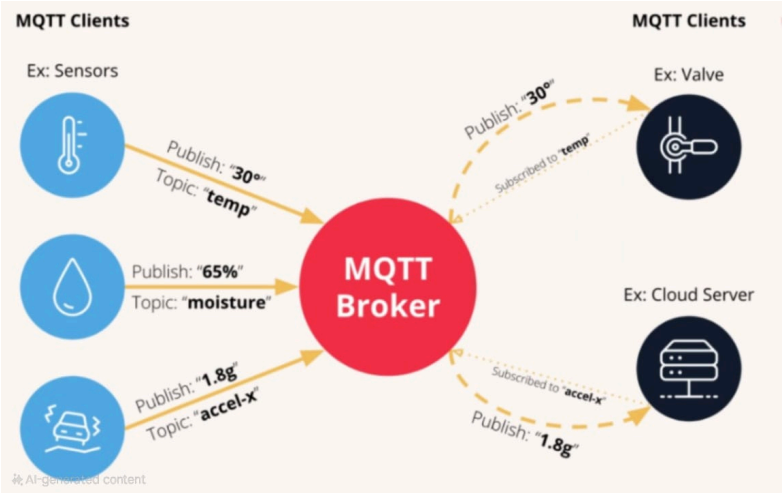
MODULE 5 - IOE

IoT protocols define **how devices communicate**, **exchange data,** and ensure **reliability, security, and efficiency**. They operate at different layers:

- **Application Layer:** MQTT, XMPP, DDS, AMQP, CoAP, REST
    - **MQTT**: Message Queuing Telemetry Transport
    - **AMQP**: Advanced Message Queuing Protocol
    - **CoAP**: Constrained Application Protocol
    - **DDS**: Data Distribution Service
    - **XMPP**: Extensible Messaging and Presence Protocol
    - **REST**: Representational State Transfer
- **Network/Adaptation Layer:** IPv6, 6LoWPAN
- **Physical/Link Layer (Wireless Communication):** e.g., Wi-Fi, Zigbee, BLE (Bluetooth Low Energy) , LoRaWAN (Long Range Wide Area Network)

A] MQTT - Message Queuing Telemetry Transport

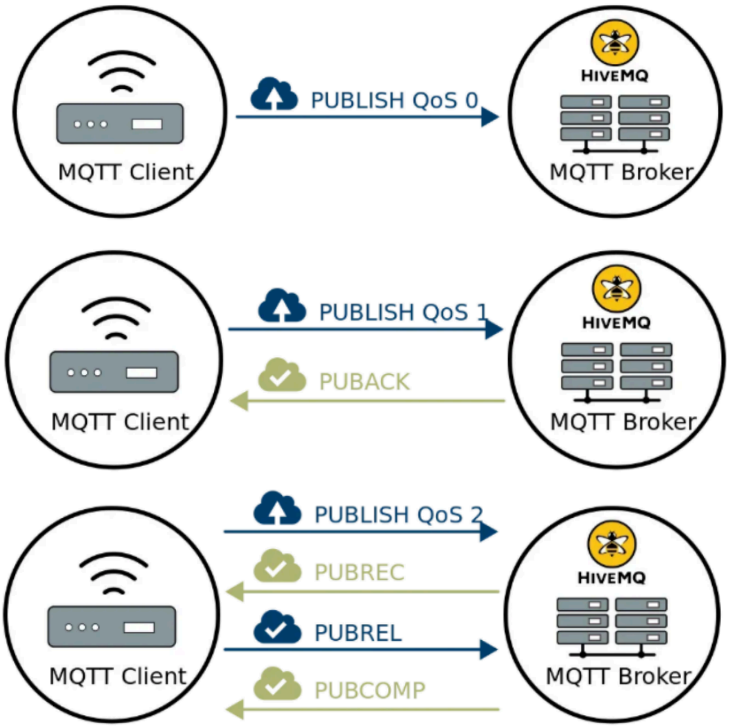


- **Overview:** A **lightweight, publish/subscribe** messaging protocol designed for **low-bandwidth and high-latency networks.**
- **How it works:**
    - **Publishers** send messages to a **broker** on a particular topic.
    - **Broker:**
        - Central communication hub. Receives messages from publishers. Sends them to the correct subscribers.
        - If a subscriber disconnects, the broker can **buffer messages** and send them when it reconnects.

