MAC delay (for channel access).
- **End-to-end delay (E2E):** it consists of (at least) three components:
  - **MAC delay** (under the control of the system designer)
  - **Access link and gateway delay** (under the control of the system designer)
  - **OWD** (depends on the WAN)

QoS and performance

Delay and "real-time"

- The expression "real-time" may have different interpretation:
  - Synchronization to a common time reference.
  - Responding fast.
  - **Responding within predictable response times.**
- **Hard real-time:** if the response exceeds a limit, the system does not operate well.
  - Example: respond within 5 ms
  - CASE 1: The application always respond in 4 ms
    - Average delay: 4 ms; Failure rate: 0%
  - CASE 2: The application respond in 1 ms for 99.9% of the times, and in 6 ms in 0.01%.
    - Average delay: 1.005 ms; Failure rate: 10^{-3}
- **Soft real-time:** it involves some response time percentiles.

QoS and performance

Packet loss

- In an IoT network, loss may occur if:
  - The buffers of a node overflow.
  - Transmission errors (negligible in wired LANs, dominant in wireless LANs).
  - **Delay much larger than expected may be considered as a loss.**
- How to have realistic measures of packet loss?
  - Long-lasting measurements: averaging of data (no indications on the short period).
  - Short measurements: no statistically meaningful data.
  - Intense bursts: too intrusive.

QoS and performance

Capacity

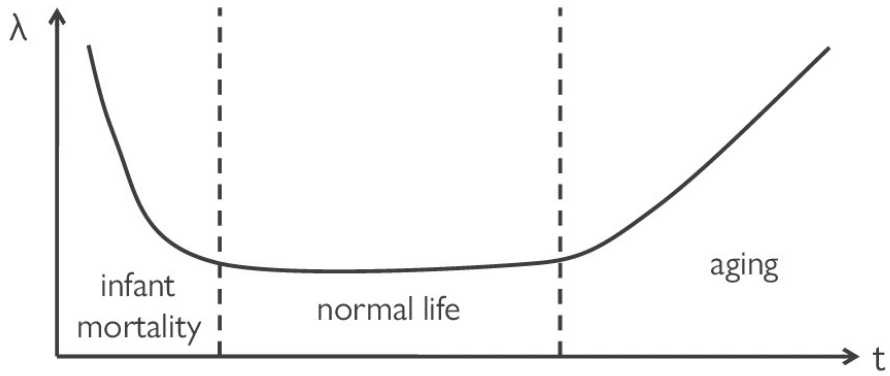
- Hardly a problem, as applications are not (in general) bandwidth demanding.
  - For some high-end applications (e.g., telemedicine, IIoT, automotive, etc.), capacity is generally less important than other metrics such as **reliability**.
  - Example: metering application
  - Collect 1M samples (of 128B) every 15 minutes
  - Average throughput: 1’000’000 x 128B x 8 (bit) / (15 x 60) = 1.3 Mbit/s
  - If the same measurements take place as unsolicited events from the endpoint in the same second, the link capacity shall be around 1’000’000 x 128B x 8 (bit) = **1 Gbit/s**

QoS and performance

Reliability

- R(t): probability that a system is working as expected at a certain time t.
  - For most systems, the early phase of "high **mortality**" is short and occurs during tests.
  - The system is generally replaced for obsolescence as the **aging** phase approaches.
  - In "**normal lif**e", most systems shall be reliable.
- R(t) is **memoryless**: the residual life of failure does not depend on how long the system has been working → the system **does not age during "normal life**."

BATHTUB failure rate curve



QoS and performance

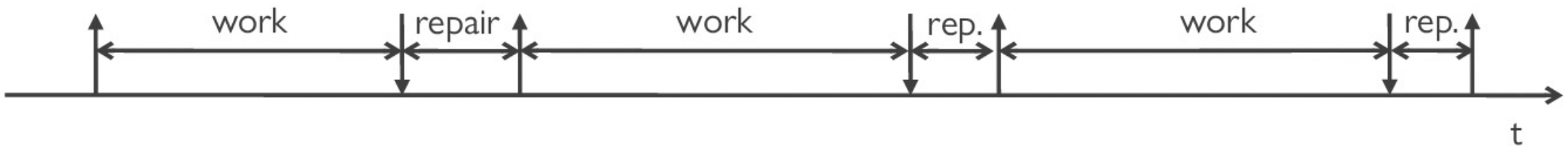
Reliability (proof)

