Unit IV
IoT Protocols

Course Objective:

To learn the IoT protocols, cloud platforms and security issues in IoT

Points to cover

- Protocol Standardization for IoT
- M2M and WSN Protocols
- RFID Protocol
- Modbus Protocol
- Zigbee Architecture
- IP based Protocols:
  - MQTT (Secure)
  - 6LoWPAN
  - LoRa.
- #Exemplar/Case Studies :LoRa based Smart Irrigation System.

Standardizing the IoT

- Smart objects produce large volumes of data.
- This data needs to be managed, processed, transferred and stored securely.
- Standardization is the key to achieve universally accepted specifications and protocols for true interoperability between devices and applications.
- The use of standards:
  - ensures interoperable and cost-effective solutions
  - opens up opportunities in new areas
  - allows the market to reach its full potential

IoT Standardization Efforts

- The IoT- A (Internet of Things architecture) is targeting a **holistic(universal) architecture for all IoT sectors.**
- 17 European organizations from nine countries are a part of IoT- A.

Protocol Standardization for IoT

- IoT-Architecture is one of the few efforts targeting a holistic architecture for all IoT sectors
- This consortium consists of 17 European organizations from nine countries
- Summarized current status of IoT standardization as
    - Fragmented architectures
    - No holistic approach to implement IoT has yet been proposed
    - Many island solutions do exist (RFID, sensor nets, etc.)
    - Little cross-sector reuse of technology and exchange of knowledge

Current IoT Standardization is a problem, so….. What could be Done to Solve this???

Proposed Solution By IoT-A for Standardization

- Create Architectural foundation for IoT, that will be operable with future Internet
- Use Existing technologies instead of creating new ones.
- Demonstrating the applicability of IoT in a set of use cases
- Establish a strong stakeholder group to remove the barriers and accept IoT on wide scale
- Combine various IoT technologies into a single entity

Groups doing IoT Standardization

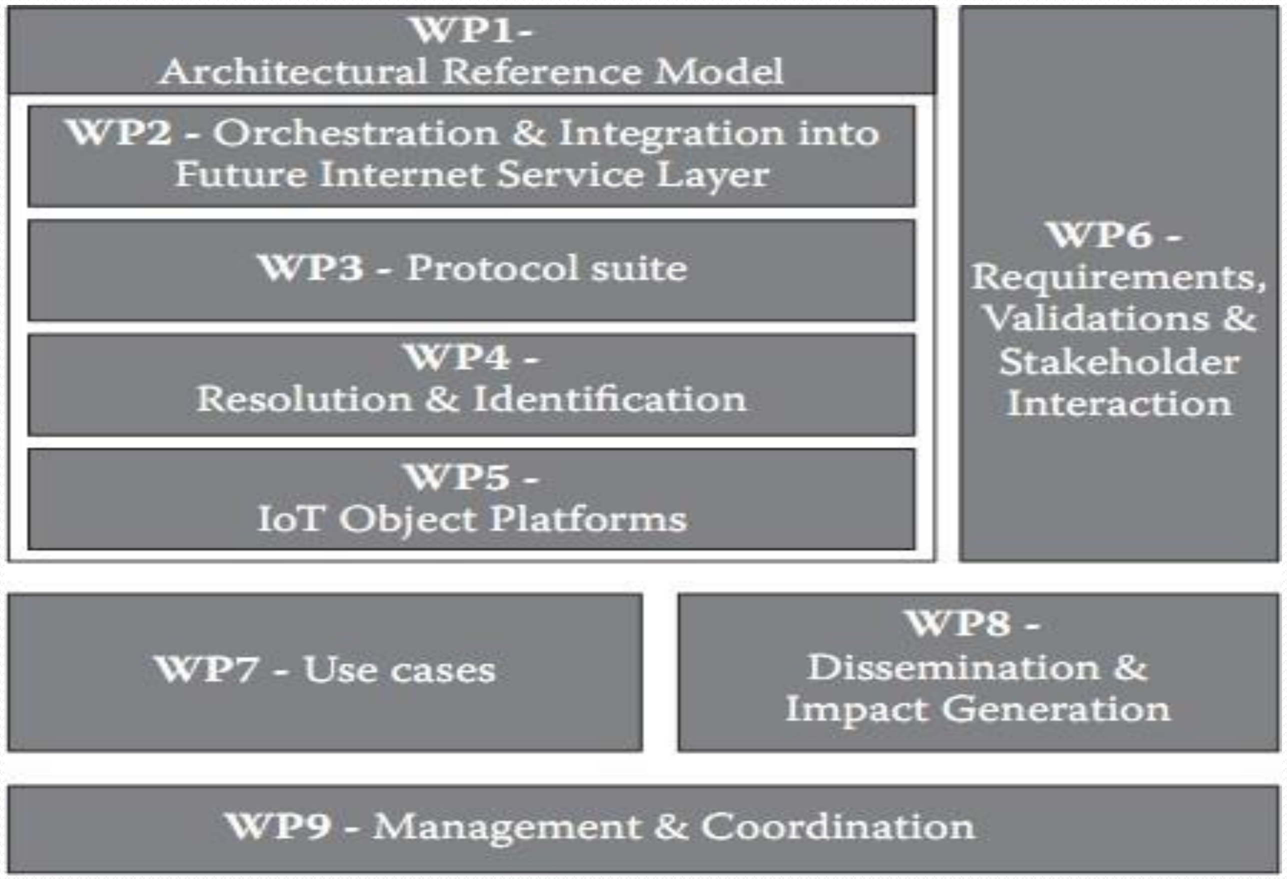
- Work Package Framework **(WPF)**
- International Telecommunication Union Standardization Sector**(ITU-T)**
- Internet Protocol for Smart Objects **(IPSO)**
    - Aim to form an open group of companies to market and educate about how to use IP for IoT smart objects based on an all- IP holistic approach

