**WSN (Lesson2)**

# What are the main features of a WSN? (fault tolerance, scalability, lifetime, security, programmability, affordability, heterogeneity and mobility)

Wireless Sensor Networks (WSNs) possess several key features that distinguish them from other types of networks. Here are the main features of a WSN:

1. **Fault Tolerance**: WSNs are designed to operate in dynamic and often harsh environments, where individual sensor nodes may fail or become unreliable. Fault tolerance mechanisms, such as redundancy and self-healing algorithms, are employed to ensure the network continues to function properly even in the presence of node failures.
   "faults in WSN nodes should not take down the whole WSN, or hamper transmissions"
2. **Scalability**: WSNs can be deployed with a large number of sensor nodes, ranging from a few to thousands or more. The network architecture and protocols are designed to handle the scalability requirements, allowing for efficient communication and management of the sensor nodes.
   "a future increase or decrease of sensor node units, must be supported without changing the whole implementation."
3. **Lifetime**: Sensor nodes in WSNs are often battery-powered and have limited energy resources. Maximizing the network's lifetime becomes crucial. Energy-efficient protocols and techniques, such as duty cycling, data aggregation, and power management, are employed to extend the operational lifespan of the network.
   **"Long life time**: time at which
   • First node depletes its battery
   • Last node depletes its battery
   • x % of nodes are alive
   • x % of the area is covered (e.g., x=75,90,95)
   As it is not feasible to regularly recharge the node batteries it is necessary to provide **energy management:**
   a) duty cycle
   b) sensing range adaptation
   c) selective activation"
4. **Security**: Security mechanisms are implemented to protect data confidentiality, integrity, and authenticity. Techniques such as encryption, authentication, and secure key management are utilized to ensure secure communication and prevent unauthorized access.
   "The security of WSNs, if not considered, can easily compromise with the security of the whole system.
   The management algorithms and protocols of the network extend the attack surface of the system! Think about **mobile sensors**."
5. **Programmability**: WSNs often require flexibility and adaptability to suit various application requirements. Programmability allows for easy customization and reconfiguration of sensor nodes to perform specific tasks or adapt to changing conditions. Programming interfaces and development frameworks are provided to enable application-specific functionality.

"ensures reuse, robustness of these systems, and dynamic, autonomous configuration for adaptation to the environment or scale-up or scale-down operation."

6. **Affordability**: WSNs are often deployed in large-scale deployments where cost is a crucial factor. Sensor nodes are designed to be cost-effective, small in size, and low-power. The affordability factor enables widespread deployment and scalability of WSNs.
   "As the WSNs generally require multiple units, typically in the range of tens or hundreds of WSN nodes, low cost implies acceptability of the system (except for military and some industry)"
7. **Heterogeneity**: WSNs can consist of diverse types of sensor nodes with varying capabilities, communication protocols, and sensing modalities. Heterogeneity allows for specialized nodes tailored to specific sensing tasks and efficient resource utilization in the network.
   support for a wide number and types of sensors.
8. **Mobility:** While WSNs are typically designed for static deployments, there are scenarios where sensor nodes may have mobility capabilities. Mobility introduces additional challenges, such as dynamic topology management and location tracking. Mobile WSNs are used in applications like wildlife tracking, asset monitoring, and mobile healthcare.
   fine tuning of coverage and connectivity, rapid deployability.
   These features collectively enable WSNs to collect, process, and transmit data from the physical world efficiently and reliably. Depending on the specific application requirements, these features are considered and tailored to ensure the successful operation of the WSN.

## How would you define the lifetime of a WSN?

The lifetime of a Wireless Sensor Network (WSN) refers to the duration of time during which the network can operate and fulfill its intended functions effectively. It represents the period from the deployment of the network until the point where it can no longer perform its tasks due to various factors, such as energy depletion, node failures, or network degradation.

The lifetime of a WSN is primarily influenced by the energy constraints of the sensor nodes, as they are typically powered by limited-capacity batteries. The energy consumption of the network is determined by various factors, including the sensing, processing, communication, and data aggregation activities of the nodes.

The lifetime of a Wireless Sensor Network (WSN) refers to its operational duration before the sensor nodes' energy is depleted. To extend the lifetime, energy management techniques are employed:

1. **Duty Cycle**: Nodes are periodically activated and deactivated to reduce energy consumption. Latency may be introduced.
2. **Sensing Range Adaptation**: Adjusting the sensing range based on proximity and application requirements conserves energy.
3. **Selective Activation**: Activating only relevant nodes based on events or triggers reduces overall energy consumption. These techniques strike a balance between network performance and energy

efficiency, ensuring long-term operation and functionality based on application needs and available resources.

### 3. Long life time: time at which

- First node depletes its battery
- Last node depletes its battery
- x % of nodes are alive
- x % of the area is covered (e.g., x=75,90,95)

As it is not feasible to regularly recharge the node batteries
it is necessary to provide **energy management**:
a) duty cycle
b) sensing range adaptation
c) selective activation

**CPS (Lesson3)**

## What are the main features of a Cyber Physical system? ( real-timeliness , intelligence, predictability, interoperability, heterogeneity, scalability, security )

## Cyber Physical Systems

• Cyber-Physical Systems (CPS) are networked monitoring and controlling systems, governed by feedback-based control algorithms.
• Interdisciplinary domain that involves expertise in mechanical, electrical, computing, electronics, and many more.
• The most interesting aspect of CPS is the involvement of the concept of **human-in-the-loop**, which is an integral part of many CPS-based solutions.
• The striking difference of CPS from paradigms such as WSN and M2M is **the inclusion of a compulsory feedback system**

A Cyber-Physical System (CPS) refers to a combination of physical components, such as sensors, actuators, and computational devices, connected through a network infrastructure and integrated with computing systems to enable intelligent monitoring, control, and coordination of physical processes. CPS integrates the physical and twin
l worlds, creating a tightly coupled system where the physical and cyber components interact and influence each other.
Here are the main features of a Cyber-Physical System:

1. **Real-timeliness**: CPS often operate in real-time or near real-time environments, where timely and predictable response to events is critical. They must be able to sense, process, and actuate within strict time constraints to ensure the desired functionality and performance.

The CPS depends on real-time communication, processing, and feedback to effectively provide control to the environment they are deployed.
For example, in a CPS-based industrial chemical concentration monitoring system, the real-timeliness of the process from sensing to feedback to actuation is crucial for maintaining the operations of the chemical manufacturing plant and prevention of disasters.

2. **Intelligence**: CPS incorporate intelligent capabilities, such as data analytics, machine learning, and decision-making algorithms, to process and interpret large amounts of data from the physical world. They use this intelligence to make informed decisions, optimize system behavior, and adapt to dynamic environments.

   Intelligent and **adaptive decision making** is crucial for the maintenance of CPS-based functionalities.
   • In the event of random or sudden changes in the environment being controlled, this feature ensures
   **1. effective control** of the environment and
   **2. effective coordination** between the various dependent sub-processes and systems.
   For example, in case of an electrical fault in a section of a smart-grid system, this feature would enable the re-routing of the electrical supply flow through other paths instead of bringing the whole system to a complete standstill.

3. **Predictability**: CPS aim to provide predictable and reliable behavior. They employ models and analysis techniques to ensure the system's behavior can be accurately predicted and analyzed in different operational scenarios. Predictability is crucial for safety-critical applications where system failures can have severe consequences.

   This feature implies the **prediction of outputs and events** based on past behavior under similar constraints and conditions.
   • The prediction of events enables the activation of **precautionary measures** to control the damage if a harmful event occurs.

   For example, the trend of minor line disturbances and noise in a communication channel might lead to network data loss in a backhaul network. This feature would help in the timely activation of preventive countermeasures to avoid network outage in case the network starts massively dropping packets.

4. **Interoperability**: CPS often interact with a wide range of heterogeneous devices, systems, and networks. Interoperability allows seamless communication and coordination between different components of the CPS, enabling them to work together effectively. Standardized protocols, interfaces, and data formats are essential for achieving interoperability.

   CPS-based systems may include software as well as hardware from **a variety of manufacturers**.
   • This would lead to data, speed and **format mismatch** under normal circumstances, however CPS-based interoperability prevents this and enables **systems from various vendors to work in sync with one another** as a single system.
   • This feature also ensures that legacy systems already-in-place are not replaced but are added to the CPS infrastructure.

5. **Heterogeneity**: CPS involve the integration of diverse technologies, devices, and subsystems. They bring together components with varying capabilities, communication protocols, and interfaces.

Heterogeneity allows for the integration of specialized components and enables the CPS to leverage different technologies for specific tasks.

Heterogeneity in CPS-based systems may be in

- Types of actuators, sensors, processors
- Data formats being used
- Sensing types, software, and application types

However, provisions are already present in CPS to accommodate these types of challenges.

6. **Scalability**: CPS can range from small-scale systems with a few components to large-scale deployments with thousands or more interconnected devices. Scalability ensures that CPS can accommodate the growth in system size, complexity, and the number of interconnected components while maintaining performance and efficiency.

   Scalability in CPS-based systems may be in terms of:

   - network bandwidth,
   - number of sensors
   - number of actuators
   - size of the deployment area
   - other factors

   CPS-systems should be able to handle such demands even after preliminary deployment.
   For example, replace conventional **scalar sensors** with **camera sensors** without changing the whole system.
7. **Security**: CPS operate in interconnected and often open environments, making them vulnerable to cybersecurity threats. Security features, such as authentication, encryption, access control, and intrusion detection, are crucial to protect CPS from unauthorized access, data breaches, and malicious attacks.

   These features collectively define the main characteristics of Cyber-Physical Systems. They enable CPS to interact with the physical world, process data, make informed decisions, adapt to changing conditions, and achieve efficient and reliable operation in various domains, including transportation, healthcare, energy, and manufacturing.

   - The security of CPS is crucial as almost all of the traffic flows through a network and eventually over the Internet.
   - Provisions should be in place to avoid unauthenticated use of the CPS and its hijacking by unscrupulous elements or even attacks.

## What is a digital twin? Why is it relevant and what are the challenges of patrolling solution where mobile nodes are used for updating the state of the twin of local sensors?

**Digital twins** are behavioral and functional mathematical models of actual physical systems.
Virtual simulated models that are mostly used in industrial and machine health monitoring.
A digital twin refers to a virtual representation or digital counterpart of a physical object, system, or process. It is a digital replica that mirrors the real-world entity in

terms of its characteristics, behavior, and interactions. The concept of a digital twin is based on the idea of creating a virtual model that provides insights, analysis, and simulation capabilities to enhance understanding and decision-making.

two advantages of digital twins:

**Huge savings**: during the use of digital twins, in the event of any failure or damage during experimentation or diagnostics, no harm comes to the actual pieces of machinery and processes.
**Extensive experimentation**: tests can be conducted in a variety of simulated scenarios, including safety critical ones with no risk for humans and equipments and no time loss.

1. **Huge Savings**: Digital twins offer significant cost savings by eliminating the need for expensive physical experimentation or diagnostics. Any failures or damages during experimentation on the digital twin do not impact the actual assets, avoiding costly repairs or replacements.
2. **Extensive Experimentation**: Digital twins allow for extensive experimentation in a variety of simulated scenarios, including safety-critical ones. They provide a risk-free platform to test new strategies, optimize performance, and identify potential issues before implementing changes in the physical environment.

Digital twins are behavioral and functional mathematical models of actual physical systems. This model guarantees **huge savings**, beacuse during the use of it, in the event of any failure or damage during experimentation, no harm comes to the actual pieces of machinery and processes, and **Extensive experimentation**, beacuse tests can be conducted in a variety of simulated scenarios, including safety critical ones with no risk for humans and equipments and no time loss.
**With patrolling we want to minimise the inter-visit time and maximise the number of local sensors successfully patrolled by the mobile nodes. Challenges of the Network of mobile nodes (drones, FANET) are Communication (Topology changes frequently, Frequent disconnections when deployed over large areas or with obstacles, High interference especially in urban environments, Low data rate) High mobility and Energy constraints.**

In a patrolling solution using mobile nodes to update the state of local sensors in a digital twin system, several challenges need to be addressed. These include synchronization and data consistency between physical sensors and their digital twins, communication and connectivity issues, accurate localization and positioning of mobile nodes, energy efficiency to prolong operational time, scalability and network management for large deployments, and ensuring security and privacy of transmitted data. Overcoming these challenges requires the implementation of robust synchronization mechanisms, reliable communication links, accurate localization technologies, energy-efficient strategies, efficient algorithms for network management, and strong security measures. Addressing these challenges enables effective patrolling solutions with mobile nodes, enhancing the benefits of digital twin technology in various applications.

Digital twins are relevant due to the benefits they offer in various domains. They provide enhanced understanding, predictive maintenance capabilities, optimization and

performance improvement, and enable remote monitoring and control. However, challenges arise when using mobile nodes for updating the state of digital twins in patrolling solutions. These challenges include connectivity and communication issues, data synchronization, localization and mapping complexities, energy efficiency concerns, and security and privacy considerations. Addressing these challenges requires careful system design, protocol development, and leveraging advancements in technology. Overcoming these challenges enables effective patrolling solutions with mobile nodes, unlocking the full potential of digital twin technology in various applications.

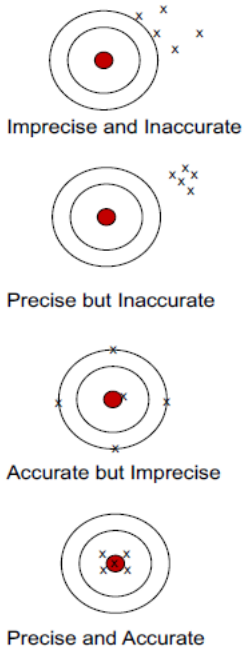
**IOT Devices (Lesson 8)**

## Define the following fundamental properties of a sensor : resolution, accuracy and precision (lez8.pdf)

**Resolution**: The smallest detectable change a sensor can measure.
**Accuracy**: The closeness of a sensor's measurement to the true value.
**Precision**: The consistency and repeatability of measurements obtained from a sensor.

### Sensor Characteristics: Sensor Resolution

The smallest change in the measurable quantity that a sensor can detect is referred to as the **resolution** of a sensor.
For digital sensors, the smallest change in the digital output that the sensor is capable of quantifying is its sensor resolution.
The **accuracy** of a sensor is the ability of that sensor to measure the environment of a system as close to its true measure as possible.
For example, a weight sensor detects the weight of a $100Kg$ mass as $99.98Kg$.
We can say that this sensor is 99.98% accurate, with an error-rate of $\pm 0.02\%$.
A sensor's **accuracy does not depend upon its resolution**.
For example, a temperate sensor **A** can detect up to $0.5^oC$ changes in temperature, whereas another sensor **B** can detect up to $0.25^oC$ changes in temperature. Therefore, the resolution of sensor **B** is higher than the resolution of sensor **A**.
The accuracy is irrelevant in defining the resolution
The **sensor's precision** is related to the repeatability of the sensor' measurements.
If, upon multiple repetitions, the sensor is found to have the same error-rate, then only it can be deemed as highly precise.



| | |
|---|---|
| For example, in the above experiment, the weight sensor reports measurements of 98.28*Kg*, 100.34*Kg*, and 101.11*Kg* upon three repeat measurements for a mass of actual weight of 100*Kg*. Here, the sensor precision is not high because of significant variations in temporal measurements for the same object under the same conditions. | |

## Define the K-coverage objective in a sensor network.

Typical approaches to optimize the sensing range in deployments include **static k-coverage** and **dynamic k-coverage**.

• A lifelong fixed k-coverage requires redundancy, the sensing range of at least k sensors overlap. In contrast, dynamic k-coverage incorporates mobile sensor nodes post detection of an event, which, however, is a costly solution and may not be deployable in all operational areas and terrains.

**K-coverage** in sensor networks ensures a certain number of sensors (K) monitor an area. There are two types:

1. **Static K-Coverage**: ensures a constant number of sensors (K) continuously covering a fixed target area. providing constant and reliable monitoring.
2. **Dynamic K-Coverage**: K-coverage adapts to changes like node failures or mobility, maintaining the desired level of coverage.