QoS and performance

Availability

- Probablity that the system is working when observed at a random time instant.
  - We assume that the system can be periodically repaired if needed and put back in service.
  - **Mean Time Between Failures (MTBF)**
  - **Mean Time To Repair (MTTR)**

$$A = \frac{MTBF}{MTBF + MTTR}$$



QoS and performance

Availability

- For memoryless systems, MTBF = **Mean Time To Fail (MTTF)** = $1/\lambda$.
- Example: prevention of fires in a forest
  - N sensors that monitor fires in the forest
  - Sensors fail at rate λ.
  - The prevention works if x% of sensors are active and alive.
  - Period of effectiveness of the system (T): $Ne^{-\lambda T} = xN \rightarrow T = {-\ln x}/{\lambda}$

UNIVERSITÀ DEGLI STUDI DI PADOVA

Master's degree ICT Internet Multimedia Engineering

03 – System architecture
Energy constraints

Marco Giordani (marco.giordani@unipd.it)
Department of Information Engineering (DEI) – SIGNET Research Group
University of Padova – Via Gradenigo 6/B, 35131, Padova (Italy)

Energy consumtpion

Introduction

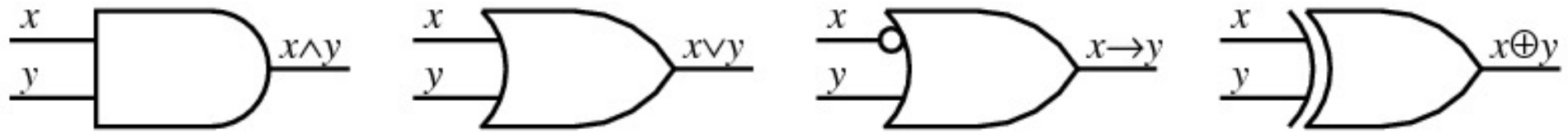
- Most sensors and components use **electrical energy**, i.e., they exploit the movements of electrons across a conducting medium.
- Powering is a very important issue, and depends on the endpoint capabilities and the network architecture procedures.

Powering	Wireline	Wireless	Powerline
Mains / manually recharged	Field buses	Smartphones	Smart grid devices
Battery		Most sensors	
Data line	Videocameras		
Harvesting		Environmental sensors	

Sources of consumption

Computing

- **Computing** requires a certain energy for each cycle, corresponding to one basic operation. Some operations (e.g., sinusoidal functions) require multiple cycles.
  - Energy consumption is usually **a function of the number of cycles required.**
- If a device has a certain clock frequency, we can also compute the power it requires to perform calculations.
- Computing energy can be formalized based on the **number of logical operations that are required to perform**: AND, NOT, OR, XOR.

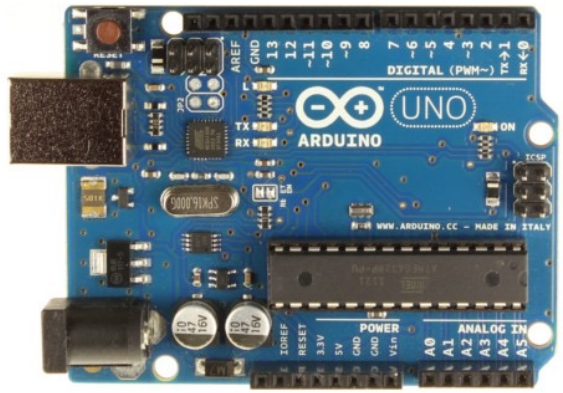


Sources of consumption

Boards and PLCs

- **Boards and PLCs** are standard circuits used in industrial automation. Larger IoT nodes also use Arduinos or other pre-made boards
- Smaller, more efficient nodes are often designed ad hoc, to use as little energy as possible. In general, IoT nodes have limited computational capabilities

Example: Arduino-UNO
The Arduino Uno is a microcontroller board based on the ATmega328. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started



Sources of consumption

Communication

- Digital communication usually requires more steps than just sending a wave, and each **consumes energy**:
  1. Encoding and error protection
     - Use known redundant codes: the simplest example is the parity check bit.
     - Decoders are often complex, but can correct errors in transmission and recover the original packet. However, complexity is paid for in energy
  2. Preambles for synchronization and channel estimation
  3. Modulation from the baseband digital signal to the passband analog signal
  4. **Transmission** (power is applied to the antenna) (*not required for reception*)
     - IoT transmissions are usually slow (low bitrate), but distance plays a large role.

