**1. Introduction to IoT​**

This presentation focuses on the fundamentals of the **Internet of Things (IoT)**, covering its definition, components, and applications:

* **What is IoT?**: IoT is a system where computing, sensing, communication, and actuation take place. It's a network connecting humans, non-human objects, and cyber-objects for automation and decision-making.
* **Smart Objects**: IoT consists of "Smart Objects" or "Internet Connected Objects," which sense the environment, process data, self-configure, and exchange information with humans or other objects.
* **Examples of Smart Devices**: Examples include fitness trackers, smart watches, and smart footwear that help track health metrics like heart rate, sleep patterns, steps, posture, and calories burned.
* **IoT Applications**: IoT is applied in various domains such as smart cities, healthcare, transport, retail, safety, and security.
* **Key IoT Components**:
  + **Sensors**: Collect data from the environment or human activity.
  + **Actuators**: Execute actions based on the data received from sensors.
  + **Edge Computing**: Provides local computing power near the data source, reducing latency in critical workflows.
  + **Cloud Computing**: Larger-scale data storage and processing for IoT applications.
* **Challenges**: Issues such as security vulnerabilities, networking delays, and data governance are mentioned as barriers to IoT adoption in areas like healthcare.

**2. IoT Platforms and Devices​**

This document discusses the technological infrastructure required for IoT, with an emphasis on devices, processors, and platforms:

* **Main Components of IoT Devices**:
  + **Processors (ARM Cortex)**: Microcontrollers (MCU) and Microprocessor Units (MPU) for handling computations.
  + **Sensors/Actuators**: Key elements for gathering data and taking action.
  + **Memory**: For data storage and processing.
  + **Communication Devices**: Enable connectivity (e.g., Bluetooth, Wi-Fi).
* **Operating Systems for IoT**: Common OS for IoT devices include **FreeRTOS**, **Zephyr OS**, and **Mbed**, with support for lightweight communication protocols like MQTT, CoAP, and IPv6.
* **Prototyping Boards**:
  + **Arduino**: Open-source hardware used for prototyping.
  + **Raspberry Pi Pico**: A versatile microcontroller.
  + **ESP32**: A low-power dual-core processor with Wi-Fi and Bluetooth connectivity.
* **IoT Platforms**:
  + Popular cloud platforms for IoT include **Amazon Web Services (AWS)**, **Google Brillo**, **Apple HomeKit**, and **ARM mbed**.
* **Communication Technologies**: Technologies like **5G**, **LoRa**, **NB-IoT**, and **Bluetooth** are explored for IoT connectivity, varying in range, power consumption, and data transfer rates.
* **Edge Computing and AI**: The integration of AI with IoT, particularly through platforms like **NVIDIA Jetson Nano**, enables real-time processing closer to the data source, improving response times.

**3. IoT Protocols​**

This document elaborates on the communication protocols used in IoT, aligning them with the **ISO/OSI model**:

* **IoT Protocol Stack**: The IoT protocol stack follows the layers of the OSI model and includes various protocols for different layers (e.g., **CoAP, MQTT, HTTP** for the application layer; **TCP, UDP** for transport; **IPv4, IPv6, 6LoWPAN** for networking).
* **Communication Models**:
  + **Request-Response Model**: Stateless communication where the client sends requests, and the server responds (common in HTTP).
  + **Publish-Subscribe Model**: A model where data is published by devices and consumed by subscribers, often used in **MQTT**.
* **Wireless Standards**:
  + **Wi-Fi (IEEE 802.11ah)**: Common for high-data-rate applications.
  + **IEEE 802.15.4**: Low-power, low-data-rate standard for small, battery-operated devices (e.g., ZigBee).
  + **Bluetooth Low-Energy (BLE)**: For small data transfers with low power consumption.
* **Long-Range IoT Protocols**:
  + **LoRaWAN**: A protocol for long-range communication, commonly used in smart city applications.
  + **Narrowband IoT (NB-IoT)**: A 3GPP standard designed for low-power, wide-area IoT applications, focusing on indoor coverage, long battery life, and high connection density.

**4. File: IoT Protocols Overview**

This document provides a comprehensive overview of the key **IoT protocols**, breaking them down into messaging and communication protocols relevant to various IoT applications. Here's a detailed summary of its key points:

**IoT Protocols Hierarchy**

* **Protocols range** from cellular networks, NFC, Wi-Fi, to long-range technologies (LoRa, NB-IoT).
* Communication covers different distances from a few centimeters (NFC) to hundreds of kilometers (LoRa, cellular networks).