The choice depends on application requirements. Static is for fixed areas, while dynamic handles changing conditions. Achieving K-coverage involves sensor deployment, position optimization, and algorithms. K-coverage enables **reliable monitoring for surveillance, environmental monitoring, and event detection.**

## Name and describe different types of actuators.

actuators can be divided into seven classes

- **Hydraulic**
- **Pneumatic**
- **Electrical**
- **Thermal/Magnetic**
- **Mechanical**
- **Soft Shape Memory Polymers.**

**Hydraulic** actuators are devices that convert hydraulic pressure or fluid power into mechanical motion or force. They are commonly used in various industrial applications where high force, precise control, and reliable operation are required. Hydraulic actuators utilize the properties of incompressible fluids to generate motion or force.

**Pneumatic** actuators are devices that convert compressed air or gas into mechanical motion or force. They are widely used in various industries for applications that require quick and reliable actuation. Pneumatic actuators utilize the power of compressed air or gas to generate linear or rotary motion.

**Electrical** actuators are devices that convert electrical energy into mechanical motion or force. They are widely used in various applications where precise control, automation, and electrical power sources are available. Electrical actuators offer advantages such as high precision, controllability, and compatibility with electrical systems.

## Electric Actuators

Typically, electric motors are used to power an electric actuator by generating mechanical torque.

• This generated torque is translated into the motion of a motor's shaft or for switching (as in relays).

• For example, actuating equipment such as solenoid valves, control the flow of water in pipes in response to electrical signals.

• This class of actuators is considered one of the cheapest, cleanest and speedy actuator types available.

**Thermal and magnetic** actuators are types of actuators that utilize thermal or magnetic principles to generate mechanical motion or force. These actuators are commonly used in various applications, including microelectromechanical systems (MEMS), robotics, automotive systems, and industrial automation.

• **Thermal or magnetic actuators** are typically compact, lightweight, and economical.

• One classic example of thermal actuators is shape-memory materials (SMMs) such as shape memory alloys (SMAs).

• These actuators do not require electricity for actuation.

• These are not affected by vibration and can work with liquid or gases. Magnetic shape memory alloys (MSMAs) are a type of magnetic actuator.

**Mechanical** actuators are devices that convert mechanical energy into linear or rotary motion to perform a specific task or control a system. They are widely used in various applications ranging from industrial machinery to robotics and automotive systems. Mechanical actuators are typically driven by mechanical force, such as manual operation, electric motors, or hydraulic/pneumatic systems.

• In **mechanical actuation**, the rotary motion of the actuator

is converted into linear motion to execute some movement.

• The use of gears, rails, pulleys, chains, and other devices are

necessary for these actuators to operate.

• These actuators can be easily used in conjunction with pneumatic,

hydraulic, or electrical actuators.

• They can work in a standalone mode also. The best example of a

mechanical actuator is a rack and pinion mechanism.

• For example, a hydroelectric generator converts the water-flow induced

rotary motion of a turbine into electrical energy.

**Soft shape memory polymers (SMPs)** actuators are a type of smart material actuator that can change their shape in response to external stimuli, such as heat, light, or moisture. SMP actuators are made from shape memory polymers, which are polymer materials that can undergo a reversible change in shape when exposed to specific triggering conditions.

• **Soft actuators** (e.g., **polymer-based**) consists of elastomeric polymers, which are used as embedded fixtures in flexible materials such as cloth, paper, fiber, particles, and others.

• The conversion of molecular level microscopic changes into tangible macroscopic deformations is the primary working principle of this class of actuators.

• These actuators have a high stake in modern-day robotics. They are designed to handle fragile objects such as agricultural fruit harvesting, or performing precise operations like manipulating the

internal organs during robot-assisted surgeries.

**Shape memory polymers** (SMP) are considered as smart materials, which respond to some external stimulus by changing their shape, and then revert to their original shape once the affecting stimulus is removed.

Features such as high strain recovery, biocompatibility, low density, and biodegradability characterize these materials.

SMP-based actuators function similar to our muscles.

Modern-day SMPs have been designed, which respond to a wide range of stimuli such as pH changes, heat differentials, light intensity, and frequency changes, magnetic changes, and others.

**Light Activated Polymers**

• Photopolymer/light-activated polymers (LAP) are a particular type

of SMP, which require light as a stimulus to operate.

• LAP-based actuators are characterized by their rapid response

times. Using only the variation of light frequency or its intensity,

LAPs can be controlled remotely without any physical contact.

• The polymer retains its shape after removal of the activating light.

In order to change the polymer back to its original shape, a light stimulus of a different frequency has to be applied to them.

**Wireless PHY (Lesson 9.1)**

## What is the multipath fading in wireless transmission, and why does it affect the communication capabilities of wireless protocols?

Signal can take many different paths between sender and receiver due to reflection, scattering, diffraction
Time dispersion: signal is dispersed over time
è interference with "neighbor" symbols, Inter Symbol Interference (ISI)
§ The signal reaches a receiver directly and phase shifted
è distorted signal depending on the phases of the different parts
Signal replicas received via different propagation paths are combined at the receiver
• The results depend on
– The number of replicas
– Their phases
– Their amplitudes
– Frequency
Resulting signal can be **attenuated or amplified**

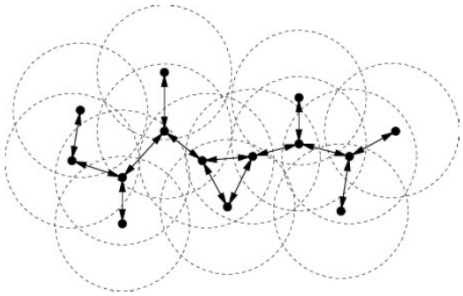
**CHAT GPT :**
Multipath fading is a wireless communication phenomenon caused by signal reflections, diffractions, and scattering. It affects wireless protocols in the following ways:

1. Signal Attenuation: Fading weakens or loses certain signal frequencies, leading to decreased signal strength and potential connectivity issues.
2. Signal Distortion: Fading causes time dispersion, resulting in overlapping symbols and decoding errors.
3. Bit Error Rate (BER): Fading increases the probability of errors in received data.
4. Channel Capacity: Fading reduces the maximum data rate that can be reliably transmitted.

However, fading can also have positive effects, such as diversity gain and spatial diversity, improving signal quality and reliability. To mitigate fading effects, techniques like equalization, diversity reception, and error correction coding are employed to enhance wireless communication performance.

## What is a unit disk graph, and what type of approximation does it imply when used in modeling wireless topologies?

Fading-why is it important?



Unit disk graph

Pay attention: making the assumption that the network topology is a unit-disc graph is a strong approximation to be aware of.
Solutions relying heavily on this approximation sometimes fail completely in real life

**It's a simplified graph model, where nodes represent wireless devices or sensors. This approximation implies that every device has the same signal propagation ranges (range of 1 unit)**

A unit disk graph is a simplified graph model used for wireless topologies. Nodes represent wireless devices or sensors, and edges exist between nodes within a unit distance of each other. This model assumes nodes have a transmission range of 1 unit. each point (or node) is associated with a disk of a fixed radius, typically one unit. In a unit disk graph, two nodes are connected by an edge if their associated disks intersect or touch each other
It simplifies connectivity analysis and algorithms but doesn't capture real-world complexities like signal fading and interference. While useful for initial analysis, it may not fully represent practical wireless networks.

## Bluetooth (Lesson 9.1)

## What kind of topologies are used by Bluetooth devices?

### Piconet and Scatternet

In Bluetooth, a piconet is a network formed by one master device and up to 7 active slave devices. The master device initiates and controls the communication within the piconet. The slaves synchronize their communication with the master and only communicate with the master, not

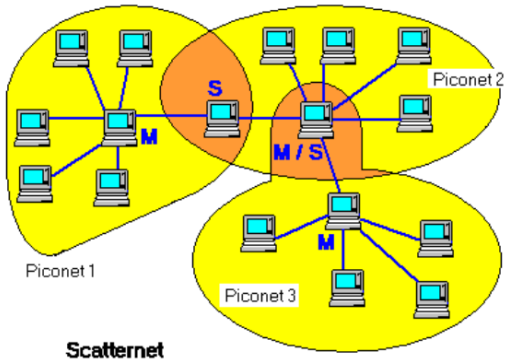
directly with each other. The piconet operates on a single hopping channel within the 2.4 GHz ISM band. All devices have the same timing and frequency hopping sequence.

A scatternet, on the other hand, is a network formed by multiple piconets. In a scatternet, one or more devices can act as both a master and a slave, allowing them to participate in multiple piconets simultaneously: A device can be a master of one piconet and slave of another piconet at the same time. The devices in a scatternet may share common hopping channels or operate on different channels. Devices acting as bridges or relays facilitate communication between different piconets within the scatternet. There is no time or frequency synchronization between piconets.

The concept of piconets and scatternets in Bluetooth enables the formation of flexible and scalable networks, allowing devices to participate in multiple communication groups and extend the range and capabilities of Bluetooth-enabled systems.

# Establishing Piconets

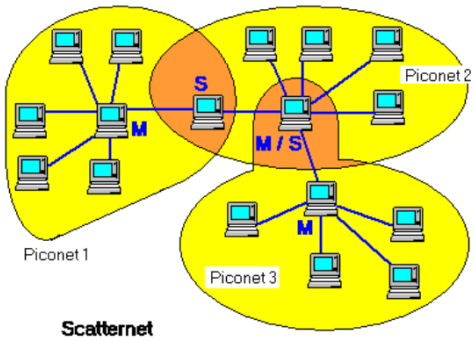
- Whenever there is a connection between two Bluetooth devices, a piconet is formed
- Always 1 master and up to 7 active slaves
- Any Bluetooth device can be either a master or a slave
- Can be a master of one piconet and a slave of another piconet at the same time (scatternet): m/s, s/s, but not m/m!
- All devices have the same timing and frequency hopping sequence



Scatternet

# Scatternets

- Formed by two or more Piconets
- Master of one piconet can participate as a slave in another connected piconet
- No time or frequency synchronization between piconets




## Describe the channel access technology employed by Bluetooth.

?

**frequency hopping spread spectrum (FHSS):** The Bluetooth channel is represented by a pseudo-random hopping sequence through the entire 79 frequencies, each of 1 MHz. The devices hop from a channel to another with a nominal hopping sequence of 1600 per second. The channel is divided in time slots of 1/1600 s (time-division duplex) in which subsequent slot are alternatively used for transmitting and receiving; **even** slots are used by the **master**, **odd** ones by the **slaves**.

Bluetooth uses frequency hopping spread spectrum (FHSS) as its channel access technology. FHSS allows Bluetooth devices to efficiently share the available frequency spectrum and overcome interference from other devices or wireless technologies in the same frequency band. Here's how it works:

1. Frequency Hopping: Bluetooth devices hop between 79 different frequencies within the 2.4 GHz ISM band.
2. Synchronization: All devices synchronize their hopping sequences, starting at the same time and using the same pattern.
3. Hopping Pattern: A pseudo-random algorithm defines the hopping sequence, ensuring devices follow the same pattern.
4. Time Division Duplex (TDD): Bluetooth divides communication into time slots for transmission and reception, typically lasting 625 microseconds. 1600 hops per second through 79 1MHz channels **even** slots are used by the **master**, **odd** ones by the **slaves**.
5. Packet Transmission: Data is transmitted in packets during assigned time slots, consisting of headers, payload data, and error correction codes.
6. Adaptive Frequency Hopping: Bluetooth adaptively switches frequencies in the hopping pattern to minimize interference or congestion.

## Describe the FHSS access technology used by Bluetooth.

FHSS: The Bluetooth channel is represented by a pseudo-random hopping sequence through the entire 79 frequencies, each of 1 MHz. The devices hop from a channel to another with a nominal hopping sequence of 1600 per second. The channel is divided in time slots of 1/1600 s (time-division duplex) in which subsequent slot are alternativly used for transmitting and reciving; even slots are used by the master, odd ones by the slaves.

Bluetooth utilizes Frequency Hopping Spread Spectrum (FHSS) as its access technology. FHSS is a modulation technique that allows Bluetooth devices to share the available frequency spectrum efficiently and mitigate interference from other devices or wireless technologies operating in the same frequency band.

Bluetooth uses Frequency Hopping Spread Spectrum (FHSS) as its access technology. Here's how it works:

1. Frequency Hopping: Bluetooth devices rapidly switch between different frequencies within the 2.4 GHz ISM band.
2. Time Division Duplex (TDD): Bluetooth divides communication into time slots for transmission and reception, typically lasting 625 microseconds.
3. Communication is divided into time slots, each assigned to a specific frequency channel.
   1600 slots in a second through 79 1MHz channels
   **even** slots are used by the **master**, **odd** ones by the **slaves**.
4. Hopping Pattern: Devices use a **pseudo-random hopping pattern** generated by a defined algorithm.
5. Synchronization: Devices synchronize their hopping sequences based on timing signals from a master device.
6. Adaptive Frequency Hopping: Devices adaptively switch to avoid congested or interfered frequencies.
7. Interference Avoidance: Rapid hopping minimizes interference from other devices or technologies.

## How do the Bluetooth nodes of a piconet agree on a common sequence?

During the formation of the piconet, during the paging process the master node utilize the slaves' id to generate a sequence sending FHS (Frequency Hopping Synchronization) packet to the slaves. At the end of paging, a new synchronized connection is established

## Page scan

- The page scan state is entered by the slave from either the standby or the connection state.
- The receiver listens to packets addressed to it (receiver ID known by the master after the inquiry)

## Page response

- Upon reception of the page message the slave enters the page response state
- It sends a response containing its ID **at the frequency for the next slot after the page was received**

## Step 1: The Page Command

- Device broadcasts a page message out to the device that it wants to set up a connection with
  - Does this in a similar manner as inquire messages (on 2 frequency trains of 16 frequencies each)
- Once the device receives a page response, it will stop paging and move on to step 2

## Paging: Steps 2 & 3

- Step 2: In the page response, an acknowledgement is sent back to the master containing the slave ID
- Step 3: In the master response, the frequency hopping generator is stopped (the sequence is chosen) and the master issues an FHS packet to the slave

## Paging: Step 4

- The slave issues a final slave response transmission that is aligned to the slave's native clock
- Using the data from the FHS packet, the slave calculates the master's frequency hopping pattern and synchronizes to its clock

## Paging: Step 5

- When the master receives the packet, it jumps back to its frequency hopping pattern and assigns the slave an Active Member Address (AMA) for the piconet
- Master sends out a poll packet to ensure that the slave is on its frequency hopping pattern

## Paging: Step 6

- Once the slave receives the poll packet, the slave replies with any kind of packet to ensure that it is on the right channel
- The acknowledgement must be received by the Master within the timeout period
- At the conclusion of step 6, a new synchronized connection is established between the master and the slave

**Describe the piconet formation protocol in Bluetooth.**

## Inquiry Scan

- A device that wants to be discovered periodically enters this mode and listens for inquiry packets at a single frequency – chosen out of 16 frequencies
  - Inquiry hop sequence depends on device address
- Stays in the state long enough for an inquiring device to cover 16 frequencies
- Will **re-enter inquiry scan** state even after responding to an inquire

## Inquiry Response

- When radio receives inquire, it will wait between 0 and .32 seconds before sending an FHS packet as a response
  - This is done to avoid collision with another radio that also wants to send an FHS packet
- FHS Packet contains:
  - Device ID
  - Clock
- After inquiring radio is done with inquiring procedure, the inquirer knows all of the radios (that are discoverable) within range

**Inquiring**: A device that wants to be discovered by Bluetooth devices periodically starts a scan (inquiry scan) to discover available devices in the area. This request is broadcasted over a specific period and frequency range out of on 16 different frequencies. (inquiry scan mode). The receiver performs long and frequent inquiring scans to intercept the inquiring message; when it's received, the receiver sends an inquiry response packet (FHS) after a random number of slots (waits between 0 and 0.32 secs). The packet contains the receiver id and clock.