Sources of consumption

Communication (transmission)

- 3GPP TS 38.806 provides a formula to calculate the maximum data rate **for cellular networks** mainly depending on the number of spatial layers and the number of carriers that an end-user device supports.

$$\sum_{j=1}^{J}\left(v_L^{(j)} \cdot Q_m^{(j)} \cdot f^{(j)} \cdot R_{max} \cdot \frac{N_{RB}^{(j)}}{T_s} \cdot \left(1 - OH^{(j)}\right)\right)$$

Sources of consumption

Communication (transmission)

$$\sum_{j=1}^{J}\left(v_L^{(j)} \cdot Q_m^{(j)} \cdot f^{(j)} \cdot R_{max} \cdot \frac{N_{RB}^{(j)}}{T_s} \cdot (1 - OH^{(j)}\right)$$

- J: number of aggregated component carriers in a band.
- v: number of **layers**
- Q: bits per **modulation** symbol.
  - From Q=1 (BPSK) to Q=8 (for 256QAM).
- f: **scaling factor**, signaled via higher layers.
  - It should be permissible for a UE to indicate lower performance wrt the max. data rate.

Sources of consumption

Communication (transmission)

Estimate of the nominal data rate

$$\sum_{j=1}^{J}\left(v_L^{(j)} \cdot Q_m^{(j)} \cdot f^{(j)} \cdot R_{max} \cdot \frac{N_{RB}^{(j)}}{T_s} \cdot (1 - OH^{(j)})\right)$$

- R_{max}: Maximum possible **code rate**.
- N_{PRB}: Maximum number of **resource blocks** the bandwidth part or lower.
- T_S: Average **symbol duration**.
- OH: **Overhead** due to signaling information.

Source of energy

Mains power

- Provide virtually unlimited power, making it possible to employ complex protocols, work at high bitrates, and achieve long-distance communication.
- There are some **drawbacks**:
  - Only work in wired fixed systems.
  - Can be dangerous in environments a risk of fire and explosions.
  - Need to lay out an elecric line (challening in hard-to-reach remote/rural areas)
  - Need to put transformers (expensive)

Source of energy

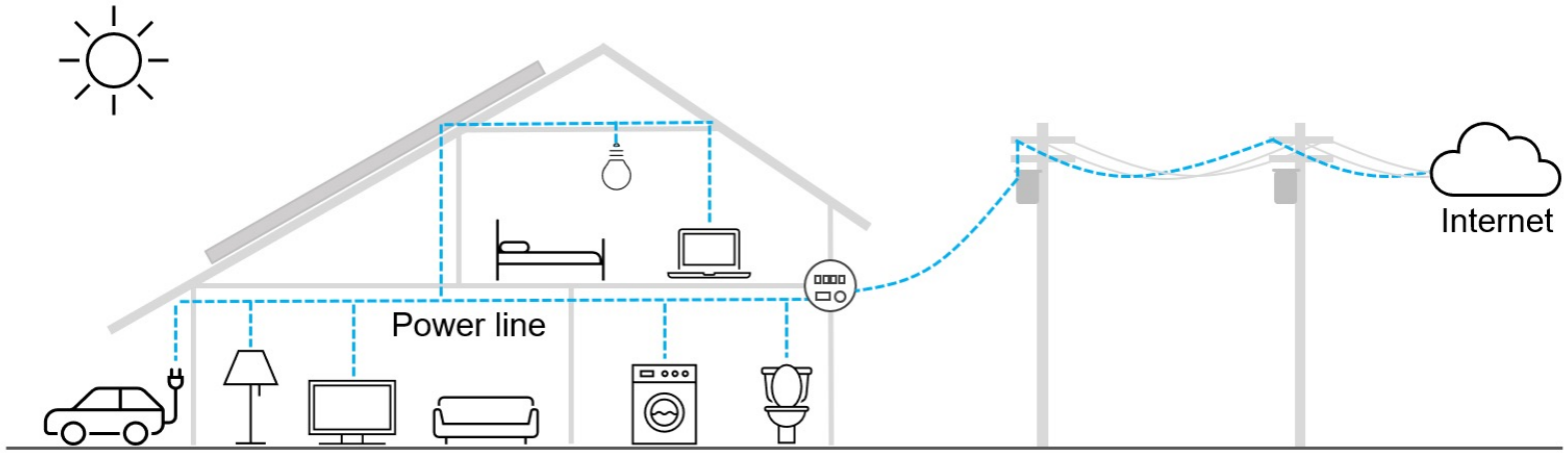
Power over Ethernet (PoE)

- Employ the wired communication infrastucture also for powering.
- Can **carry DC power on the twisted pair** used for Ethernet connectivity.
  - Only one cabled infrastructure needs to be in place.
  - Transformers are not needed.