Work Package Framework

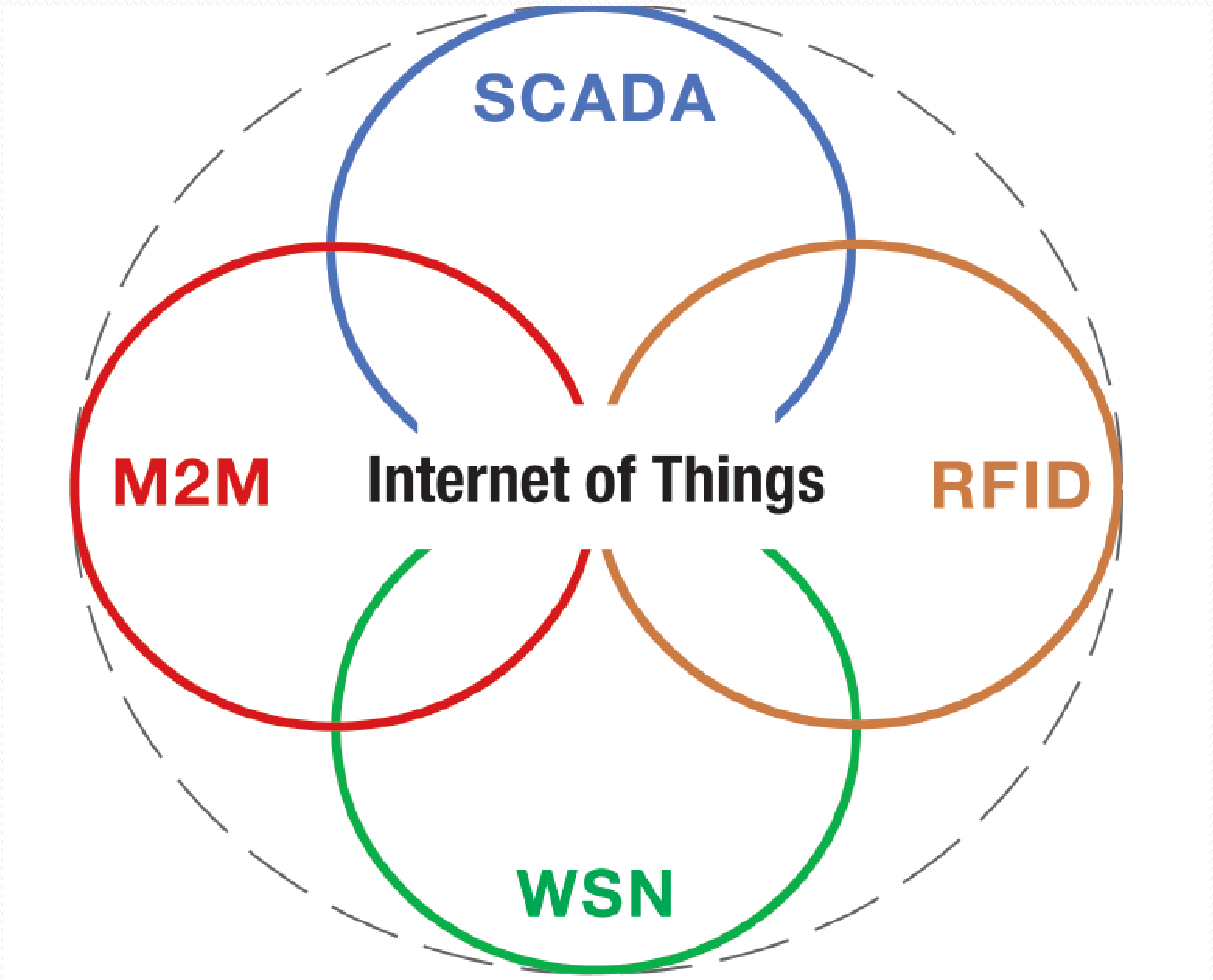
- An IoT framework is a set of tools, standards, and protocols that provide a structure for developing and deploying IoT applications and services.
- It typically includes hardware, software, networking elements, device management, security, data management, application development, and a cloud-based platform

Work Package Framework

- The WPF divides the implementation standards of IoT into hierarchical groups of tasks


WP1- Architectural Reference Model
WP2 - Orchestration & Integration into Future Internet Service Layer
WP3 - Protocol suite
WP4 - Resolution & Identification
WP5 - IoT Object Platforms
WP6 - Requirements, Validations & Stakeholder Interaction
WP7 - Use cases
WP8 - Dissemination & Impact Generation
WP9 - Management & Coordination

Four Pillars of IoT



M2M

- Machine to Machine
- Enables **flow of data between machines** which **monitors data by** means of **sensors** and at other end **extracts the information on gathered data and processes it**.
- Subset of IoT
- It uses WAN, GPRS, Cellular and Fixed N/w's

M2M Architecture

- Components of M2M architecture are :