**Paging**: The master sends a page message (using the slave to be paged clock it received after inquiring). The slave in page scan mode receive the message and sends an ack response containing its ID, then the master generates a sequence and sends a FHS packet to the slave. Salve uses the information inside of it to synchronize its clock with the master frequency hopping pattern. Then the master assigns an Active Member

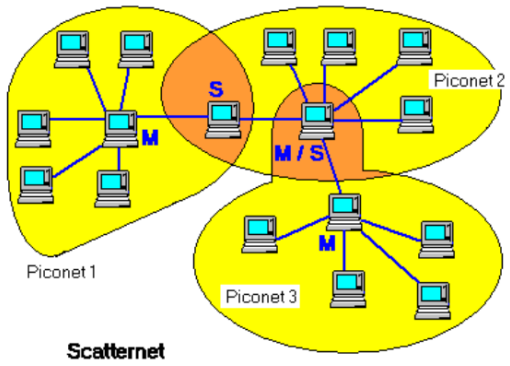
Address to the slave for the piconet and sends a poll packet to ensure that the slave is on its frequency hopping pattern and after the ack response from the slave the new synchronized connection is established.

The inquiry procedure is used by a Bluetooth device, known as the inquiring device, to discover other nearby devices within its range. The inquiring device sends out inquiry messages to search for devices that are in discoverable mode. When a discoverable device receives an inquiry message, it responds with an inquiry response containing its device information. By performing an inquiry, the inquiring device can create a list of available devices that can potentially be connected to.

Once the inquiring device has identified a specific device it wants to connect to, it initiates the paging procedure. Paging is the process of establishing a connection between two Bluetooth devices. The inquiring device sends a paging message to the target device, specifying its device address. The target device receives the paging message and responds with a paging response. If the paging is successful, a connection is established between the two devices, forming a piconet.

## Establishing Piconets

- Whenever there is a connection between two Bluetooth devices, a piconet is formed
- Always 1 master and up to 7 active slaves
- Any Bluetooth device can be either a master or a slave
- Can be a master of one piconet and a slave of another piconet at the same time (scatternet): m/s, s/s, but not m/m!
- All devices have the same timing and frequency hopping sequence



Scatternet

**Consider a scatternet with two master nodes and an s/s gateway. How do the two masters agree on a common frequency hop scheme, if any.**

There is no frequency synchronization between different piconets from a scatternet.
Synchronization should be within single piconet . The slave is active for one master (at a specific time)

An s/s gateway can't be in active mode in two different piconets at the same time.
(in order for a slave to be active in another piconet of the same scatternet it got to enter to the hold mode in its current piconet)

## Which modes are available for Bluetooth nodes participating in a piconet? (active, hold, sniff and park)

Bluetooth nodes participating in a piconet have access to different modes that dictate their behavior and power consumption. The modes available for Bluetooth nodes in a piconet are as follows:

1. **Active Mode**: In the active mode, the node actively participates in the piconet by engaging in regular data transmission and reception. It stays synchronized with the hopping sequence, performs its assigned tasks, and maintains full communication capabilities. The active mode is the default mode when a node is actively involved in data exchange within the piconet.

### Active Mode

- Limited to 7 Active slaves for each master
- Three bit address (AM_ADDR) given to each active slave
- Unit actively participates on channel
- Can receive communications in any given frame
- Active slaves are polled by master for transmissions
- Unit operates on high-power

2. **Hold Mode**: The hold mode allows a node to temporarily suspend its participation in the piconet while maintaining its connection with the master device. In hold mode, the node halts data transmission and reception for a specified period, conserving power during idle times. The hold mode is typically used when the node expects to be temporarily inactive but intends to resume its communication later.

# Hold Mode

- Frees slave to
- Attend another Piconet
- Perform scanning, paging, or inquiry operations
- Move into low-power sleep
- Unit keeps active member address
- Unit does not support ACL packets on the channel but may support SCO packets
- Master and slave agree on a one time hold $T_{hold}$ duration after which the slave revives and synchronizes with channel traffic
- Unit operates on low-power

3. **Sniff Mode**: The sniff mode is a power-saving mode designed to reduce a node's power consumption while still maintaining a certain level of communication capability. In sniff mode, the node synchronizes with the piconet's hopping sequence but reduces the frequency of active

participation. It periodically wakes up to check for any data transmission or reception activity, allowing for intermittent communication while extending battery life.

# Sniff Mode

- This is a low power mode in which the listening activity of the slave is reduced
- Very similar to hold mode
- Slave is freed for reoccurring fixed time intervals $T_{sniff}$ at give offsets $D_{sniff}$ for a given number of times $N_{sniff}$
- Master can only communicate during arranged "sniff" time slots
- The parameters $T_{sniff}$, $D_{sniff}$ and $N_{sniff}$ are sent by the master with a SNIFF command

4. **Park Mode**: The park mode is the most power-efficient mode in a piconet. When a node enters the park mode, it temporarily releases its connection with the master device and relinquishes its active participation in the piconet. The node enters a low-power state, effectively becoming inactive. While in park mode, the node periodically listens to the piconet's beacon packets to maintain synchronization but remains dormant most of the time. The park mode is useful for devices that are not actively involved in data exchange but need to remain within the piconet for occasional communication.

# Park Mode (up to 255 nodes in a piconet)

- Parked unit gives up active member address and is assigned
- 8 bit Parked member address (PM_ADDR) – allows master to unpark slave
- Unit stays synchronized to channel
- Operates in very low-power sleep

- Provides the ability to connect more than 7 devices to a master (8 bit PM_ADDR allows 255 parked devices)
- Active and Parked slaves can be switched in and out to allow many connections to a single piconet

- Master establishes a beacon channel and beacon interval when a slave is parked

- Parked slave wakes up at regular beacon interval to
- Maintain synchronization
- Listen for "broadcast" messages
  - Potentially make access (unpark) request to master

Bluetooth nodes can dynamically switch between these modes based on their operational requirements and power-saving considerations. The ability to switch between modes enables efficient power management within a piconet, allowing nodes to optimize their power consumption while maintaining the necessary connectivity for communication.

**LPWAN (Lesson 10)**

## Which protocols are commonly used to ensure the communication in an LPWAN , and which challenges should they address ?

LPWAN (Low-Power Wide Area Network) technologies typically employ several protocols to ensure reliable communication over long distances while optimizing power consumption. The specific protocols used may vary depending on the LPWAN technology being utilized. Here are some common protocols used in LPWANs:

1. **LoRaWAN**: LoRaWAN is a popular LPWAN technology that uses the LoRa modulation scheme. It employs the following protocols:
   a. **LoRa** (Physical Layer): LoRa is the physical layer protocol that utilizes chirp spread spectrum modulation to enable long-range communication with low power consumption.
   Enables very-long-range transmissions (more than 10 km in rural areas) with **low power consumption**
   b. LoRaWAN (MAC Layer): LoRaWAN is the Medium Access Control (MAC) layer protocol that provides network-wide control, authentication, and encryption. It manages the communication between end-devices and the LoRaWAN network server.
   c. MQTT (Message Queuing Telemetry Transport): MQTT is an application-layer protocol commonly used with LoRaWAN for lightweight and efficient messaging between devices and applications.
2. **Sigfox**: Sigfox is another LPWAN technology that utilizes Ultra Narrow Band (UNB) modulation. It employs the following protocols:
   a. Sigfox Protocol: Sigfox has its own proprietary protocol for communication between devices and the Sigfox network. It includes features like device registration, authentication, and data transmission.

b. RESTful API: Sigfox provides a RESTful API that allows devices and applications to interact with the Sigfox network, enabling device management, data retrieval, and integration with external systems.

3. **NB-IoT** (Narrowband Internet of Things): NB-IoT is an LPWAN technology that leverages existing cellular infrastructure. It uses the following protocols:
   a. LTE (Long-Term Evolution): NB-IoT operates within the LTE framework and utilizes LTE protocols for communication, including physical layer modulation, access control, and data transfer.
   b. IP-based Protocols: NB-IoT devices can use IP-based protocols for communication, such as TCP/IP and UDP/IP, enabling seamless integration with existing IP networks and services.

**Challenges** of LPWAN (Low-Power Wide Area Network) technologies arise due to their specific requirements and operating conditions. Here are some challenges associated with LPWAN:

**LPWAN** (Low-Power Wide Area Network) technologies face specific **challenges**:

1. Ultra-Low Power and Weak Reception: LPWAN devices operate on minimal power, resulting in weak reception levels, especially in challenging environments.
2. Very Low Bit Rates: LPWANs have low data rates, below 1 kbps, to ensure long-range coverage and power efficiency.
3. Interference from Other Services: LPWANs share frequency bands with Wi-Fi, Bluetooth, and other devices, leading to potential interference.
4. Increased Interference at Base Stations: Base stations are prone to interference due to their exposed antennas.
5. Hidden Node Problem and CSMA Limitations: Traditional CSMA protocols may not work well in LPWANs due to the hidden node problem* and weak signals over large distances.
6. Spread Spectrum and Frequency Hopping: LPWANs employ spread spectrum and frequency hopping techniques to mitigate interference and improve robustness. Addressing these challenges requires careful design, signal processing, and resource utilization in LPWAN environments.

*The hidden node problem occurs when two nodes in a wireless network can communicate with a third node, but cannot communicate with each other directly due to obstacles or being out of range. This can lead to collisions at the third node when both nodes transmit simultaneously.

## Which is the typical topology in LoRaWAN ?

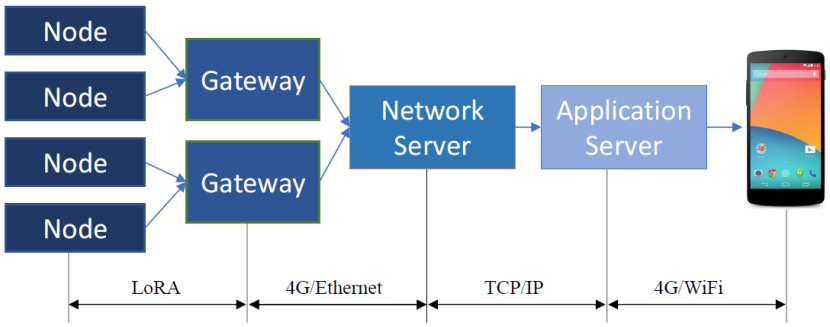
The typical topology in a LoRaWAN (Long Range Wide Area Network) deployment is a star-of-stars topology. In this topology, there is a central entity called the Network Server (NS) that acts as a coordinator for the network. The NS is responsible for managing the network, handling device registrations, managing security keys, and routing data between end-devices and application servers.

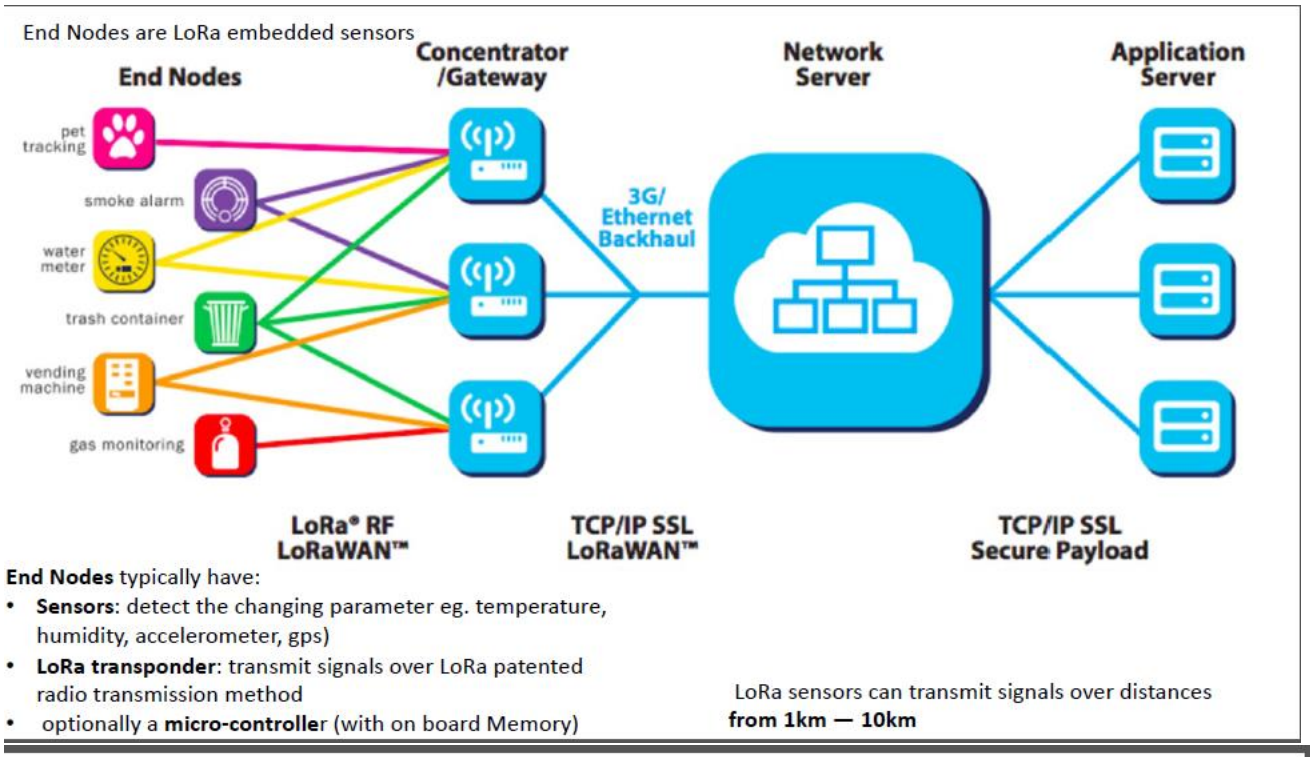
LoRaWAN deployments typically use a star-of-stars topology:

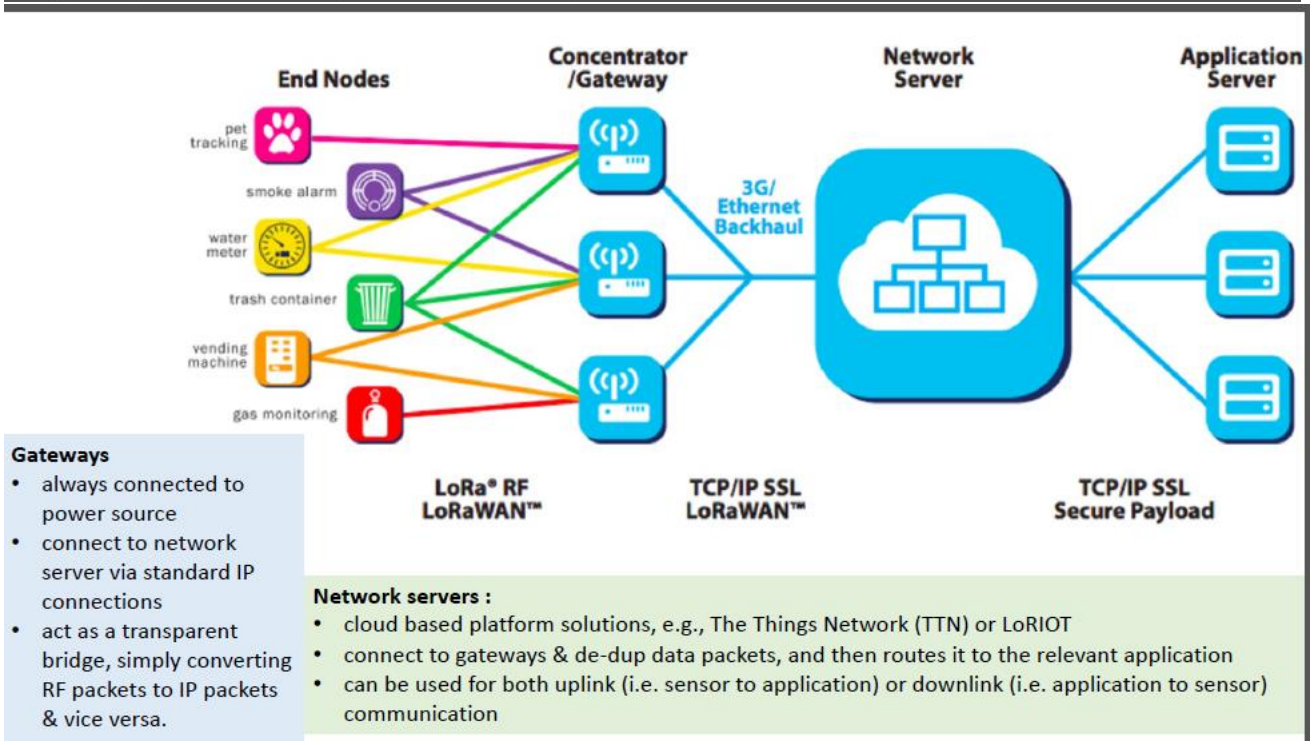
1. End-Devices: Battery-powered devices with LoRa transceivers that communicate with the network.
2. Gateways: Bridge between end-devices and the LoRaWAN network, forwarding signals to the Network Server.
3. Network Server: Central control and coordination point managing device registrations, security, and data routing.
4. Application Server: Processes data received from end-devices, performs analysis, and interacts with the Network Server. End-devices communicate with the nearest gateway, which transmits data to the Network Server. The Network Server manages the network and routes data to the appropriate application server(s). This topology allows for scalability, flexibility, and reliable communication in LoRaWAN deployments.