        - If a publisher disconnects unexpectedly, the broker can send its **Last Will and Testament (LWT)** message.
    - **Subscribers** receive the messages from the broker if they have subscribed to that topic.

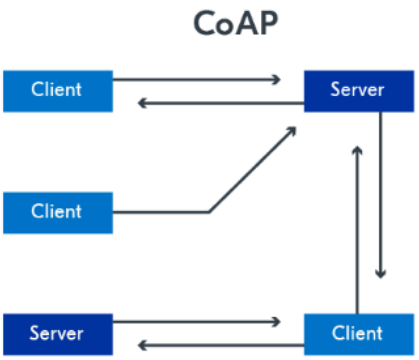
- **Features:**
    - Runs over **TCP/IP**
    - Small packet size (2-byte header)
    - It uses a **connection-oriented** approach.

QoS	Model	Reliability	Description
0	Fire and Forget	Unreliable	Message is delivered without duplication; no acknowledgment is required.
1	At Least Once	Reliable	Message is delivered at least once with possible duplications; acknowledgment is required (PUBACK).
2	Exactly Once	Reliable without duplications	Message is delivered exactly once by using a four-step handshake (PUBLISH, PUBREC, PUBREL, PUBCOMP) without duplication.



- **Advantages:**
  ✅ Lightweight
  ✅ Reliable communication with QoS
  ✅ Supports thousands of devices.
- ✅ Fast delivery and Bi-directional communication.
- **Disadvantages:**
  ❌ Needs a central broker
  ❌ TCP overhead (not best for very low-power devices)
- **Applications:** Smart homes (temperature sensors, smart lights) , Connected vehicles

B] CoAP - Constrained Application Protocol



- **Overview:**
  A web transfer protocol designed for **low-power** and **constrained devices**.
- **How it works:**
  - Similar to HTTP (RESTful), but runs on **UDP** instead of TCP.
  - Uses methods like **GET, POST, PUT, and DELETE**.
    - GET: Read a resource
    - POST: Create a new sub-resource
    - PUT: Create or update a resource
    - DELETE: Remove a resource
  - Supports **Multicast** and **Asynchronous communication**.
  - Uses a **4-byte header - Lightweight**

Reliability Mechanism:

1. **Confirmable Messages (CON)**
   - Reliable messages.
   - Client keeps retransmitting the CON message until it gets an ACK with the same message ID.
   - If the server cannot process the message, it sends RST (reset) instead of ACK**.**
2. **Non-Confirmable Messages (NON)**
   - Unreliable messages used for non-critical data.
   - No acknowledgment is sent by the server.

- **Advantages:**
  ✅ Low power consumption
  ✅ Easy to integrate with REST/HTTP systems

- **Disadvantages:**
  ❌ Less reliable than TCP-based protocols
  ❌ Security is limited without DTLS
- **Applications:**
  - Smart energy grids

1️⃣ CoAP vs MQTT — Quick Comparison

Feature	CoAP	MQTT
Transport	**UDP**	**TCP**
Model	Request–Response (REST)	Publish–Subscribe
Reliability	CON / NON	QoS 0/1/2
Overhead	Very low	Low
Best For	Simple IoT devices, sensors	Real-time messaging, many subscribers

C] AMQP: Advanced Message Queuing Protocol (jitna yad rahe utna)

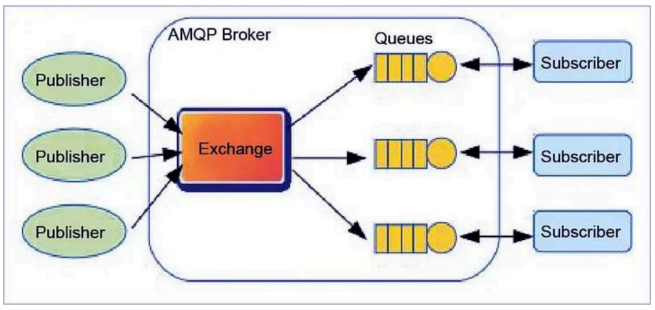
- AMQP is a **message queuing protocol**, similar to MQTT.
- Communication happens through **message queues**.
- Supports **asynchronous communication** between publishers and subscribers in **IoT** and **enterprise systems**.