1)M2M Devices

2)M2M Area Network i.e Device Domain

3)M2M Gateway

4)M2M Communcation N/w's : Network Domain

5)M2M Applications i.e Application Domain

M2M Architecture

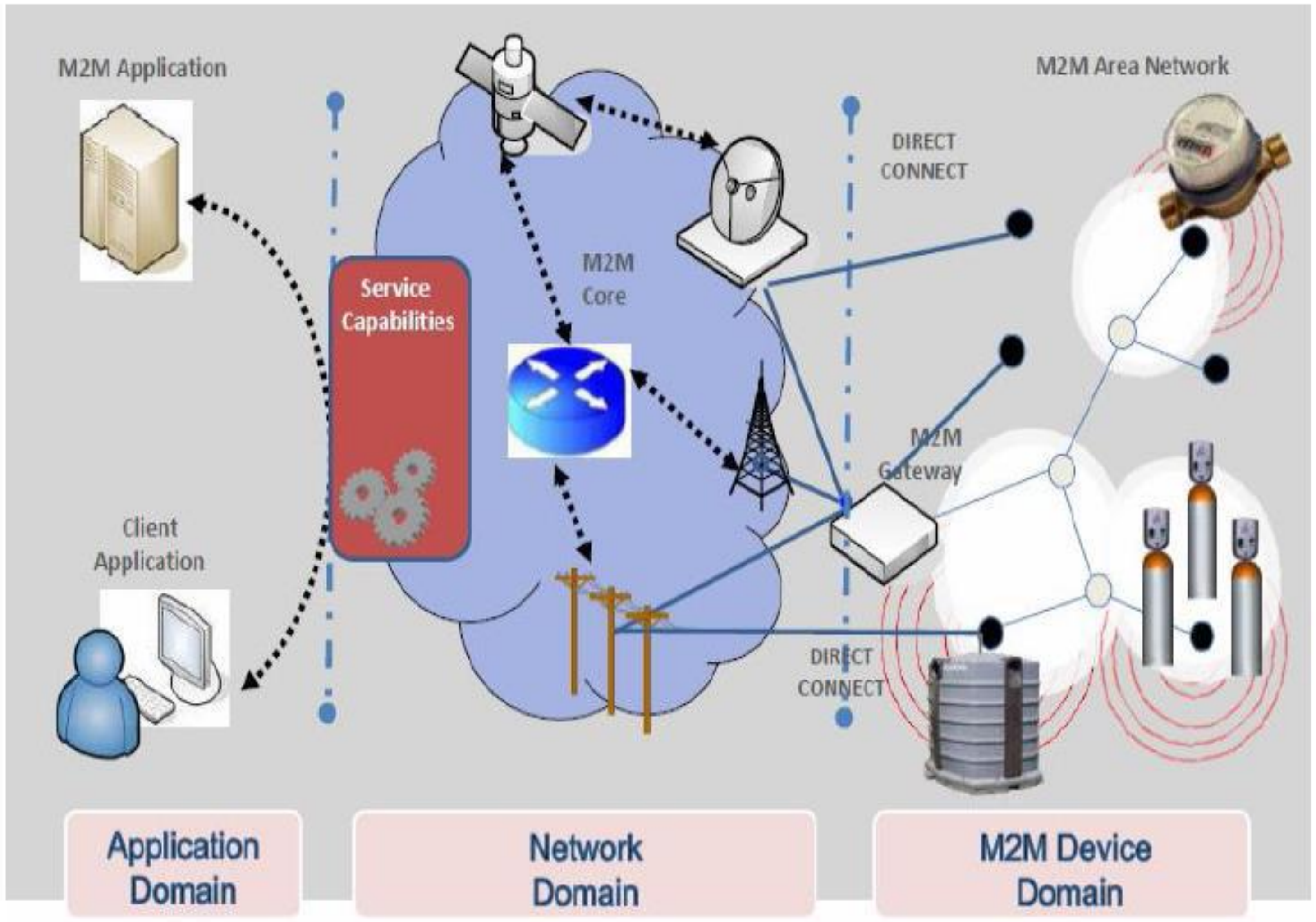


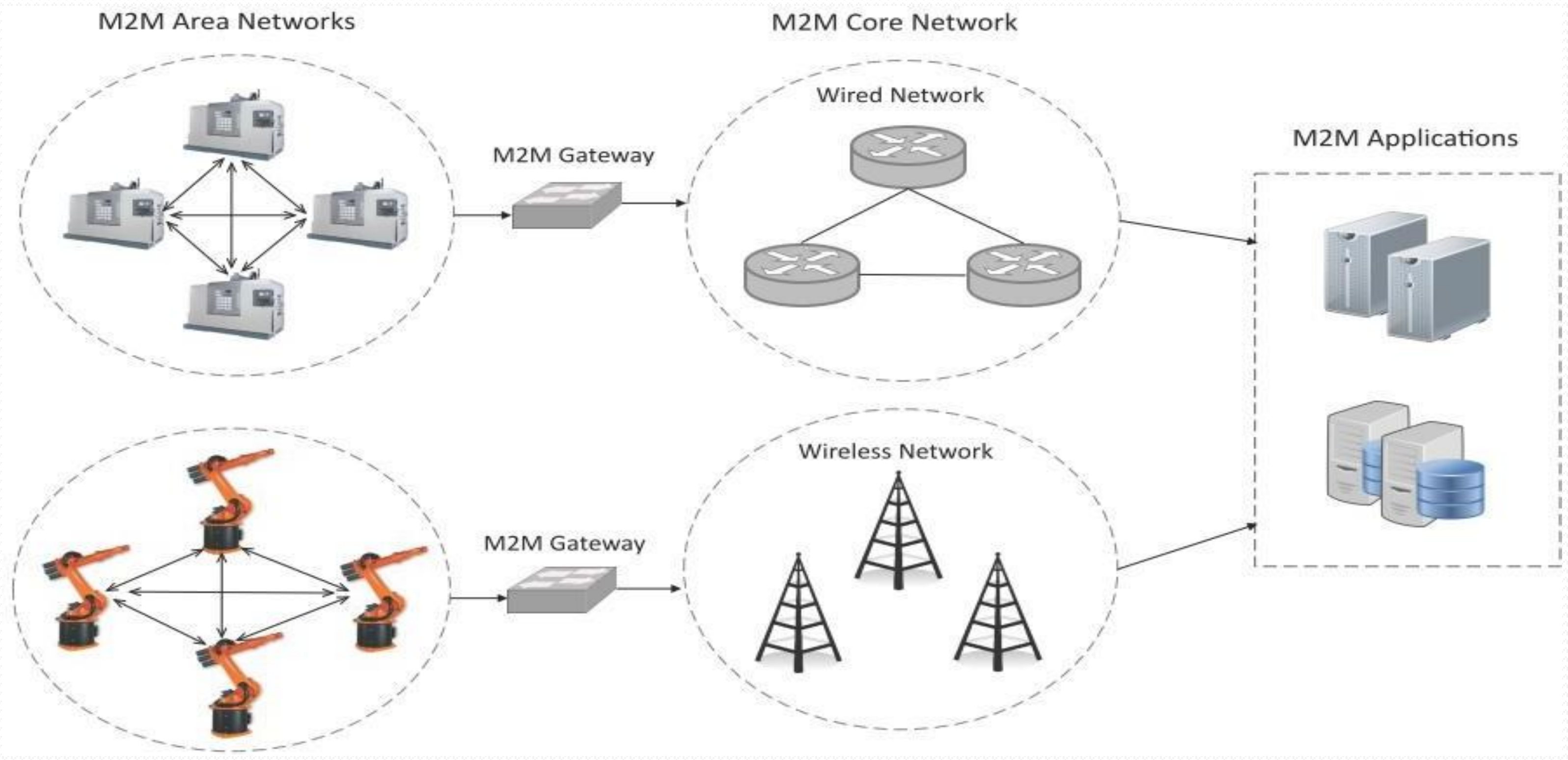
Figure 1: Architecture of M2M system

M2M Devices

- Device that are capable of replying to request for data contained within those devices or capable of transmitting data autonomously are M2M Devices.
- **Sensors and communication devices** are the endpoints of M2M applications.

Machine-to-Machine (M2M)

- Machine-to-Machine (M2M) refers to networking of machines (or devices) for the purpose of remote monitoring and control and data exchange.