## LoRaWAN Network Architecture

- Deployed in a star-of-stars topology
- Have base stations relaying the data between sensor nodes & network server
- Communication between sensor nodes & BS goes over the wireless channel utilizing LoRa physical layer, whilst the connection between gateways & central server are over a backbone IP-based network
- **End Nodes** transmit directly to all gateways within range, using LoRa
- **Gateways** relay messages between end-devices & central network server using IP






## Which type of modulation technique is used by LoRa?

LoRa (Long Range) technology utilizes a modulation scheme known as chirp spread spectrum (**CSS**) modulation. **Chirp spread spectrum modulation is a form of linear frequency modulation where the frequency of the transmitted signal varies linearly with time.**
In LoRa modulation, the data signal is modulated by encoding it onto a chirp waveform. A chirp waveform is characterized by a continuous and smooth frequency sweep over time. The

transmitted signal's frequency starts at a low value and increases or decreases linearly over the symbol duration.
Key characteristics of LoRa modulation include:

LoRa (Long Range) technology uses chirp spread spectrum (CSS) modulation, a form of linear frequency modulation. It encodes data onto a chirp waveform with a continuous frequency sweep. Key characteristics include:

1. Chirp Waveform: Data is encoded on a smooth, continuously changing frequency signal.
2. Wide Signal Bandwidth: Signal spreading over a wide bandwidth improves reception and interference resilience.
3. Spreading Factor: Flexibility in spreading factors (SF) allows trade-offs between range and data rate. SF ranges from 7 to 12.
4. Low Data Rate: Optimized for low data rate applications, achieved through longer symbol durations and wider bandwidths. LoRa modulation is ideal for long-range, low-power IoT applications, providing reliable communication and power efficiency. It forms the basis of LoRaWAN technology for IoT deployments in various industries.

**Describe the CSS (Chirp Spread Spectrum) technique. Which IoT protocol utilizes this technique and why?**

CSS is the modulation technique utilized by LoRa. The frequency of the signal continuously varies over time and it's used by Lora to reach very long distances

## Describe the CSS (Chirp Spread Spectrum) technique. Which IOT protocol utilizes this technique and why ?

Chirp Spread Spectrum (CSS) is a modulation technique that is used in wireless communication systems, including some IoT (Internet of Things) protocols such as LoRaWAN (Long Range Wide Area Network).
CSS works by modulating the data signal onto a chirp waveform, which is a continuous and linearly varying frequency signal over time. In a chirp waveform, the frequency of the signal either increases or decreases linearly over the symbol duration.
Here are the key aspects of the CSS technique:

Chirp Spread Spectrum (CSS) is a modulation technique used in wireless communication systems, including IoT protocols like LoRaWAN. It involves modulating data onto a chirp waveform with a linearly varying frequency over time. Key aspects of CSS are:

1. Chirp Waveform: It uses a continuous and linearly changing frequency signal.
2. Spread Spectrum: CSS spreads the signal's energy over a wider bandwidth, improving resistance to interference.
3. Robustness: CSS is resilient to interference and noise, ensuring reliable communication in noisy environments.

4. Long Range: CSS enables long-range communication by surpassing obstacles and maintaining signal integrity.
5. Low Power Consumption: CSS is suitable for low-power applications, conserving energy in battery-operated devices. LoRaWAN, a popular IoT protocol, utilizes CSS for its long-range coverage, interference robustness, and power efficiency. It enables efficient and reliable communication in various IoT applications.

## What is the spreading factor in LoRa and why do we use different SFs ?

The spreading factor is a parameter that determines the spreading of the signal in CSS. Different SFs coincede with different distances, so we use more then one (LoRa uses 6 different SFs, form SF7 to SF12)

> In LoRa modulation, the spreading factor (SF) is a configurable parameter that determines the spreading of the transmitted signal and impacts the communication range, data rate, and robustness to noise and interference. The spreading factor represents the ratio of the spreading bandwidth to the data rate.

In LoRa modulation, the spreading factor (SF) determines signal spreading and affects range, data rate, and robustness. Here's why different spreading factors are used:

1. Communication Range: Higher SF allows longer-range communication with slower data rates but better sensitivity to weak signals, ideal for wide area coverage.
2. Data Rate: Higher SF lowers data rate, as more chirps represent each symbol, enabling trade-offs between data rate and range to adapt to different requirements.
3. Robustness: Varying SF provides resistance to interference and noise, with higher SF offering better reliability in challenging environments.
4. Channel Capacity: Different SFs optimize channel utilization, allowing multiple devices to operate simultaneously without significant interference. In summary, spreading factors in LoRa optimize range, data rate, robustness, and channel capacity for diverse IoT applications.

## Sensor Networks and IOT

> Define a Voronoi diagram and say how it can be applied to define responsibility regions of things or gateways in the IOT?

**Given a set of points P = {p1…pn} $\epsilon$ $R^2$.**

**Voronoi cell: V(pi) = {q $\epsilon$ $R^2$ | d(pi,q) < d(pj,q) for all pj != pi}**

**Voronoi diagram is the planar subdivision obtained by removing all voronoi cells from $R^2$**

**A voronoi diagram can define a network in the IoT, where the points are the devices of the network.**

**Applications:**

**Nearest neighbor (node in a network) queries: For a given query point, locate the nearest point site in n(log n) time -> point location**

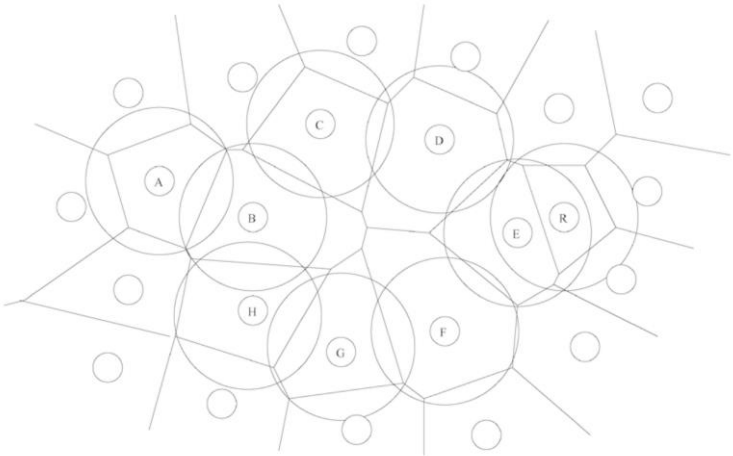
**Closest pair computation: a site and its closest sites share a voronoi edge -> check all voronoi edges in O(n)**

**Facility location: in order to minimize interference, we want to put any new device on a voronoi edge (absolutly not sure about this last one point, didn't really undestand the slide)**

A Voronoi diagram is a geometric partitioning technique that divides a given space into regions based on the proximity to a set of points called "sites" or "seeds." Each region represents the area closest to its corresponding seed point than any other seed point in the space.

In the context of the Internet of Things (IoT), Voronoi diagrams can be used to define responsibility regions of things or gateways. Here's how it can be applied:

By using Voronoi diagrams to define responsibility regions in the IoT, the distribution of tasks and responsibilities can be optimized based on proximity and locality. It enables efficient resource allocation, reduces communication overhead, and facilitates decentralized decision-making in IoT networks. Additionally, as the distribution of things or gateways changes dynamically, the Voronoi diagram can be updated accordingly to adapt to the changing responsibilities and assignments within the IoT system.




## Connectivity Technologies (Lesson 11)

## Which type of modulation technique is used by IEEE 802.15.4 ?

- IEEE 802.15.4 uses the Direct Sequence Spread Spectrum (DSSS) modulation technique enabling a **wider bandwidth** of operation with

**enhanced security** by **modulating pseudo-random noise signal**.
• This standard exhibits high **tolerance to noise and interference** improving link reliability.
• The method for channel access implements Carrier Sense Multiple Access with collision avoidance (CSMA-CA)
• Temporal multiplexing is also used to enable access to the same channel by multiple users or nodes at different times.

DSSS is a spread spectrum technique that spreads the signal bandwidth over a wider frequency range than the original data signal. DSSS offers several benefits such as increased resistance to interference, improved signal quality, and enhanced security.

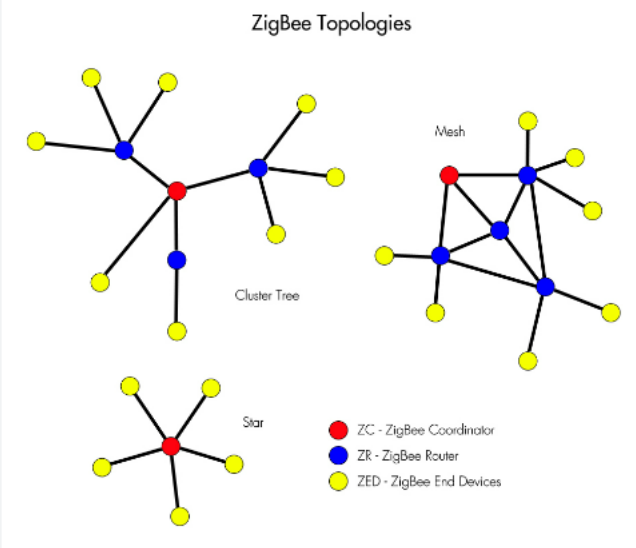
### IEEE 802.15.4 : Direct Sequence Spread Spectrum (DSSS)

**DSSS spreads a signal over a wider bandwidth using a spreading code that is a unique sequence of digital chips that is added to the original signal to spread it over a higher frequency and wider bandwidth.**
**Transmitter and receiver use the same spreading code to both modulate and demodulate the signal.**
The *spreading code* is a unique sequence of digital chips that is added to the original signal to spread it over a higher frequency and wider bandwidth**.**
**The receiver must know the spreading code used by the transmitter, achieved using a synchronization sequence or a preamble transmitted before the actual data which allows the receiver to acquire the spreading code and despread the signal.**
**DSSS enables a wider bandwidth of operation with enhanced security.**

## Explain the Direct Sequence Spread Spectrum modulation technique, and which is the code that sender and receiver use while modulating and demodulating the signal.

• DSSS spreads a signal over a wider bandwidth using a ***spreading code***
• The ***spreading code*** is a unique sequence of digital chips that is added to the original signal to spread it over a higher frequency and wider bandwidth
• **Transmitter and receiver use the same spreading code to modulate and demodulate the signal, respectively**
• The receiver **MUST know the spreading code** used by the transmitter
• This is achieved using a synchronization sequence or a preamble transmitted before the actual data which allows the receiver to acquire the spreading code and spread the signal.

**How do the beacon and non-beacon mode work in Zigbee , describe the role of the coordinator.**




In a ZigBee network, the coordinator is the device that initiates and manages the network. It is responsible **for forming the network, assigning addresses to devices, and controlling the network's behavior**. The purpose of the coordinator is to ensure that all devices in the network can communicate with each other and that the network operates efficiently and reliably.

Some of the key functions of the coordinator in a ZigBee network include:

**1.Network Formation**: The coordinator is responsible for forming the network by initiating the process of joining devices. When a new device joins the network, the coordinator assigns a unique address to the device and adds it to the network.

**2.Address Assignment**: The coordinator assigns unique addresses to devices in the network.

**3.Beacon Management**: The coordinator sends out periodic beacon frames to synchronize the network and provide timing information for devices.

**4.Security Management**: The coordinator manages the security of the network by assigning security keys to devices and enforcing security policies.

**5.Routing Management**: The coordinator is responsible for managing the routing of data between devices in the network, including selecting the best path for data transmission and optimizing network performance.

A typical Zigbee network structure can consist of three different device types, namely Zigbee coordinator, Router, and End device.

Every Zigbee network has a minimum of **one coordinator** device who also acts as network router. The coordinator performs **data handling and storing operations.** It initiates, terminates, or routes communication around the network.

The **Zigbee routers** play the role of intermediate nodes that connect two or more Zigbee devices, which may be of the same or different types.

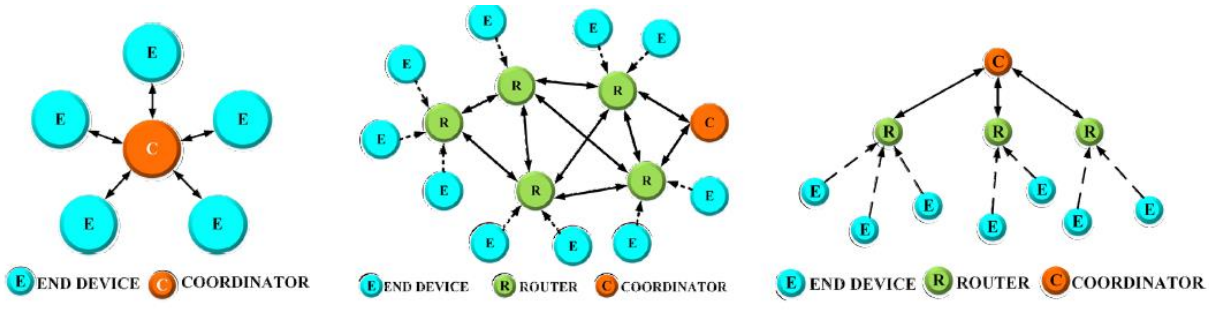
Finally, the **end devices** have **restricted functionality**, and communication is limited to the parent nodes. This reduced functionality enables them to have a lower power consumption requirement for an extended duration.

There are provisions to operate Zigbee in different modes to save power and prolong the deployed network lifetime.