Features

- Works as a **middleware messaging layer**.
- Provides **reliable message delivery** with guarantees:
  - **At most once** (unreliable)
  - **At least once** (reliable)
  - **Exactly once** (reliable and no duplicates)

Components of the AMQP Model

AMQP (Advanced Message Queuing Protocol)



1. **Exchange:** Receives messages from publishers. Routes them to correct message queues.
2. **Message Queue:** Temporarily stores messages until consumed by subscribers.
3. **Binding:** Defines the **connection/link** between an exchange and a queue.

How AMQP Works

- The **producer** creates a message and sends it to the **broker**.
- The broker delivers it to an **exchange**.
- The exchange uses **bindings** to decide which **queue** the message goes to.
- A **consumer** retrieves the message from the queue and processes it.

Advantages (already mentioned in the above ans, adv are same)

- Highly **reliable**, **secure**, and **interoperable**.
- Suitable for both **IoT systems** and **Enterprise/business messaging**.
- **Open source.**

Disadvantages: More complex to implement than MQTT.
Applications:

- Banking and financial IoT systems
- Smart cities

D] DDS: Data Distribution Service (jitna yad rahe utna)

A **middleware** protocol for **real-time, peer-to-peer communication** without requiring a broker.

- **Publisher/Subscriber Model (**Similar to MQTT but with a major difference:)
  - **DDS does NOT use a broker**.
  - Devices (nodes) communicate **directly** → **lower latency, faster communication.**

Global Data Space (GDS)

- Acts like a **virtual shared memory** accessible to all connected nodes.

- Not physical; it is a **concept** that joins data from all devices into one shared “space”.

Working

- Every topic has a **QoS contract** defining how data should be delivered.
- Publisher declares **topic + QoS policy** when sending data.
- Subscriber declares **topic** and receives data based on its QoS rules.

Two Important DDS Layers

A) DCPS – Data-Centric Publish-Subscribe Layer

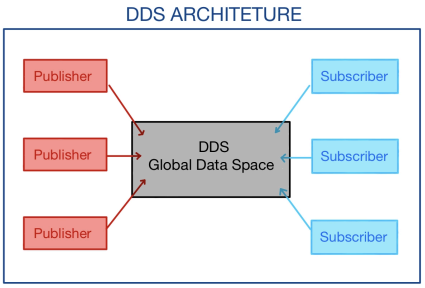
- Main layer that handles **real-time data exchange**.
- Manages: Topics and Publishers & and Subscribers

B) DLRL – Data Local Reconstruction Layer

- Object-oriented abstraction of DCPS data.

Advantages (already repeated above): 1. No central broker; 2. Real-time and peer-to-peer communication

Disadvantages: 1. Complex to implement



IoT Protocols Comparison Table

Feature	MQTT	CoAP	HTTP/HTTPS	AMQP	DDS
Communication Model	Publish–Subscribe	Request–Response	Request–Response	Message Queue	Publish–Subscribe
Transport Layer	TCP	UDP	TCP	TCP	TCP / UDP
Bandwidth Usage	Very Low	Very Low	High	Moderate–High	Moderate
Latency	Low	Very Low	High	Moderate	Very Low (Real-time)
Reliability	Medium (QoS 0–2 levels)	Low (best effort)	High	Very High	Very High
Power Consumption	Low	Very Low	High	Moderate	Moderate–High
Security	TLS/SSL	DTLS	TLS/SSL	TLS/SSL	Built-in Security QoS
Scalability	High	High	Low	Moderate	Very High
Message Size	Small	Small	Large	Large	Variable
Best For	Remote sensors, home automation	Constrained IoT devices	Web-connected IoT systems	Industrial IoT, financial apps	Autonomous systems, robotics
Example Use Case	Smart home devices, wearables	Smart lighting, meters	Cloud dashboards	Banking, logistics IoT	Self-driving vehicles, drones