**Key IoT Messaging Protocols Explained**

1. **Advanced Message Queuing Protocol (AMQP)**
   * **Message-oriented protocol** used for reliable communication in middleware systems.
   * Architecture has three components:
     + **Exchange:** Receives and routes messages to queues.
     + **Message Queue:** Stores messages temporarily until a client retrieves them.
     + **Binding:** Manages the connection between the exchange and queue.
2. **Message Queue Telemetry Transport (MQTT)**
   * **Lightweight publish-subscribe protocol** for monitoring remote devices.
   * Works on **TCP** to provide reliability and supports communication between devices with limited memory and power.
   * **Components:**
     + **Publisher:** Sends data to a **Broker**.
     + **Broker:** Manages message delivery and security.
     + **Subscriber:** Receives messages from topics via the broker.
   * MQTT is **effective for low-bandwidth, power-constrained environments.**
3. **Data Distribution Service (DDS) Protocol**
   * Designed for real-time **publish-subscribe data exchange.**
   * Two layers:
     + **DCPS (Data-Centric Publish-Subscribe):** Manages data delivery.
     + **DLRL (Data-Local Reconstruction Layer):** Provides an interface to DCPS.
   * DDS is language and hardware independent.
4. **Extensible Messaging and Presence Protocol (XMPP)**
   * **Real-time messaging** protocol originally designed for instant messaging (IM).
   * Supports **encryption, access control,** and **multi-party chats**.
   * Works well for telepresence and real-time video/voice applications.
5. **Constrained Application Protocol (CoAP)**
   * Lightweight, **UDP-based RESTful protocol** designed for constrained devices and networks.
   * Supports **HTTP-like methods (GET, POST, PUT, DELETE)** but optimized for IoT.
   * Allows both **synchronous and asynchronous communication.**

**IoT Protocol Categorization**

* **Request-Response:**
  + **HTTP (synchronous)**
  + **CoAP (asynchronous)**
* **Subscription-Notification:**
  + **MQTT, AMQP, CoAP**
* **Streamed Communication:**
  + **XMPP**

**Summary**

The document emphasizes how IoT transforms raw data into actionable insights through a mix of protocols. Each protocol has specific use cases, often optimized for low-power or constrained environments, ensuring scalability across small devices to cloud infrastructure.

**5. File: MQTT vs CoAP**

This document compares **MQTT and CoAP**, two essential IoT protocols, providing their features, use cases, and architectural details.

**Overview of the Protocols**

* **MQTT:**
  + A **publish/subscribe protocol** designed for low-bandwidth, high-latency networks.
  + Uses **TCP** as the transport layer to ensure reliable message delivery.
  + Works with a **broker** to manage communication between multiple clients.
  + Supports **three Quality of Service (QoS) levels:**
    - **QoS 0:** At most once (no guarantee of delivery).
    - **QoS 1:** At least once (duplicates possible).
    - **QoS 2:** Exactly once (highest reliability).
* **CoAP:**
  + A **RESTful, request/response protocol** that uses **UDP** for communication.
  + Designed for **resource-constrained devices and networks**.
  + Supports both **confirmable (acknowledged)** and **non-confirmable** messages.
  + Ideal for scenarios where lightweight communication is essential, such as sensors or meters.

**Comparison: MQTT vs CoAP**

| **Aspect** | **MQTT** | **CoAP** |
| --- | --- | --- |
| **Transport** | TCP | UDP |
| **Model** | Publish/Subscribe | Request/Response |
| **Use Case** | Monitoring & Control | Device communication via HTTP |
| **Performance** | Excellent for IoT messaging | Excellent for constrained networks |
| **Example** | Enterprise messaging | Smart meters (gas/water) |

**Additional Notes**

* **MQTT-S:** An extension of MQTT for **Wireless Sensor Networks (WSN)**. Uses **UDP** for communication, supporting low-power nodes and dynamic broker discovery.
* **CoAP’s Observation Pattern:**
  + Allows asynchronous notifications based on resource changes.
  + Provides multicast support, enabling efficient communication with multiple devices.

**Summary**

* MQTT and CoAP **complement** each other in IoT ecosystems:
  + MQTT: Ideal for continuous data streams and command/control scenarios.
  + CoAP: Perfect for low-power devices and networks needing lightweight communication.
* The choice between protocols depends on **use case, network, and device constraints.**