**Zigbee handles two-way data transfer using two operational modes, Non-beacon mode, and Beacon mode.**

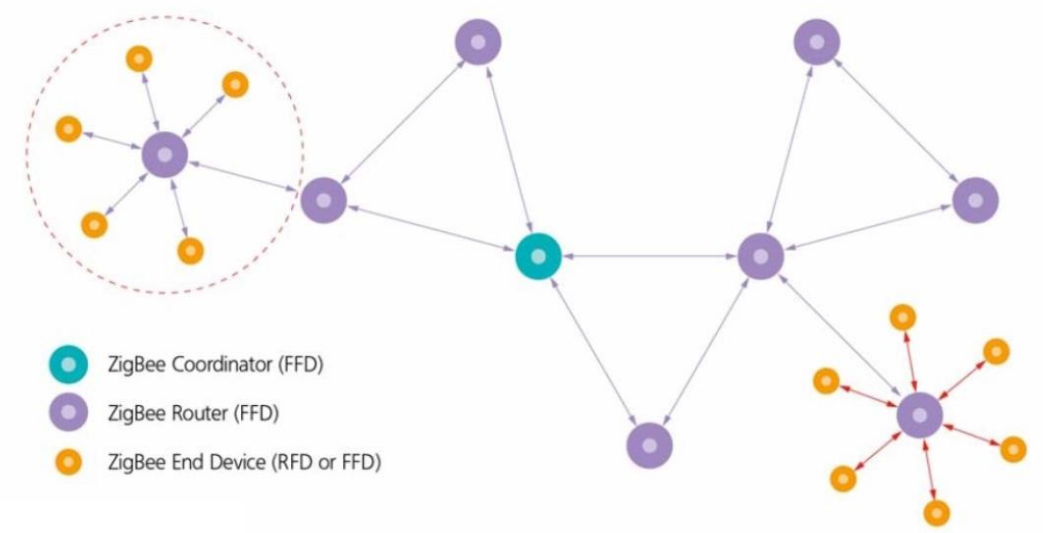
**In the non-beacon mode the coordinators and routers monitor the active state of the received data continuously. In this mode, there is no sleep mode for the routers and coordinators (intensive power mode).**

**The beacon mode allows the coordinators and routers to launch a very low-power sleep state, during the absence of data communication from end devices. The Zigbee coordinator is designed to periodically wake up and transmit the beacons to the available routers in the network.**




(a) Star (b) Mesh (c) Cluster-Tree

## Communication topologies in Zigbee




### FANETs (Lesson 11)

## Describe the tradeoff between application availability and latency for communications in FANETs and which are the possible solutions?

**To reduce the latency in communication between drones and the base station, it would be optimal for them to get closer to it, avoiding possible interferences, but doing so, we'd get a lower application availability to cover the interested area. A possible solution is based on data ferries: part of the drones are treated just as data ferries, they carry the data exploiting long range communications, the other are only used to monitor the field of interest.**

In Flying Ad-Hoc Networks (FANETs), the tradeoff between application availability and latency arises due to mobility and limited resources. UAVs constantly move, causing changes in network topology and uncertain communication availability. Interference in crowded airspace further affects latency. To address this tradeoff, efficient routing protocols, QoS mechanisms, mobility management, adaptive transmission power, and redundancy can be employed. The specific solutions depend on application requirements and network conditions. Analyzing needs and network characteristics is crucial for balancing application availability and latency in FANETs.

**Constrained Networks (Lesson12)**

## Define a constrained device.

**There are three classes of constrained devices.**

**Class 0: these devices are severely constrained regarding resources and capabilities. The barely feasible memory and processing available to these classes of devices do not allow for direct communication to the Internet. They need a gateway or a proxy.**

**Class 1: these devices are constrained concerning available code space and processing power. These devices can primarily talk to the Internet, but cannot employ a regular full protocol stack such as http: they use special protocols such as CoAP. They don't need a gateway to connect to the internet**

**Class 2: these devices are functionally similar to regular portable computers such as laptops. They have the ability and capacity to support full protocol stacks of commonly used protocols such as http. Compared two the first classes they have high power budget.**

Any network in which typical link layer features are extremely constrained

- Low bit rate
- High and variable packet loss
- Low performance with large packets due to link layer fragmentation

A constrained device refers to a type of computing or communication device that operates under certain limitations and constraints, typically with respect to its resources, capabilities, or power consumption. These devices are designed to perform specific functions within predefined boundaries and often have restricted processing power, memory, energy supply, and communication capabilities.

1. **Limited Processing Power:** Constrained devices operate with constrained computational capabilities, often relying on low-power processors or specialized microcontrollers optimized for specific tasks.
2. **Memory Constraints**: These devices have limited volatile (RAM) and non-volatile (storage) memory, typically constrained for cost, power, or physical size considerations.
3. **Power Constraints**: Constrained devices are designed to operate within specific power limitations, often relying on batteries or low-power sources. Energy-efficient designs are crucial to maximize battery life and minimize power consumption.
4. **Communication Limitations**: Constrained devices may have restricted communication capabilities, such as limited bandwidth, range, or supported protocols. Wireless standards like Bluetooth Low Energy (BLE), Zigbee, or LoRa are commonly used due to their optimization for low-power and low-data-rate applications.

5. **Physical Constraints:** Constrained devices are designed to fit specific form factors or embedded systems with limited physical space. They often feature compact designs, integrated sensors, and customized hardware to meet specific application requirements.

Constrained devices are widely used in IoT applications, serving as connected endpoints for data gathering, localized processing, and communication with other devices or cloud services. Examples include sensor nodes, smart appliances, wearables, industrial sensors, and embedded systems used in automation and control.

## What are the mesh under and route over routing modes in 6LoWPAN ?

**In IPv6 subnetworks, some datagrams could be too big and needed to be fragmented in more packets. With the mesh_under method, the fragments are reassembled at the destination (if a fragment is missing the message need to be re-sent from source), while in route over routing modes their reassembled at every hop (if a fragment is missing the message is re-sent from previous node)**

In 6LoWPAN, two routing modes are used: Mesh Under and Route Over.

1. **Mesh Under Routing**: Implemented at the MAC layer, it relies on neighbor information for routing decisions. Suited for small networks with simpler routing needs.

It uses the MAC addresses of devices to determine the next hop for data packet forwarding within the network. This approach simplifies routing and reduces resource requirements, making it suitable for small-scale deployments with constrained devices. However, it lacks global routing capabilities and is limited to the immediate neighborhood of each node.
A mesh-under network is a **single IP subnet** with a single edge router.
² Useful for small or local networks

2. **Route Over Routing**: Uses the layer-three (**IPv6**) addresses to forward data packets.

² IPv6 addresses must be routable (**Global** only).
² Deploy scalable, large-scale networks.
It allows 6LoWPAN devices to communicate by encapsulating their packets within IP packets destined for the 6LoWPAN network. This approach simplifies routing implementation, enhances interoperability, and overcomes limitations of link-layer routing in low-power wireless networks.

The choice between modes depends on network requirements. Mesh Under is simpler but limited in scalability, while Route Over supports complex routing and larger networks. Factors like network size, complexity, and available resources impact the selection. Routing mode affects overall network performance and efficiency in 6LoWPAN.

**Explain how RPL provides the construction of a DODAG and where do the DODAG nodes store the forwarding information in both storing and non-storing mode?**

# UPWARD Path:

## ❑ Creation of the **upward paths** (assumed at start-up)

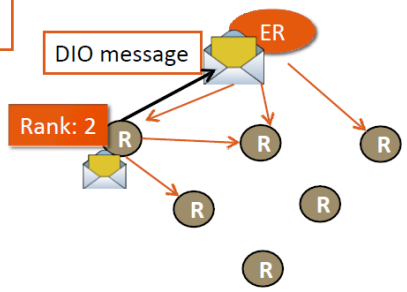
1. The Edge router creates the **DIO** message, containing its rank and DODAG id, and sends it in **multicast.**

RECEIVING **NODES**

2. Each node establishes the upward link toward **the sender.**

3. Each node computes its own rank value, based on the **root's rank and on the Objective Function.**

4. Each node rebroadcasts the DIO message by including its own computed rank**.**

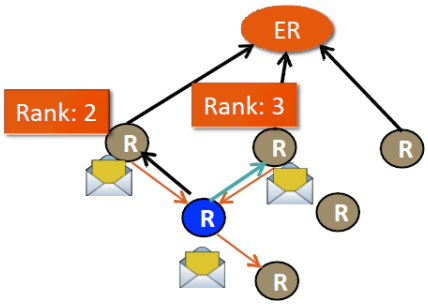



## ❑ Creation of the **upward paths** (assumed at start-up)

A node receiving multiple DIO messages (e.g the blue node)

2. Based on the used metric and constraints defined by the Objective Function, it chooses an appropriate parent:
   - Multiple parents can be established, but a **preferred parent** is selected;
   - If the node has already its own rank, and the received one is greater than the local rank, the DIO message is discarded (**loop avoidance**)

3. As before, each node rebroadcasts the DIO message by including its own computed rank**.**




The routing procedure ends when reaching the leaf nodes.

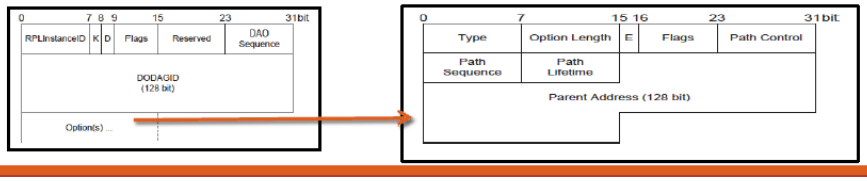
# DOWNWARD Path :

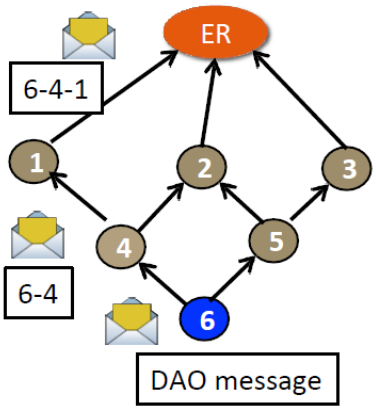
## ❑ Creation of the **downward paths** (from leaf to edge router)

NON-STORING MODE

1. Each node periodically generates a DAO message and sends it to the destination, by using the upward path established through the DIO message.

2. All the intermediate parents extend the DAO message by adding their IPv6 address in the **Transit Information Option.**



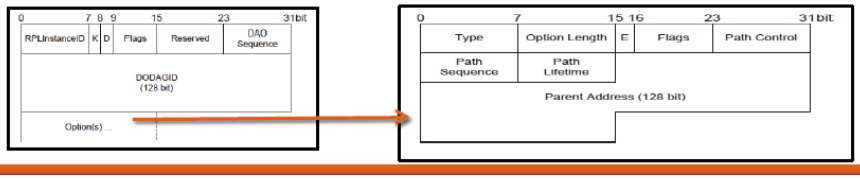



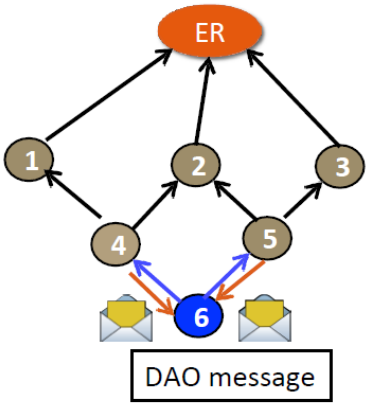

## ❑ Creation of the **downward paths** (from leaf to edge router)

STORING MODE

1. Each node periodically generates a DAO message and sends it to all parents node (differently to the previous case, the message is not forwarded toward the root).

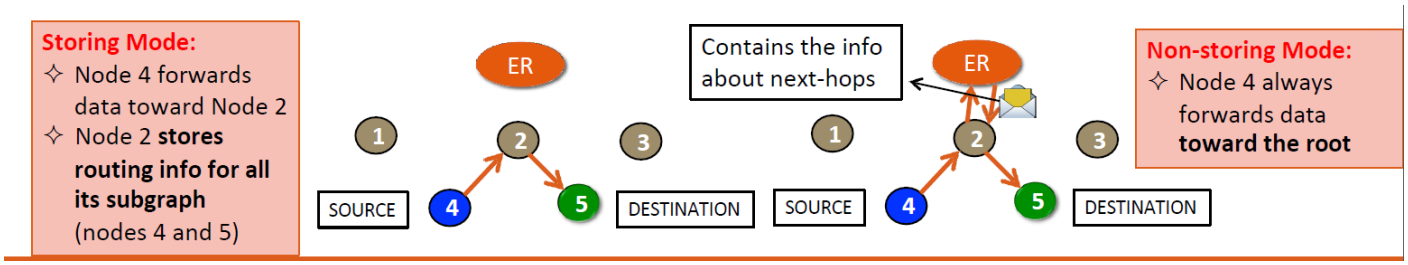
2. Each parent maintains additional routing tables for all the nodes of its sub-DODAG.




Two modes of operation: **storing** and **non-storing**

- **Storing -** each node keeps a **routing entry** for all the destinations reachable via its sub-DODAG.
- **Non-Storing -** the root is the only network node maintaining routing information; source routing is used for downward routing

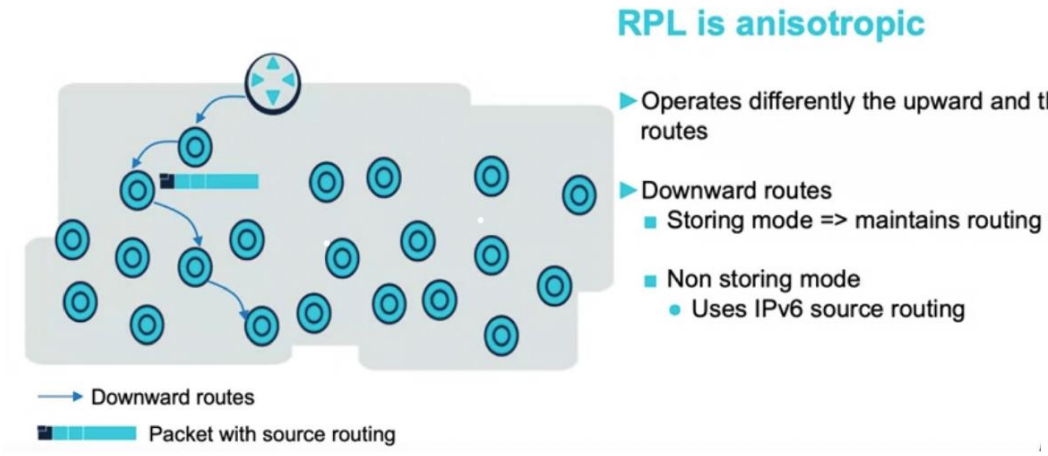



## RPL data plane

- Point-to-point communication
  - In storing mode:
    - Forward upward until a common ancestor
    - Then route downward to the destination
  - In non-storing mode:
    - Forward to the root
    - The root specifies the source routing towards the destination

## Why in RPL anisotropic ?

**Because it operates differently when constructing the upward and downward routes in DODAG (they're not symmetric)**




In RPL (Routing Protocol for Low-Power and Lossy Networks), "anisotropic" refers to the asymmetric link quality between nodes

## What are the control messages used by RPL and what is their purpose in the protocol.

RPL (Routing Protocol for Low-Power and Lossy Networks) utilizes several control messages to facilitate network management, routing establishment, and maintenance. These control messages serve specific purposes within the protocol. Here are some commonly used control messages in RPL:

1. DODAG Information Solicitation (DIS): DIS messages are sent by a node to discover neighboring nodes and gather information about the DODAG (Destination-Oriented Directed Acyclic Graph) structure. DIS messages prompt neighboring nodes to respond with DODAG Information Object (DIO) messages.
2. DODAG Information Object (DIO): DIO messages are sent by a DODAG root node and disseminated throughout the network. DIO messages contain essential information about the DODAG, including DODAG configuration, routing metrics, and other parameters. Nodes use this information to construct and maintain their routing tables and make informed routing decisions.
3. Destination Advertisement Object (DAO): DAO messages are used by non-root nodes to propagate information about specific destinations to their parent nodes. DAO messages carry route information from a node to its parent, indicating its intention to route traffic for a particular destination. DAO messages help in establishing and updating routing paths within the DODAG.
4. DODAG Information Solicitation (DIS) Acknowledgment (DIA): DIA messages serve as acknowledgments to DIS messages. When a node receives a DIS message, it may send a DIA message back to the sender, indicating that it has received the DIS message. DIA messages confirm successful reception and provide additional information about the receiving node.
5. DODAG Repair Information Object (DRIO): DRIO messages are used in situations where the DODAG structure is disrupted or needs repair due to node failures or network changes. DRIO messages carry information about the need for DODAG repair and provide instructions to affected nodes on how to repair or reconstruct the DODAG.