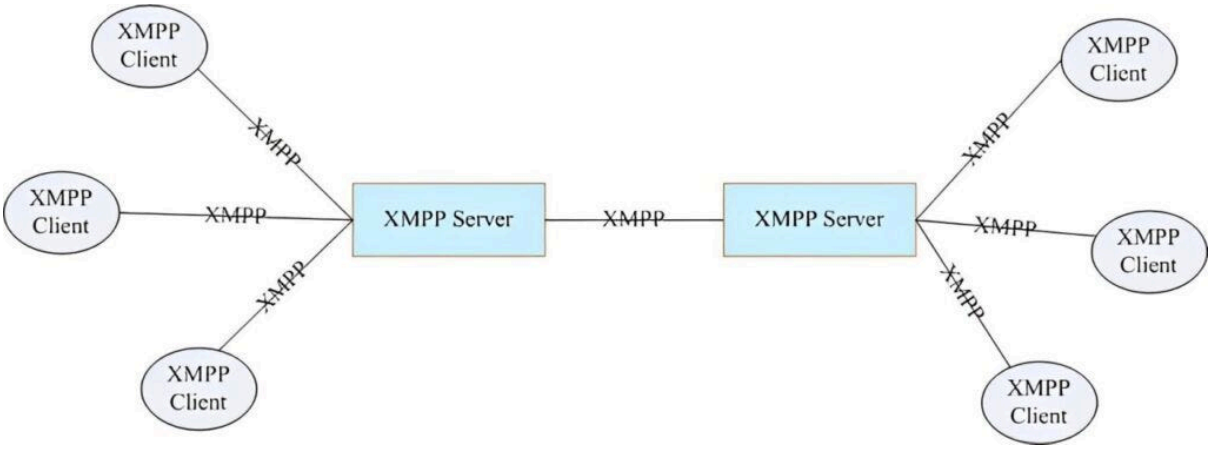
(DO ONLY THE POINTS THAT YOU CAN - MINIMUM 5 TO 7)

E] XMPP: Extensible Messaging and Presence Protocol

XML-based protocol for IoT communication, **one-to-one chats**, **voice/video calls**, and **group chats.**

- **How it works:**
  - Uses **client-server** model.
  - Devices send structured XML messages.
  - Messages are delivered even if the recipient is offline.
- **Features:**
  - Runs over **TCP**
  - One long-lived connection; no need to reconnect for every message.

Advantages:
✅ Open standard, widely supported
✅ Works with human + device communication
Disadvantages:
❌ XML overhead (heavy for constrained devices)
❌ More complex than MQTT/CoAP
Applications:
- Smart homes (chatbots, notifications)
- Healthcare communication

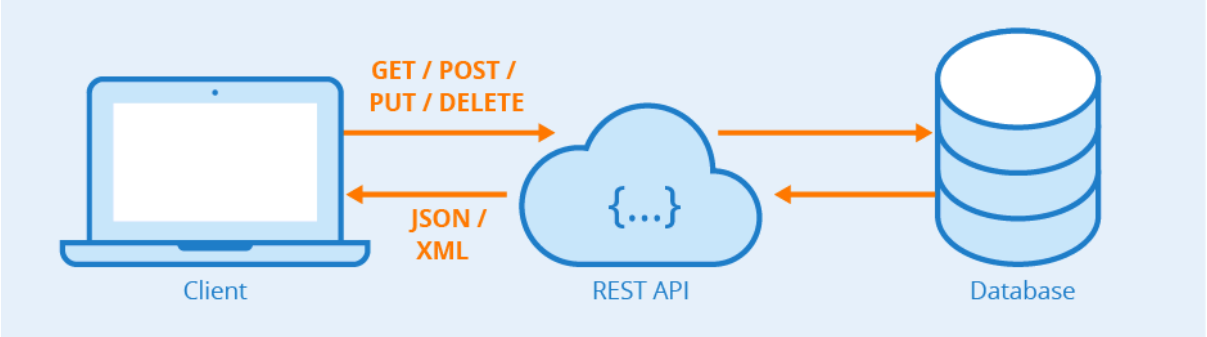


REST: Representational State Transfer

REST (Representational State Transfer)

1. **Definition** – REST is an **architectural style** used for designing **web services** that communicate over HTTP.
2. **Resource-Based** – Everything (sensor, device, or data) is treated as a **resource** identified by a **URL**.
3. **Operations** – Uses standard **HTTP methods**:
   - `GET` – Read data
   - `POST` – Create data
   - `PUT` – Update data
   - `DELETE` – Remove data
4. **Stateless** – Each client request contains all necessary information; the server doesn't store session data.
5. **Lightweight** – Ideal for **IoT** due to low bandwidth usage and easy integration with **web applications**.
6. **Example** – A temperature sensor sending data to a cloud via `POST /temperature`.