Source of energy

Powerline communication

- It is the opposite of PoE.
- Use the **powerline to carry data**.
  - Only one cabled infrastructure needs to be in place.
  - Advantageous when the system to be monitored and controlled is the **power grid itself**.



Source of energy

Batteries

- Batteries use the charge difference between ions to induce a potential difference.
  - **Rechargeable** batteries can run the circuit backwards and move the electrons back (by connecting a stronger generator).
  - Charging is never perfect: chemical impurities build up.
- If the battery runs out completely, it is often impossible to recharge without direct intervention (which may be *expensive*): the objective is **never to let it run out**!
  - Apply **statistical data** and direct information from the meter to define a plan of proactive on-field intervention if needed → carry out with proper timing to minimize the number of substitutions in the lifetime and the number of dedicated "truck rolls" or manual readings.

Sources of energy

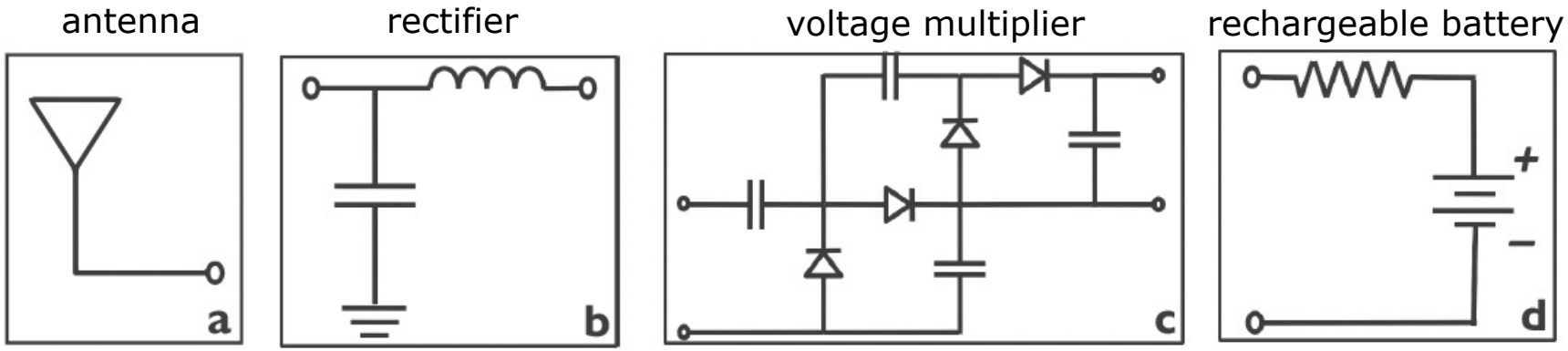
Batteries

- Energy is the basic constraint of IoT systems. What can we do?
  - Employ battery technology with low self-discharging properties.
  - Minimize the duration of active active states of the radio components with suitable scheduling algorithms and MAC protocols.
    - Duty cycles, sleep mode, alert/notification, etc.
  - Adopt energy-aware routing.
  - Introduce endpoint management protocols that communicate the state of batteries.
  - Adopt a “let it be off” appraoch
  - Recharge batteries with **energy harvesting**.

Source of energy

Energy harvesting

- **Energy harvesting**: recharge it while it is running.
- Energy harvesting sensors and actuators are equipped with some kind of generator, and can recharge their battery from it. This allows them to stay alive much longer than with a single battery charge, even if they get very little energy.