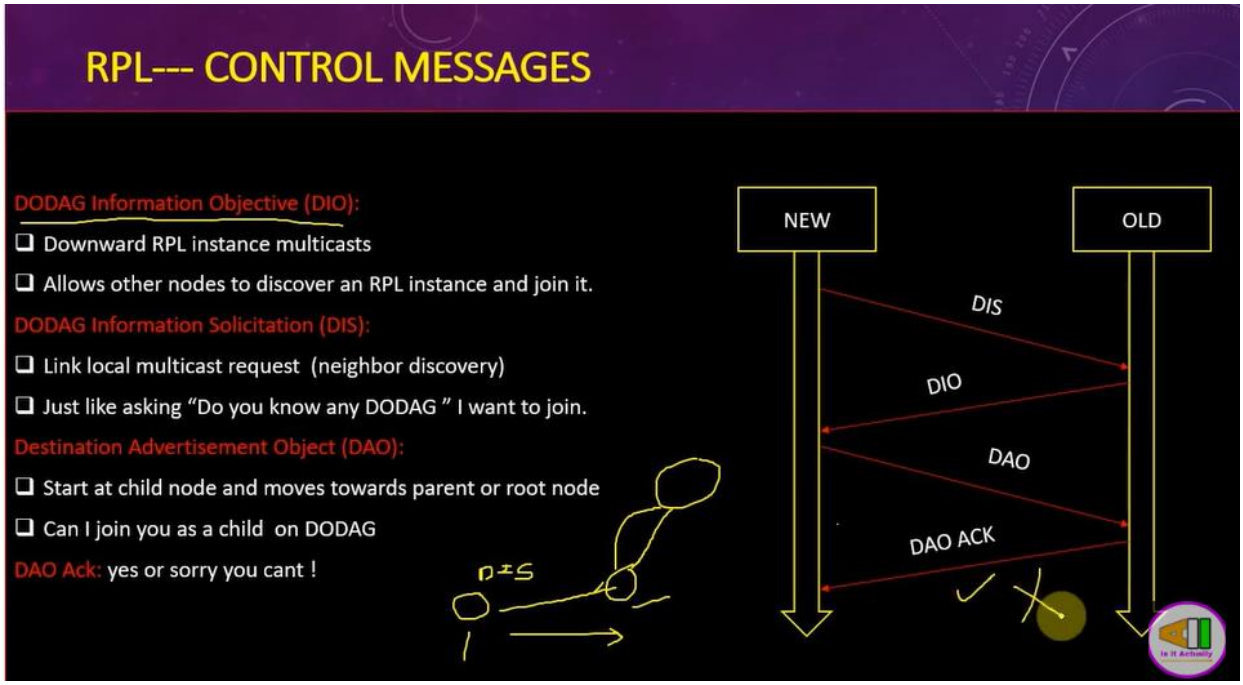
These control messages in RPL enable the exchange of vital information between nodes for the construction, maintenance, and repair of the DODAG. They support the establishment of efficient routes, propagation of routing information, discovery of neighboring nodes, and network management tasks. By exchanging control messages, RPL nodes can coordinate and adapt their routing behavior to maintain robust and scalable communication within low-power and lossy networks.

**DIS** messages are sent by a node to discover neighboring nodes and gather information about the DODAG structure.

**DIO** messages are sent by a DODAG root node or edge router and spread throughout the network ( Multicasted downwards, lets others know about the node)

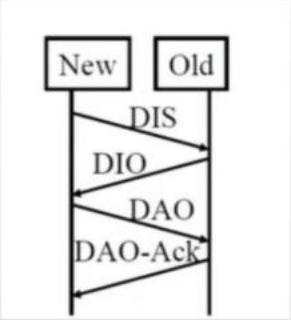
**DAO** messages are used by **non-root nodes** to propagate information about specific destinations to their parent nodes ( a request by a child to parent or root : can I join u as a child in DODAG ? )

DAO Ack : a response that is sent by a parent or a root : can be Yes or No.




## RPL Control-messages:

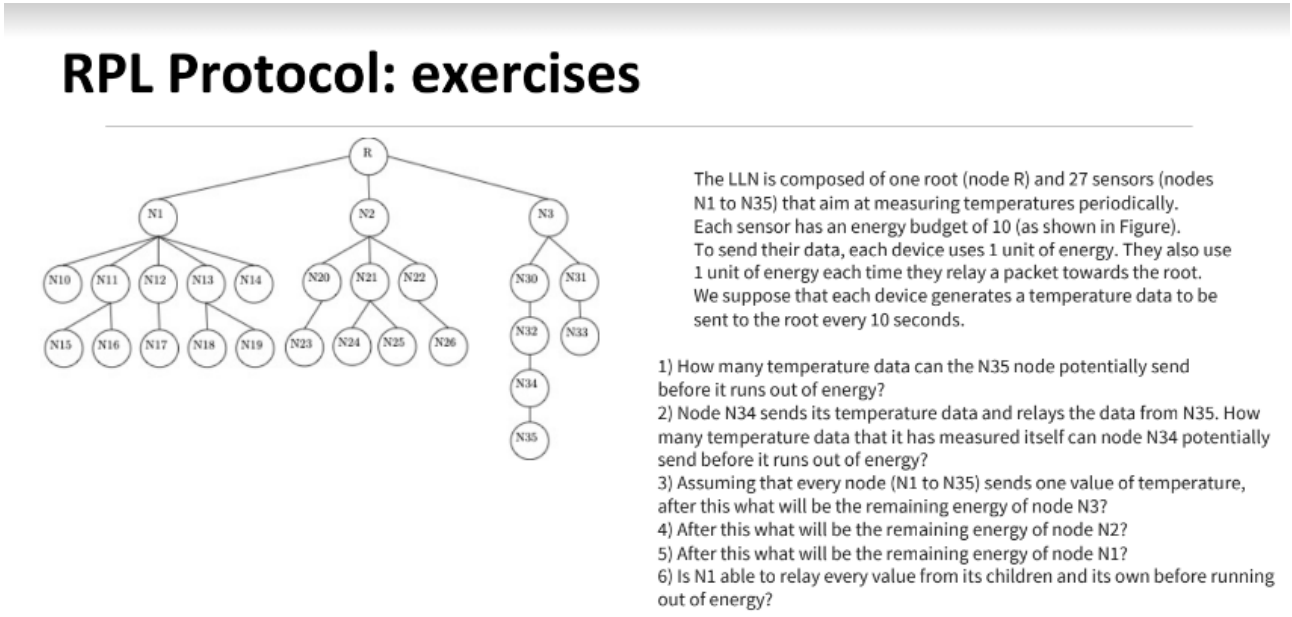
- There are 5 Control Messages, they form the Spanning tree :
- **DODAG information Object (DIO)** : This message is Multicasted downwards . A given node In a DODAG may multicast this message , which lets other nodes know about it , Things like whether the node is grounded or not, whether it storing or non-storing, and it announces other nodes "if they are interested to join , Please Let Me know. "
- **DODAG Information Solicitation (DIS)** : When no announcement is heard, and if a node wants to join a DODAG it sends a control message , For that it wants to know If any DODAG exists. So the message which it sends is Like " Is there any DODAG ? "
- **DODAG advertisement Object (DAO)** : It is A request send by a Child to parent or root. This message requests to allow the child to join to a DODAG.
- **DAO-ACK** : It is a response Send by a root or parent to the child , this response can either be a Yes or No.
- **Consistency check** : deals with security , and We need not to know much about it now.



## Explain how RPL guarantees a loop-free routing in a network of constrained devices.

The **rank** indicates the relative position of a node within the DODAG. It is used to build routes and avoid loops.
For nodes A and B:
**If the rank of A is less than the rank of B**: A is closer to the root than B. B can then choose A as parent without risking to create a loop.
**If the rank of A is equal to the rank of B**: nodes A and B have similar position in the DODAG and routing through those nodes should be avoided as it would create a loop.
**If the rank of A is greater than the rank of B**: B is closer from the root than A. B won't choose A as parent to avoid creating a loop.
Nevertheless, the instability of LLNs can still leads to loops. RPL implements mechanisms to detect and repair loops.

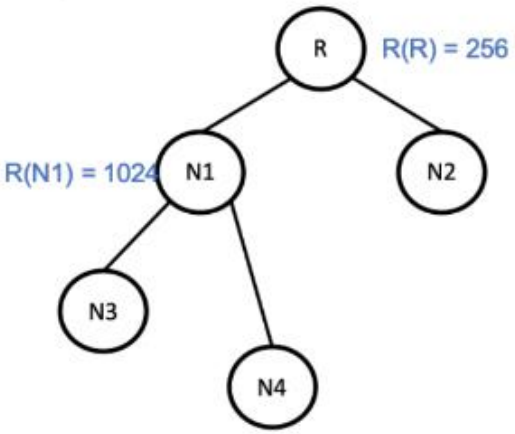
## Exercise on RPL with constrained energy nodes.




1) 10
2) 5
3) 3
4) 2
5) And 6) no

**Exercise on RPL where you are required to compute the rank of a given node, based on different objective functions (OF0 and OF1)**

# RPL Protocol: OF0, exercise




- DEFAULT_STEP_OF_RANK = 3
- MINIMUM_STEP_OF_RANK = 1
- MAXIMUM_STEP_OF_RANK = 9
- DEFAULT_RANK_STRETCH = 0
- MAXIMUM_RANK_STRETCH = 5
- DEFAULT_RANK_FACTOR = 1
- MINIMUM_RANK_FACTOR = 1
- MAXIMUM_RANK_FACTOR = 4
- DEFAULT_MIN_HOP_RANK_INCREASE = 256
- the RFC6550 sets the rank of the root to the value MinHopRankIncrease

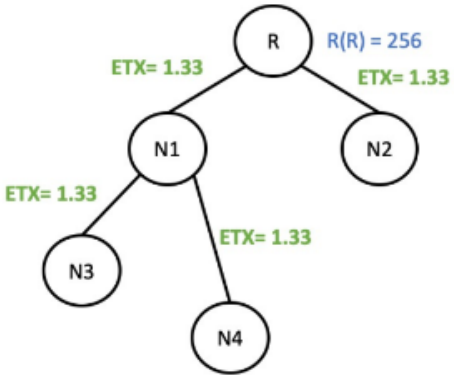
1) What is the rank of N2?
2) What is the rank of N3?
3) What is the rank of N4?

R(N1) = R(R) + (Rf*Sp + Sr) * MinHopRankIncrease
= 256 + (1*3+0) * 256
= 1024

1) R(N2) = 256 + (1*3 + 0) *256 = 1024
2) R(N3) = 1024 + ((1*3 + 0) *256) = 1792
3) R(N4) = R(N3)

# RPL Protocol: OF1 MRHOF, exercises

In the following, we will consider: **rank_increase = (Rf*Sp + Sr) * MinHopRankIncrease**
- a rank factor *Rf*=1,
- a step of rank given by *Sp*=(3∗*ETX*−2),
- a stretch of rank *Sr*=0,
- MinHopRankIncrease = DEFAULT_MIN_HOP_RANK_INCREASE = 256




1) When the last 100 data have been transmitted, 100 have been received but only 75 acknowledgements have been received. So *Df*=1 and *Dr*=0.8. What is the value of the ETX metric of the link ?

2) With the given values of ETX = 1.33, calculate the value of *Sp* rounded up to the nearest integer.

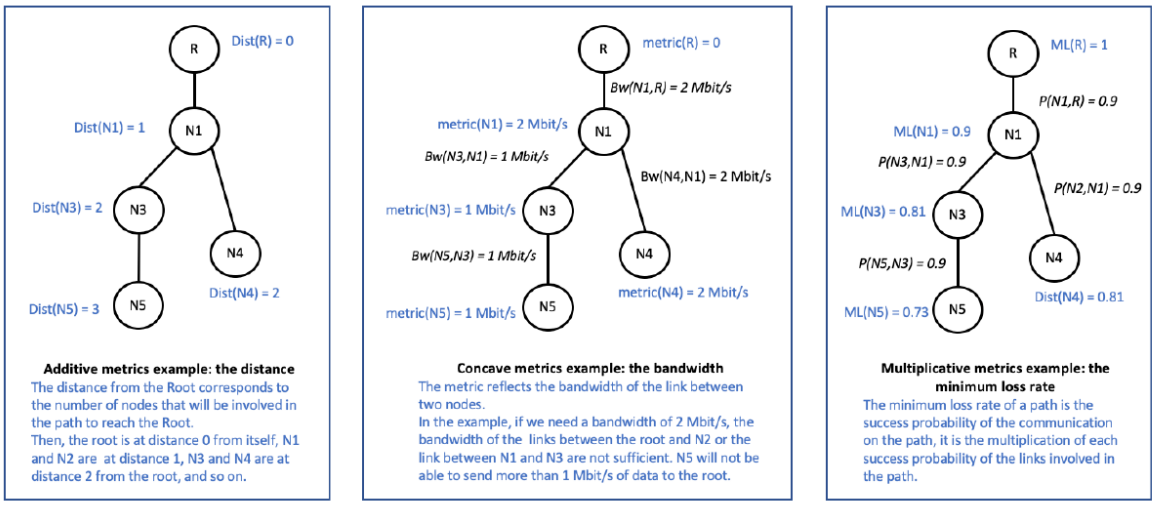
3) What is the value of the rank computed for N1 (knowing that the rank of the root is 256)?

4) What is the value of the rank computed for N3?

1) ETX = 1/df * dr (in the pic used dr = 0,75, don't know why in the text said it's 0,8) -> ETX = 1.33
2) Sp = (3*1.33 – 2) = 1,99 = 2 (text says to round up to integer)
3) R(N1) = 256 + (1 * 2 + 0) * 256 = 768
4) R(N3) = 768 + (1*2 + 0) *256 = 1280

**Show examples of additive, concave and multiplicative metrics that could be used by RPL to determine the shortest path between a source and a destination.**

# RPL Protocol: metrics for shortest path selection




**Additive metrics example: the distance.**

**The distance from the root indicates the number of nodes involved to reach the root;**

**concave metrics example: the bandwidth**

**the metric is the bandwidth between the nodes**

**multiplicative metric example: minimum loss rate**

**the minimum the loss rate, the best is the probability of success of the communication (multiplication of all probabilities)**

## Application Level protocols for the IOT ( Lesson 13 )

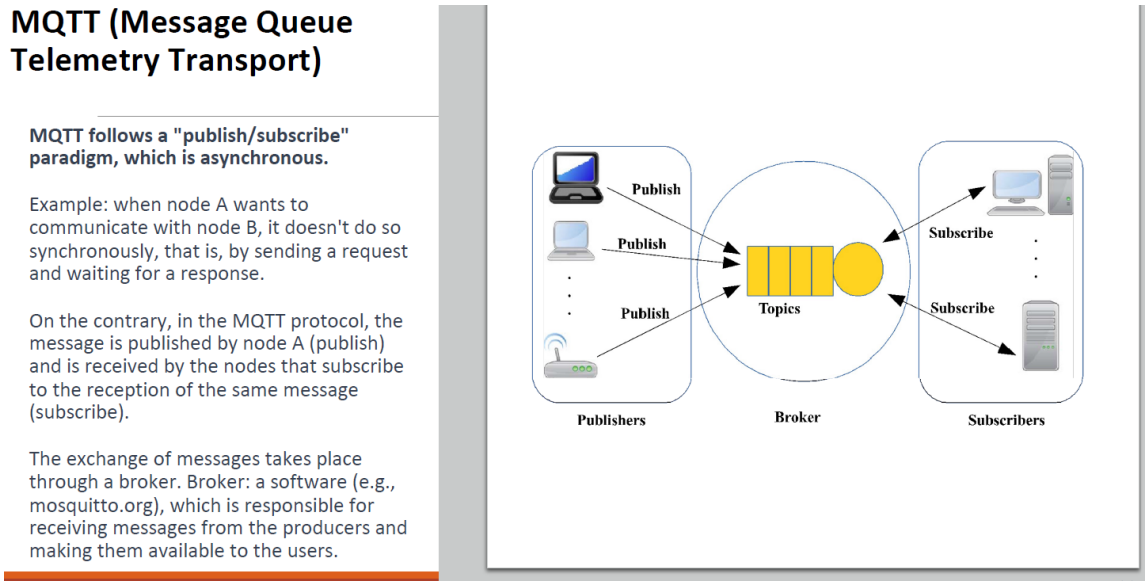
## Describe three application layer protocols used for IOT, underlining the key differences in terms of interaction paradigm between the involved entities.