- **Advantages:**
  ✅ Simple, standardized
  ✅ Human-readable (JSON/XML)
- **Disadvantages:**
  ❌ High overhead (HTTP headers)
  ❌ Not suitable for ultra-low-power devices



IPv6 (Internet Protocol version 6)

- **Overview:**
  Next-generation Internet Protocol designed to replace IPv4 and support massive IoT devices.
- **Features:**
  - Provides unique **128-bit addresses** to devices.
  - Enables direct **device-to-device** communication.
  - Large address space (**3.4×10^{38} addresses**)
  - Built-in security with **IPsec**

- **Advantages:**
  ✅ Supports billions of IoT devices
  ✅ Eliminates NAT (Network Address Translation)
  ✅ Better routing efficiency
- **Disadvantages:**
  ❌ Migration from IPv4 is slow
  ❌ Higher header size → more bandwidth needed
- **Applications:**
  - Smart cities
  - Direct device-to-device communication

6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks)

- A **communication protocol** that allows **IPv6** to run over **low-power wireless networks**.
- IPv6 header is large → 6LoWPAN **compresses it**.
- Typically works on **IEEE 802.15.4** (used in Zigbee).
- Designed for **low-cost**, **low-power IoT devices** that send small packets of data.
- IoT devices using 6LoWPAN can connect **directly to the Internet** using **IPv6**.

Mesh Networking

- Supports **multi-hop communication**.
- Means a message can pass through multiple nodes → **larger coverage**.
- Smart home sensors (temperature, motion, lighting) sending data to cloud through a **6LoWPAN gateway**.

IoT Routing Protocols

- IoT routing protocols ensure **efficient data transmission** between **sensors** → **gateways,** → **cloud**.
- IoT devices (Wireless sensor network - WSN nodes) have **low power**, **small memory**, **short range**, so routing must be:
  ✔ Energy-efficient
  ✔ Reliable
  ✔ Scalable
  ✔ Adaptive

IoT routing protocols are mainly of **two types**:

1️⃣ **Data-Centric Routing Protocols**

2️⃣ **Flat-Architecture Routing Protocols**

Data-Centric Routing Protocols

- **Concept**
  - In data-centric routing, communication is based on **data attributes (what)** rather than **node identity (who)**.
  - The idea is to **avoid redundant data transmission** and **reduce energy consumption.**
- **How It Works**
  - The sink (base station) sends a query specifying data attributes.
  - Sensor nodes that have the matching data respond.
  - Nodes may aggregate or fuse data to minimize transmission cost.

Example: SPIN (Sensor Protocols for Information via Negotiation) - Uses metadata negotiation before sending actual data

Example 1: Smart Agriculture System

Scenario:
In a large smart farm, hundreds of soil and temperature sensors are deployed across fields.

How Data-Centric Routing Works:

- The **central controller (sink)** sends a query:
  "Send me temperature readings above 35°C and soil moisture below 20%."
- Only those **sensor nodes** with matching conditions respond.
- Nearby nodes **aggregate data** (e.g., average temperature of the zone) before sending to the controller.
- This avoids sending duplicate data from every sensor.

Benefits:

- Saves energy (only relevant data is transmitted).
- Reduces network traffic.
- Enables real-time irrigation decisions.

Flat-Architecture Routing Protocols

Flat-Architecture Routing Protocols

In **flat architecture**, all sensor nodes are **equal in role and responsibility** — there are **no hierarchical levels** (like cluster heads or gateways).

Each node cooperates in sensing, processing, and forwarding data.

Characteristics:

- **Peer-to-peer communication** among nodes.
- **No fixed hierarchy** (every node can act as source or relay).
- Suitable for **small to medium-sized** networks.
- Focus on **energy balance** among nodes to prolong network life.
- Often combined with **data-centric** approaches.

- **Advantages:**
  - Simple design and easy to deploy.
  - Load distribution across nodes.

- **Disadvantages:**
  - Not ideal for large-scale IoT networks.
  - Harder to manage as the network size grows.

🏠 Example: Smart Home IoT Network

Scenario: Devices like smart lights, sensors, locks connected using Zigbee/Thread mesh.

How Flat Architecture Works:

- Every device is equal → no central node.
- Data (e.g., motion alert) hops from one device to another until it reaches the gateway.
- If one route fails, another path is chosen (**self-healing mesh**).