RF power harvesters

Source of energy

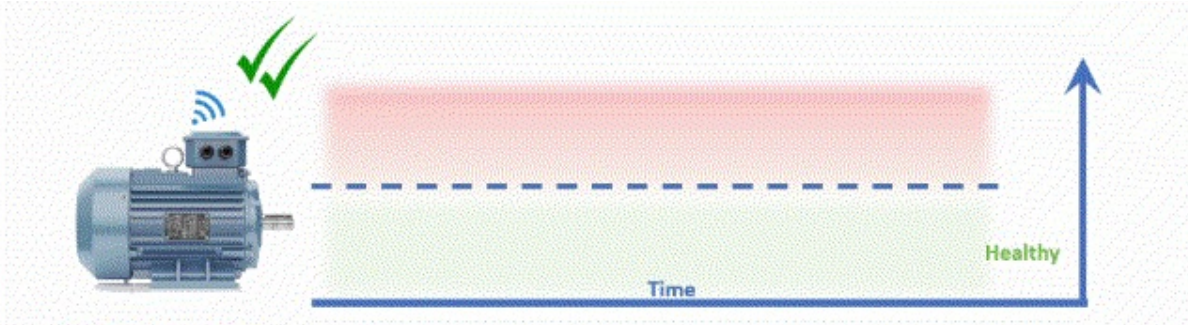
Solar and wind energy

- **Solar** panels are now cheap, and solar energy is renewable and clean.
  - The sensor needs to be exposed (shade can be a problem, underground or indoor sensors will not work).
  - Photovoltaic cells will not generate any power at night (the sensor needs to get through the night on a single charge)
- Another possibility is **wind** energy, but there are several challenges:
  - The sensor still needs to be exposed (underground or indoor sensors will not work)
  - Wind is intermittent and unpredictable.
  - Small turbines are inefficient.

Source of energy

Thermal and vibration energy

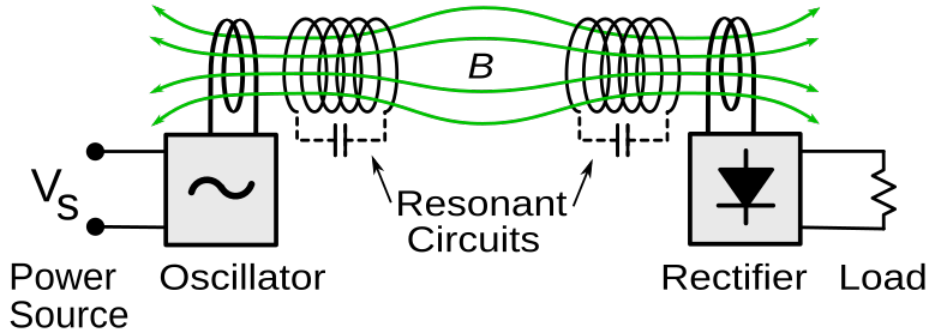
- **Thermal energy** is cheap when the sensor has access to temperature gradients:
  - The inside/outside or ground/air gradient can generate energy.
  - The power is usually very small, but may be the only option for buried sensors.
  - Need for long distances between terminals.
  - Good for weareables: the difference between body temperature and the air is high.
- **Vibration energy** can be collected using piezoelectric devices with good efficiency.
  - The constant presence of vibrations energetic enough to power the endpoint is uncertain, expect in some industrial environments.

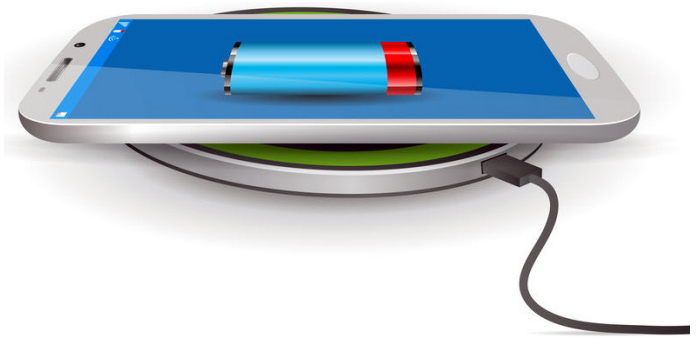


Source of energy

Wireless power transfer (WPT)

- Sensors can benefit from the **power of received transmissions**, but the benefits are tiny at longer distances: this is only feasible for very low-power sensors.
  - Inductive coupling: Uses electromagnetic fields to transfer energy between coils.
  - Capacitive coupling: Transfers energy via electric fields between conductive plates.
  - Uses RF signals to transfer power over longer distances (several meters to kilometers).
  - Uses focused laser beams to transmit power over long distances.





Source of energy

Passive sensors

- We can also use the received transmission to power the circuit directly: **RFID** tags, biometric passports, and contactless cards work this way
  - Power loss is huge: these only work over short distances.
  - The sensor cannot actively transmit or record data, only respond to messages.
  - There are privacy issues (anyone can read a tag, the only limit is the SNR).