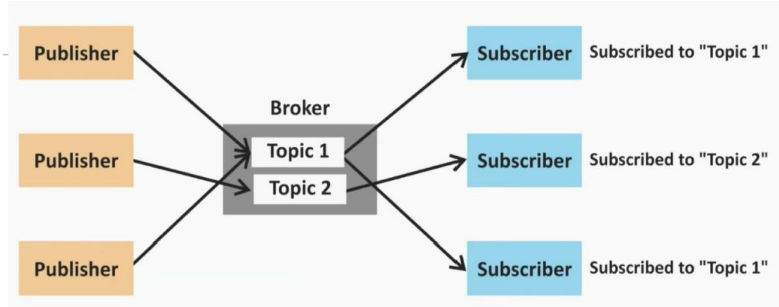
**CoAP: It's Messaging protocol for use with constrained nodes and constrained networks. Differently from MQTT, CoAP implements a request-response interaction model (similar to the HTTP protocol). It has a RESTful architecture for Costrained Environments. Each resource is addressed by an URI (Uniform**

**Resource Identifier). A server is used by a client knowing a URI that references a resource in the namespace of the server. It achieves reliability with Duplicate detection and stop-and-wait retransmission with exponential backoff (only confirmable messages).**

**MQTT: M2M communication protocol. Publishers produce data and send them to a broker. Subscribers subscribe to a topic of interest, and receive notifications when a new message for the topic is available. Broker filter data based on topic and distribute them to subscribers. (unlike CoAp, not request-response). It provides reliable communication using TCP (CoAP uses UDP).**

**AMQP: Very similar to MQTT, but brokers are composed of an exchange and queues. Once the broker gets data from the publisher, then put them in the queues following specific rules. Then the data is sent to the subscribers.**




- Topic #1: `home/groundfloor/kitchen/temperature`
- Topic #2: `office/conferenceroom/luminance`

The Internet stack (TCP/IP) adds a considerable amount of control data (overhead) that is too burdensome for constrained devices (hundreds or thousands of bytes). CoAP greatly reduces the overhead to just a few tens of bytes, enabling low-energy consumption communication an

functioning in the presence of interference-prone links. Conversion between these two protocols is possible through REST-CoAP proxies. CoAP is designed for use between devices on the same constrained network (between two smart devices), between traditional devices on the internet, and between devices on different constrained networks, all connected through the internet.
The interaction model of CoAP is similar to the **client/server model of HTTP.** The CoAP request is equivalent to that of HTTP and is sent from a client **to request an action** (using a method code) on resource (identified by a URI) on a server. The server then sends a **response with a response code**; this response can include a representation of the resource. Unlike HTTP, CoAP handles these exchanges **asynchronously over a datagram-oriented transport like UDP.** EXAMPLE: the GE method can be used by a server to request the temperature from the client using the piggyback response mode. The client returns the temperature if it exists; otherwise, it responds with a status code to indicate that the requested data was not found.

## Describe CoAP, what is its purpose and main features.

The Constrained Application Protocol (CoAP) is a specialized lightweight application layer protocol designed for constrained devices and constrained networks in the context of the Internet of Things (IoT). It is intended to provide a simple, efficient, and scalable communication protocol for resource-constrained devices with limited processing power, memory, and energy resources.

The IETF Constrained RESTful Environments (CoRE) working group has created CoAP, an application-layer protocol for IoT applications.

It is a **data transfer protocol** suitable for **devices and networks with limited capabilities** (energy, computation, memory, low power, packet loss).

CoAP is based on a **Request/Response model** (client/server) between communication endpoints.

It is designed to **easily interface with HTTP** for integration with the Web while also meeting specialized requirements such as multicast support, very low overhead, and simplicity for constrained environments..

# CoAP: light and fast variant of HTTP

CoAP significantly reduces the amount of exchanged bytes for communication between two devices.

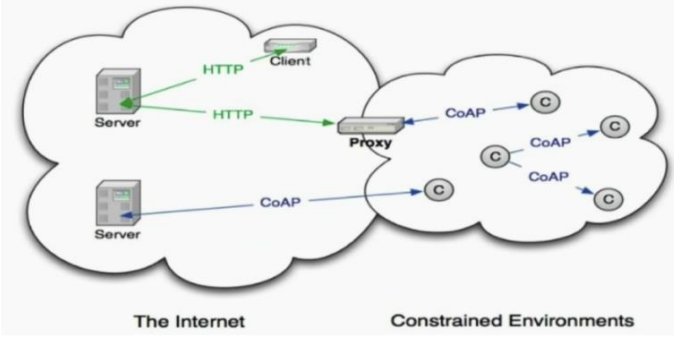
The Internet stack (TCP/IP) adds a considerable amount of control data (overhead) that is too burdensome for constrained devices (hundreds or thousands of bytes).

CoAP greatly reduces the overhead to just a few tens of bytes, enabling low-energy consumption communication and functioning in the presence of interference-prone links.

Conversion between these two protocols is possible through REST-CoAP proxies.

# CoAP: interacting agents

CoAP is designed for use between devices on the same constrained network (between two smart devices), between traditional devices on the internet, and between devices on different constrained networks, all connected through the internet.



# The COAP Protocol

## ✧ Constrained Application Protocol (CoAP)

- ✧ Messaging protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks.
- ✧ Differently from MQTT, CoAP implements a **request-response interaction** model (similar to the HTTP protocol).
- ✧ **RESTful** architecture for Costrained Environments (**CoRE**).
- ✧ Each resource is addressed by an **URI** (Uniform Resource Identifier).

## ✧ Each resource is addressed by an **URI** (Uniform Resource Identifier).

```
coap://dante.cs.unibo.it/temperature/serverRoom
```

DIFFERENCES COMPARED TO THE HTTP PROTOCOL

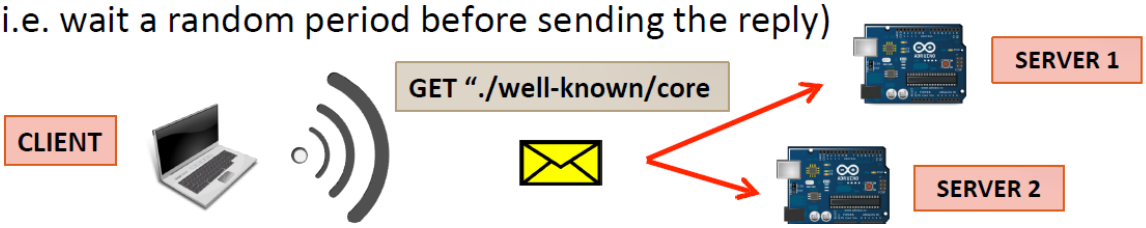
- ✧ Based on the **UDP** protocol (but optional mechanisms can be used for **enhanced reliability**, i.e. Confirmable messages + Retransmissions)
- ✧ Asynchronous Request/Response paradigm
- ✧ Different (Shorter) Packet Header
- ✧ Service Discovery and Proxy mechanisms

## ✧ CoAP implements some lightweight reliability mechanisms:

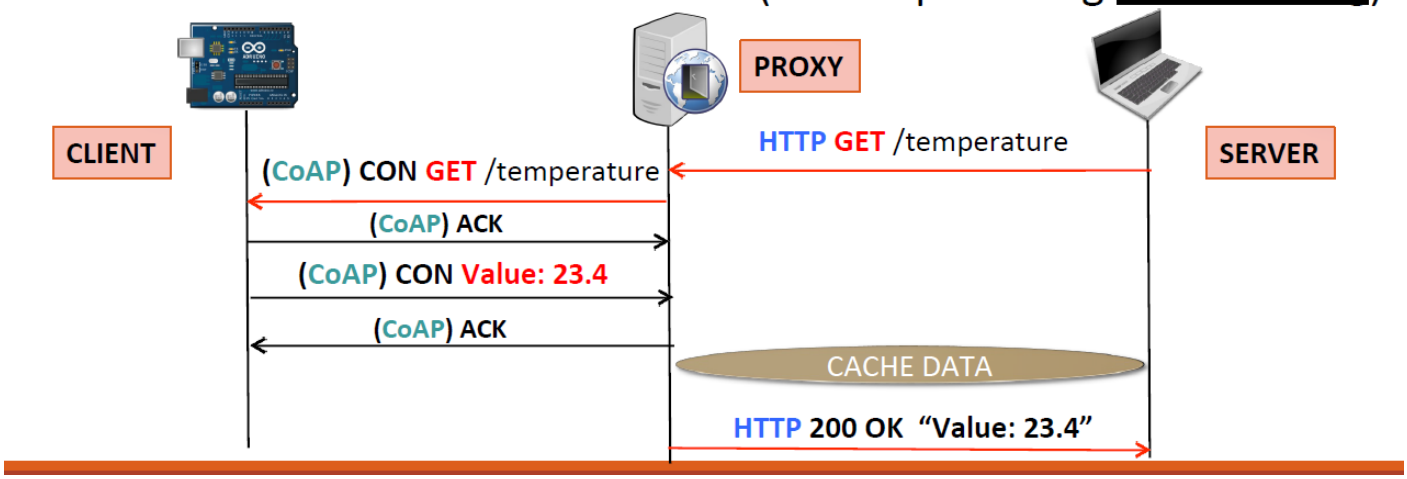
- ➢ **Duplicate detection** for both Confirmable (CON) and Non-Confirmable (NON) messages
- ➢ Simple **stop-and-wait retransmission reliability** with **exponential back-off** for Confirmable messages

✧ The sender retransmits the Confirmable message at exponentially increasing intervals, until it receives an ACK (or RST message) or runs out of attempts.

- A **server** is used by a client knowing a URI that references a resource in the namespace of the server.
  - Alternatively, clients can use **multicast CoAP requests** (on the default port 5683) the "All CoAP Nodes" multicast address to find CoAP servers
  - Multicast requests are NOT Confirmable (i.e. no ACK messages are sent).
  - If a server does decide to respond to a multicast request, it should back-off (i.e. wait a random period before sending the reply)




- CoAP only supports a limited subset of HTTP functionality,
- However, cross-protocol proxy mechanisms can guarantee seamless **HTTP-CoAP interactions** (beside providing data caching).




1. Purpose of CoAP: CoAP serves as a machine-to-machine (M2M) protocol for IoT devices to interact and exchange data in a constrained manner.
2. Lightweight: CoAP is designed to be efficient, with smaller code and message size compared to traditional protocols like HTTP, conserving network resources.
3. Request-Response Model: CoAP follows a simple request-response interaction model similar to HTTP, enabling familiar communication paradigms.
4. URI Support: CoAP uses URIs to identify and locate resources, allowing for resource access and manipulation in a RESTful manner.
5. RESTful Integration: CoAP integrates well with REST principles, enabling resource-oriented communication and following a stateless client-server model.
6. CoAP Options: CoAP introduces options for additional functionality, such as content negotiation, observe functionality, and block-wise transfers.
7. CoAP Observing: CoAP supports observing resources, allowing clients to subscribe to changes in resource states for efficient real-time updates.

8. Security: CoAP can be secured with DTLS, providing end-to-end security for data confidentiality, integrity, and authentication. Overall, CoAP provides a lightweight, efficient, and scalable protocol for communication in IoT environments, catering to resource-constrained devices while minimizing overhead.

## How does CoAP achieve reliability ?

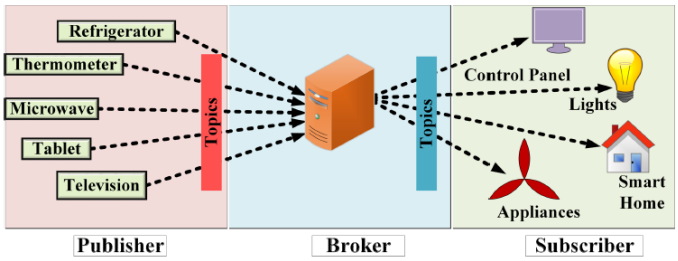
**CoAP: It's Messaging protocol for use with constrained nodes and constrained networks. Differently from MQTT, CoAP implements a request-response interaction model (similar to the HTTP protocol). It has a RESTful architecture for Costrained Environments. Each resource is addressed by an URI (Uniform Resource Identifier). A server is used by a client knowing a URI that references a resource in the namespace of the server. It achives reliability with Duplicate detection and stop-and-wait retransmission with exponential backoff (only confirmable messagges).**

## Describe MQTT, what is its purpose and main features. Does it provide reliable communications?

**MQTT: M2M communication protocol. Publishers produce data and send them to a broker. Subscribers subscribe to a topic of interest, and receive notifications when a new message for the topic is available. Broker filter data based on topic and distribute them to subscribers. (unlike coap, not request-response). It provies reliable communication using TCP (CoAP uses UDP).**

# The MQTT Protocol

- **Message Queuing Telemetry Transport Protocol (MQTT)**
  - **Lightweight messaging protocol** designed for M2M (machine to machine) telemetry in resource-constrained environments.
  - Proposed initially by Andy Stanford-Clark (IBM) and Arlen Nipper in 1999 for connecting Oil Pipeline telemetry systems over satellite.
  - Released Royalty free in 2010 and as OASIS standard in 2014




- Message Queue Telemetry Transport or MQTT is a simple, lightweight publish-subscribe protocol, designed mainly for messaging in constrained devices and networks.
- It provides a one-to-many distribution of messages and is payload-content agnostic.

# The MQTT Protocol

- The MQTT protocol implements a **publish-subscribe messaging** mechanism, involving three main actors:

CLIENTs

- **Publishers** → produce data and send them to a broker.
- **Subscribers** → subscribe to a topic of interest, and receive notifications when a new message for the topic is available.
- **Broker** → filter data based on topic and distribute them to subscribers.

- MQTT works reliably and flawlessly over high latency and limited bandwidth of unreliable networks without the need for significant device resources and device power.
- The MQTT paradigm consists of numerous clients connecting to a server – referred to as a *broker*.
- The clients can have the roles of information publishers (sending messages to the broker) or information subscribers (retrieving messages from the broker).
- This allows MQTT to be largely decoupled from the applications being used with MQTT.
- MQTT is built upon the principles of hierarchical topics and works on TCP for communication over the network.
- Brokers receive new messages in the form of topics from publishers.

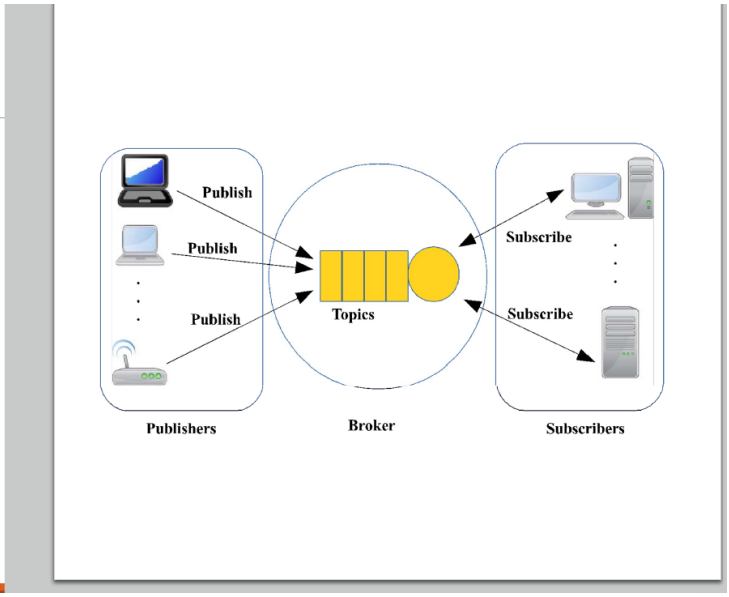
## MQTT (Message Queue Telemetry Transport)

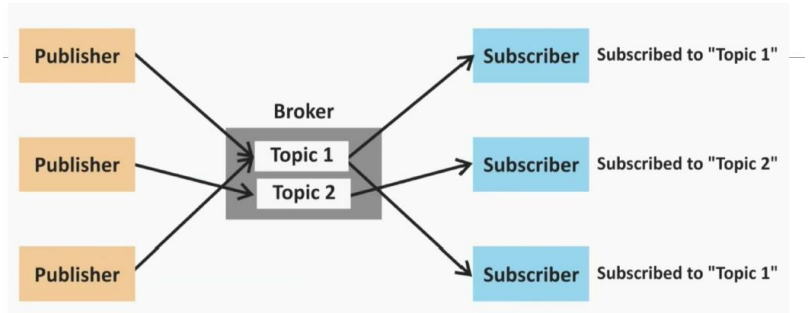
**MQTT follows a "publish/subscribe" paradigm, which is asynchronous.**

Example: when node A wants to communicate with node B, it doesn't do so synchronously, that is, by sending a request and waiting for a response.