Flooding – IoT Routing Technique

Flooding is the **simplest data dissemination technique** in IoT, where **every node broadcasts a received packet to all its neighbors**, except the one it got it from.

How Flooding Works

1. A source node generates a packet.
2. It broadcasts the packet to all neighbors.
3. Each receiving node checks:

    - **If the packet is new** → re-broadcast it
    - **If duplicate** → discard it
4. Process continues until:
    - Packet reaches the **sink**, OR
    - **TTL (**Time to Live**) expires**

Example: Smart Forest Monitoring

- Node A detects fire → broadcasts alert.
- Nodes B, C, and D re-broadcast to their neighbors.
- Alert reaches the base station through **multiple paths**.

Used in:

- Emergency alerts (fire, gas leak)

Advantages

- Very **simple** → no routing tables needed.
- Works even if nodes **move** or **fail**.

Disadvantages

- Many **duplicate packets** (redundancy).
- **High energy** consumption (bad for IoT).
- **Network congestion** → nodes get overwhelmed by too many messages.

Gossiping – IoT Routing Technique

Gossiping is a **controlled version of flooding**.
Instead of sending to all neighbors, a node forwards the packet to **one random neighbor**.

How Gossiping Works

1. Node sends a packet to **one random neighbor**.
2. That neighbor forwards to another random node.
3. Continues until the message spreads through the network.

Example: Smart Home Network

- The motion sensor sends a message to **one light node**.
- Light sends to the **door lock**.
- The message slowly propagates to all nodes.

Used in:

- Smart grids (load balancing updates)

- Peer-to-peer IoT/blockchain
- Smart city distributed networks

Advantages (any 2/3)

- Fewer duplicates → **low redundancy**
- **Low energy** consumption
- Avoids congestion
- Simple — no routing tables

Disadvantages

- **Slower** message spread
- Possible **message loss** (if a node drops out)

🔥 Flooding vs Gossiping (Exam Table)

Feature	Flooding	Gossiping
Forwarding	To all neighbors	To one random neighbor
Speed	Very fast	Slower
Redundancy	High	Low
Energy Use	High	Low
Reliability	High (multiple paths)	Medium (can lose data)
Best For	Emergency alerts, device discovery	Periodic updates, P2P systems

SPIN AND ITS TYPES (sab almost similar, sab ka protocol working and example scenario is very similar way of writing.)

SPIN (Sensor Protocols for Information via Negotiation)

- A **data-centric, adaptive protocol** for **energy-efficient data dissemination** in wireless sensor networks.
- **Goal:** The **main goal** of SPIN is to eliminate **redundant data transmission** and conserve energy.
- **Developed by:** Heinzelman (1999)
- **Architecture:** Data-Centric, Flat

Key Concepts of SPIN

1. Meta-Data Negotiation:

- Instead of sending raw data blindly, nodes first exchange metadata (a short descriptor of the data).
- This helps avoid redundant transmission of the same data to multiple neighbors.

2. Resource Awareness:

- Nodes decide participation based on **available energy**.

3. Localized Algorithms:

- Each node makes decisions **independently**, without needing a global network topology.
- This makes SPIN scalable and easy to implement.

4. Data Advertisement Model:

- Nodes **advertise** available data.
- Neighbors **request only if needed**. This reduces unnecessary flooding.

SPIN Protocol Working (can write in your own words)

SPIN works in **three main stages:**

1. ADV (Advertisement):

- When a node gets new data, it **broadcasts an ADV message** (metadata) to nearby nodes.
- Example: Node A → ADV → Node B

2. REQ (Request):

- If a neighbor **needs that data**, it sends a **REQ message** asking for it.
- Example: Node B → REQ → Node A

3. DATA (Data Transmission):

- The sender then transmits the **actual DATA packet**.
- Example: Node A → DATA → Node B

Example Scenario

- Node A gets new temperature data → sends **ADV** to B & C.
- Node B doesn't have it → sends **REQ** → gets **DATA**.
- Node C already has data → ignores ADV → **energy saved**.
- Prevents flooding and saves energy by sending data **only when needed**.

Advantages of SPIN

(Write the same answer as Key Concepts written above for advantages -> Meta Data Negotiation, Localized Algorithm, Data Advertisement Model, and Resource Awareness)

Limitations of SPIN

❌ **Incomplete Data Spread:** If an intermediate node isn't interested, data may not reach others.
❌ **Not for Real-Time Use:** Negotiation may cause delay or data loss.
❌ **No Global View:** Lacks full network knowledge;

Applications of SPIN : Environmental monitoring (temperature, humidity, pollution), smart agriculture systems, energy efficient systems.

SPIN-PP (Point-to-Point) (Jo jo yaad aya likh dena)

Introduction:

- **SPIN-PP** is the **unicast version** of SPIN.
- Uses **point-to-point** communication instead of broadcast.
- Best for **sparse networks** or when broadcasting is **costly or unreliable**.

Key Idea

- The **ADV–REQ–DATA** handshake is still used, but messages are sent to one neighbor at a time.
- This approach helps to **avoid collisions**, **reduce contention**, and **save bandwidth**.

SPIN-PP Protocol Working

SPIN-PP works just like the original SPIN protocol, but uses **unicast** (one-to-one) communication instead of broadcast.

1. ADV (Advertisement):

- Nodes with new data send an **ADV message** to **one neighbor** (unicast).
- The neighbor is chosen using a **neighbor table or routing info**.
  👉 *Node A → ADV → Node B*

2. REQ (Request):

- If Node B **needs the data**, it sends a **REQ message** back to Node A.
  👉 *Node B → REQ → Node A*

3. DATA (Transmission):

- Node A sends the **actual DATA packet** to Node B.
  👉 *Node A → DATA → Node B*

💡 **Flow:** ADV → REQ → DATA (all unicast)

Example Scenario

- **Node A** checks its neighbor table and selects **Node B** as the next hop.
- A sends **ADV** to B (unicast).

- If B doesn't have this data, it sends **REQ** back.
- A then transmits the **DATA** to B.
- B then repeats the process with its next neighbor, say **Node C**.
- This ensures the data is **propagated node by node**, avoiding the flooding that occurs in basic SPIN.

Advantages of SPIN-PP

✅ **Less Collisions & Contention** - Uses Unicast
✅ Saves Bandwidth.

Avoids flooding

Limitations:

❌ **Slower** data spread (one node at a time)
❌ Needs **neighbor knowledge**
❌ Not ideal for **dense networks**

Use Case:

- Sparse sensor networks
- Scenarios where broadcasting is **costly or unreliable.**

SPIN-EC (Energy-Conserve)

Introduction:

- **SPIN-EC** is an **energy-efficient** version of SPIN.
- Nodes decide whether to **participate in communication** based on their **remaining energy**.

Key Idea:

- Each node has an **energy threshold**.
- Only nodes with **energy > threshold** take part in ADV–REQ–DATA.
- Low-energy nodes **skip communication** to **save battery** and maintain network connectivity.

SPIN-EC Working Principle

SPIN-EC follows the same **ADV** → **REQ** → **DATA** process as SPIN,
but adds **energy checks** before each step to save power.

1. ADV (Advertisement):

- Node sends ADV **only if** its energy **> threshold**.

- If energy is low → **no advertisement** (saves power).
  💡 *If Energy(node) > Threshold → Send ADV; else skip.*

2. REQ (Request):

- Neighbor checks its energy before replying.

- Sends REQ only if **energy > threshold**, else **ignores ADV**.
  💡 *If Energy(neighbor) > Threshold → Send REQ; else ignore.*

3. DATA (Data Transmission):

- Source sends DATA **only if it has enough energy** left.
- If energy drops below threshold → **cancel transmission**.
  💡 *If Energy(source) > Threshold → Send DATA; else cancel.*

Example Scenario

Consider **Node A** has sensed a new temperature value:

1. **Node A** checks its energy. If it has enough, it sends an ADV.
2. **Neighbor Node B** receives ADV. If B has enough energy, it sends **REQ**. If B's battery is low, it ignores the ADV.
3. **Node A** receives REQ and checks energy again. If still enough, it sends **DATA** to B.
4. Nodes with **low residual energy** do not take part, thus **avoiding premature energy depletion**.

Advantages: (write your own advantages from ans, it's okay. Whatever you remember)

- Saves energy by skipping low-power nodes
- Saves battery.
- Improves network life.