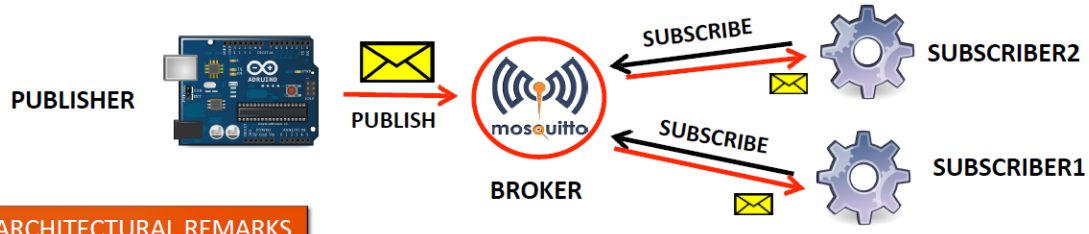
On the contrary, in the MQTT protocol, the message is published by node A (publish) and is received by the nodes that subscribe to the reception of the same message (subscribe).

The exchange of messages takes place through a broker. Broker: a software (e.g., mosquitto.org), which is responsible for receiving messages from the producers and making them available to the users.



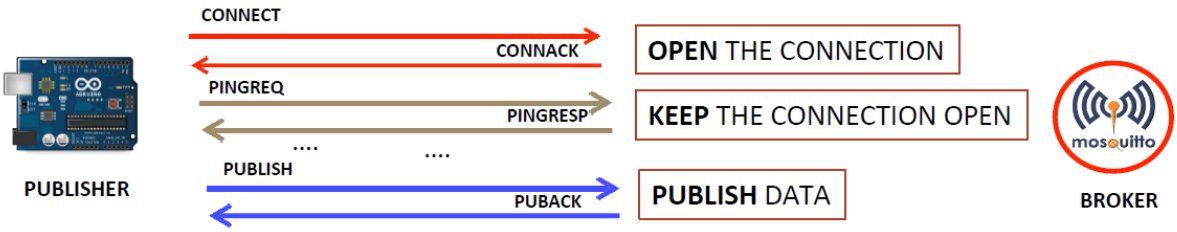

- Topic #1: `home/groundfloor/kitchen/temperature`
- Topic #2: `office/conferenceroom/luminance`




ARCHITECTURAL REMARKS

- A topic defines the **message context** (e.g. temperature data).
- **No direct communication** between clients (the data messages are always forwarded via the broker).
- **Roles are purely logical**: the same device can serve as Publisher (on a topic), and Subscriber (on a different topic).

- MQTT is **built on top of the TCP protocol**
  - In-order delivery, connection-oriented, ACK e retranmissions.
  - .. but also longer TSP header size and higher complexity.
  - **MQTT-SN** → uses **UDP**, supports topic IDs (instead of names).
- MQTT keeps the **TCP** connection between a client and a broker open as long as possible, by means of **PINGREQ** messages.




**WOT (Lesson 13)**

## Describe the key differences between WOT and IOT ?

**The Web of Things (WoT) paradigm enables access and control over IoT resources and applications through the Web. The scope of IoT applications is much broader, which includes non-IP-based systems that are not accessible through the web.**

The key differences between the Web of Things (WoT) and the Internet of Things (IoT) can be summarized as follows:

1. Focus and Scope:
- IoT: Interconnection of physical objects with sensors and connectivity for data exchange and automation in various domains.
- WoT: Extension of IoT, integrating IoT devices into the web ecosystem using web standards and protocols.
2. Interoperability:
- IoT: Ensuring communication and compatibility between devices, platforms, and systems in IoT ecosystems.
- WoT: Enabling interoperability between IoT devices and web services or applications through web protocols and APIs.
3. Protocol and Communication:
- IoT: Utilizes various communication protocols and technologies specific to IoT, such as MQTT, CoAP, Zigbee, and more.
- WoT: Relies on existing web protocols like HTTP, WebSocket, JSON, and RESTful APIs for communication and interaction.
4. Application Development:
- IoT: Involves programming languages, frameworks, and APIs specific to IoT device ecosystems.
- WoT: Leverages web technologies, allowing developers to use existing web development skills and tools for building applications that interact with IoT devices.

In summary, IoT focuses on interconnecting physical objects, while WoT integrates IoT devices with the web infrastructure, emphasizing interoperability, web protocols, and web-based application development.

## Describe the role of HTTP in WOT and how we can support push based IOT applications in WOT?

**WoT Client server architecture is based on http. Web Things must be an HTTP server. With the GET, POST, PUT and DELETE commands the client can make http requests to the servers. We can support push-based applications with Webhooks/HTTP callbacks, where the Web Thing and the Web client serve both as http client and http server, with polling or long polling, where the client sends the HTTP request to the server; the server holds the request till a new value of the resource is available, then it sends a response, or with websockets, where the Client sends an HTTP request to the server, asking for an upgrade to WebSockets; The server replies (if it supports WebSockets), then a bidirectional TCP socket is open and used for the data transfer**

HTTP (Hypertext Transfer Protocol) is vital in the Web of Things (WoT) as a standardized communication protocol for IoT device and service interaction. It enables seamless integration and

interoperability among devices and platforms. HTTP follows a request-response model where clients send requests to servers and receive data or perform actions in response.

To support push-based IoT applications in the WoT, the following mechanisms are used:

1. Polling Mechanism:
- Clients periodically send requests to the server to check for updates.
- However, this can lead to increased network traffic and latency.
2. Webhooks/HTTP Callbacks:
- Clients register a callback URL with the server.
- The server sends an HTTP request to the callback URL to push updates or notifications in real-time.
3. Long-Polling Mechanism:
- Clients send a request to the server, and the server holds the request open until an update occurs.
- This reduces latency and minimizes unnecessary requests.
4. WebSockets:
- WebSockets provide a full-duplex communication channel between the client and server.
- Real-time bidirectional communication is possible without constant request initiation.

Each mechanism offers different trade-offs in terms of simplicity, efficiency, and real-time capabilities. The choice depends on the specific requirements of the IoT application and the capabilities of devices and platforms in the WoT ecosystem.

**Lab Questions**

**What are the two main actors in a ROS service? How do they work together?**

In ROS, services involve two main components: the service client and the service server. They facilitate request-response communication between nodes.

1. Service Client: Initiates a request by sending a message to the service server, specifying the service name and relevant data. It waits for a response.
2. Service Server: Provides the implementation of a specific service. Listens for requests, processes them, and generates a response message. The interaction follows a synchronous pattern:
3. Server waits for requests.
4. Client sends a request with data.
5. Server processes the request and generates a response.
6. Server sends the response to the client. Services enable structured and controlled data exchange and action triggering in ROS. They allow nodes to interact based on predefined services and message types.

**What are the main differences between a topic and a service in ROS ?**

The main differences between a topic and a service in ROS are as follows:

1. Communication Pattern: Topics in ROS follow a publish-subscribe communication pattern, while services follow a request-response pattern.
2. Data Exchange: Topics enable the exchange of messages between multiple nodes in an asynchronous manner. Publishers publish messages on a topic, and any interested subscribers receive those messages. On the other hand, services facilitate direct communication between a single client and a single server, allowing for synchronous request-response interactions.
3. Message Structure: Topics use message structures defined by ROS message types. Messages can be of any defined data type and can carry various kinds of information. Services, on the other hand, use service types, which consist of a pair of messages: one for the request and one for the response. These messages define the format of the data exchanged between the client and server.
4. Continuous Streaming vs. One-time Interaction: Topics provide continuous streaming of data, where publishers continuously publish messages, and subscribers receive those messages as they are published. In contrast, services are used for one-time interactions where the client sends a request to the server, and the server responds with a single response.
5. Message Queuing: Topics support message queuing, meaning that if a subscriber is not actively receiving messages, the messages are queued until the subscriber becomes available. Services do not have message queuing since they operate on a request-response basis, and the client waits for a response from the server.
6. Node Independence: Topics allow for loose coupling between nodes, as publishers and subscribers do not necessarily need to be aware of each other's existence. On the other hand, services establish a direct connection between a client and a server, and the client needs to be aware of the server's existence to make a service call.

In summary, topics are suitable for continuous streaming of data among multiple nodes, while services are used for one-time request-response interactions between a client and a server. Topics provide asynchronous communication, while services provide synchronous communication. The choice between topics and services depends on the specific requirements and nature of the data exchange in a ROS system.

**How would you program a drone using ROS? Describe its main components using the ROS graph.**

To program a drone with ROS, you use ROS packages and libraries for drone control and communication. The main components in the ROS graph for drone programming are:

1. Sensors: Connect drone sensors (IMU, GPS, camera, LiDAR) to the onboard computer or flight controller to provide data about the drone's state and environment.
2. Drone Driver: Interface with the drone's hardware, translating high-level commands into low-level commands understood by the drone. It communicates with the flight controller or onboard computer for control commands and sensor data.

3. Control Stack: Implement algorithms for drone motion and behavior control. Components include PID controllers, state estimation, trajectory planning, and obstacle avoidance. They receive sensor data, compute control commands, and send them to the drone driver.
4. Navigation Stack: Enable autonomous navigation and path planning. Includes SLAM for mapping, path planning algorithms, and localization methods like AMCL. It takes sensor input and generates navigation commands for the control stack.
5. User Interfaces: Allow interaction with the drone, monitoring its status, and sending high-level commands through GUIs, web interfaces, or command-line tools for manual control or autonomous missions.
6. Communication: Use ROS's publish-subscribe mechanism for inter-node communication. Nodes publish sensor data as topics, and other nodes subscribe to those topics, facilitating integration and communication between drone system modules.

In the ROS graph, nodes handle different tasks such as controlling the drone, receiving sensor data, processing data, and communication. The drone controller node sends control commands, sensor nodes handle sensor data, a teleoperation node enables remote control, and communication happens through topics. Visualization and debugging tools like RViz aid in understanding and analyzing the drone's behavior.

Programming a drone with ROS involves leveraging existing packages and developing custom modules to interface with hardware, implement control algorithms, enable navigation, and provide user interfaces for commanding and monitoring the drone.

**What is an SDF file? Briefly describe how it is made what it is used for.**

An SDF file (Simulation Description Format) is a file used in robotics and simulation environments to describe simulated objects and environments. It is written in XML format and contains details about the structure, properties, and relationships of objects. SDF files are used in robotics simulators like Gazebo, providing a standardized way to create realistic simulations. They define models, environments, links, joints, sensors, visuals, and materials, allowing users to specify the simulation parameters. SDF files are crucial for testing and validating algorithms and systems in a simulated environment before real-world deployment.

- What are the main components of the ROS Graph? How do they interact with each other?
- What are the two main actors in a ROS topic? How do they work together?
- What are the two main actors in a ROS service? How do they work toegether?
- What are the main differences between a topic and a service in ROS?
*- What are ROS interfaces and how are they used?*
- What are ROS parameters? How can they be used?
- What is a ROS ac*on composed of? Describe its main components and how they are used.
- How would you program a drone using ROS? Describe its main components using the ROS graph.

- How would you program a ground rover using ROS? Describe its main components using the ROS graph.
- What is an SDF file? Briefly describe how it is made and what is used for.
- How do ROS and Gazebo work together? Describe their main use and how they can communicate with each other.
- What is ros_gz_bridge? Describe its main uses by also providing some examples.

**LAB QUESTIONS (Giuseppe)**

**- What are the main components of the ROS Graph? How do they interact with each other?**

Nodes, topics, services, parameters and actions.

Each node in ROS should be responsible for a single, module purpose and can send and receive data to other nodes via topics, services, actions, or parameters.

Topics are one of the main ways in which data is moved between nodes and therefore between different parts of the system. A node may publish data to any number of topics and simultaneously have subscriptions to any number of topics.

Services are based on a call-and-response model, versus topics' publisher-subscriber model.

Parameters are node settings

Actions consist of three parts: a goal, feedback, and a result: An "action client" node sends a goal to an "action server" node that acknowledges the goal and returns a stream of feedback and a result.

**- What are the two main actors in a ROS topic? How do they work together?**

Publisher and subscriber. Publishers publish data on the topics, then all the subscribers subscribed to these topics get updates on the status of the applications regarding those topics.

**- What are the two main actors in a ROS service? How do they work toegether?**

Service client and service server. The service client send a specific request to the service server and once the service is executed the server sends back a response.

**- What are the main differences between a topic and a service in ROS?**

services are a more "structured" way of accessing data of a node, and, instead of a stream of continuous data, always consist of a call by an external node, a particular behavior from the receiving node, and a response after the service is executed. (pratically the merged version of the two previous questions)

**- What are ROS interfaces and how are they used?**

In ros the interfaces are the components of the ros graph

OR

Interfaces in ROS are Topics and services

(i dont know, chatgpt said the second one, and topics and services are actually component of the graph… i don't know if every component is an interface, nodes should be i guess... in internet i found this: ROS applications typically communicate through interfaces of one of three types: messages, services and actions)

**- What are ROS parameters? How can they be used?**

A parameter is a configuration value of a node. They're like the node setting (coil be boolean, integers etc.)

**- What is a ROS action composed of? Describe its main components and how they are used.**

Actions consist of three parts: a goal, feedback, and a result: An "action client" node sends a goal to an "action server" node that acknowledges the goal and returns a stream of feedback and a result.

**- How would you program a drone using ROS? Describe its main components using the ROS graph.Thess**

## How would you structure a ROS drone like this?

Nodes:

- A node for the main body.
- One node for each motor (for a total of four).
- A node for the camera.

Topics:

- One topic where motors are subscribed, where messages to move them can be published.
- One topic for the camera to publish the image feed.
- Battery-related topic: where the status of the battery is published (the battery itself may be another node).

Services:

- Take-off service. Which allows the drone to lift from the ground.
- Land service. Which does the opposite.

More?
There is no "right" solution, it's all about implementation.

Never did a graph, u can find a stilized version in the lab slides

**- How would you program a ground rover using ROS? Describe its main components using the ROS graph.**

same

**- What is an SDF file? Briefly describe how it is made and what is used for.**

Gazebo generates 3D world by reading from Simulation Description Format (SDF) files.

SDF is an XML format which is used to describe objects and environments for robot simulations and visualizations.

In SDF, every model is a group of links, which can be then connected together using joints.

Inside the link, we define multiple properties:

-First of all, the visual properties, using the visual tag

-We have then our collision tag, which defines the colliding properties of this link.

-We then define the intertia properties of the link, using the inertial tag.

**- How do ROS and Gazebo work together? Describe their main use and how they can communicate with each other.**

We can bridge them togheter with the ros_gz_bridge application. Using this application we can control the gazebo enviroment with ros commands

**- What is ros_gz_bridge? Describe its main uses by also providing some examples.**

Same answer as before i guess, don't have examples…