Limitations:
❌ Difficult to select correct energy threshold
❌ Slightly more complex than basic SPIN

Use Case

- **Environmental monitoring** where data is periodic, and nodes can sleep if energy is low

SPIN-BC (Broadcast)

Introduction:

- **SPIN-BC** is the **broadcast version** of SPIN.
- Designed for **broadcast communication networks** (like radio).
- A single message reaches **multiple neighbors at once**.

Working:

1. Node advertises data once; all neighbors **receive the ADV simultaneously**.
2. Interested neighbors send REQ messages.
3. Data is then broadcast to all interested neighbors in a **single DATA transmission**.

💡 **Flow:** One ADV → Multiple REQs → One DATA broadcast

Example:

- Node A advertises metadata using **broadcast**.
- Nodes B, C, and D receive it simultaneously.
- B and C send REQ; D ignores.
- A **broadcasts the DATA once** → both B and C receive it.

SPIN-RL (Reliable)

Introduction:

- **SPIN-RL** is a **reliable version** of SPIN.
- Designed for **lossy or unstable networks** where data loss and link failures are common.
- Ensures data **eventually reaches the sink** even with interruptions.
- Keep track of which neighbors have received the data.

Working:

1. Follows normal **ADV** → **REQ** → **DATA** steps.
2. If no REQ is received after ADV → node **retransmits ADV** after some time.
3. Nodes **cache metadata** and may forward data later when new neighbors appear.

Example:

- Node A advertises data.
- Due to a temporary network failure, no neighbor responds.
- A **keeps re-advertising** after intervals.
- Eventually, Node B comes online and responds with REQ.
- Data reaches the sink reliably after multiple retries.

LEACH (Low-Energy Adaptive Clustering Hierarchy)

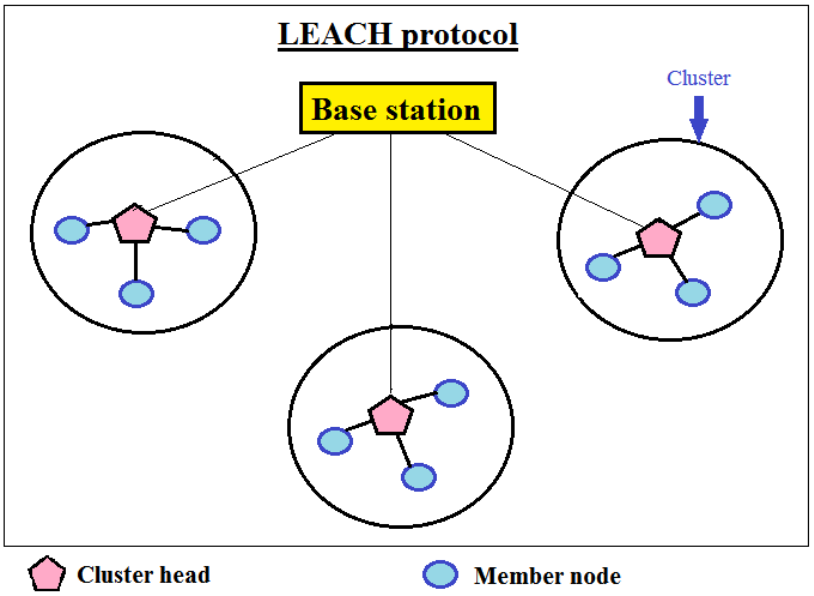
LEACH is a **hierarchical routing protocol** for WSNs aimed at **reducing energy consumption** by organizing the network into **clusters**.

- **Cluster Heads (CHs)** are elected periodically.
- CH role is **rotated** among nodes to balance energy usage.

LEACH Working Phases

1. **Set-up Phase**
    a. Nodes **self-elect** as cluster heads based on probability.
    b. CHs **broadcast** their status.
    c. Non-CH nodes **join the nearest CH**, forming clusters.
2. **Steady-State Phase**
    a. Member nodes **send data** to CH.
    b. CH **aggregates** and sends it to the **base station** (sink).
    c. After a round, new CHs are elected.

Example

In a network of 100 sensor nodes:

- 5 nodes are randomly selected as CHs.
- Each CH gathers data from its members (e.g., temperature readings).
- Instead of 100 nodes sending data to the sink, **only 5 CHs** send **aggregated data**.
- After one round, CH roles rotate to save energy.