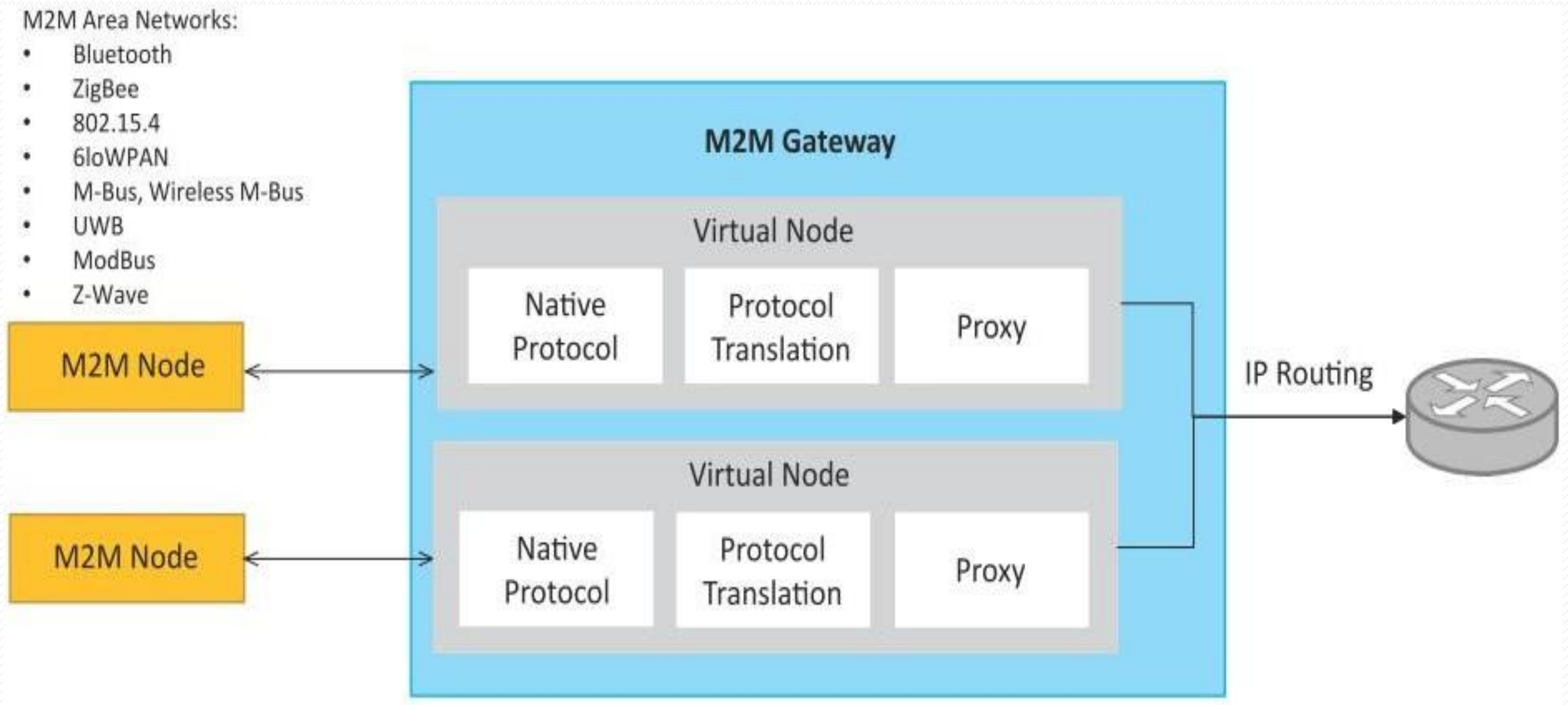


Machine-to-Machine (M2M)

- An M2M area network comprises of machines (or M2M nodes) which have embedded hardware modules for sensing, actuation and communication.
- Various communication protocols can be used for M2M local area networks such as ZigBee, Bluetooh, ModBus, M-Bus, Wirless M-Bus, Power Line Communication (PLC), 6LoWPAN, IEEE 802.15.4, etc.
- The communication network provides connectivity to remote M2M area networks.
- The communication network can use either wired or wireless networks (IP- based).
- While the M2M area networks use either proprietary or non-IP based communication protocols, the communication network uses IP-based networks.

M2M gateway

- Since non-IP based protocols are used within M2M area networks, the M2M nodes within one network cannot communicate with nodes in an external network.
- To enable the communication between remote M2M area networks, M2M gateways are used.



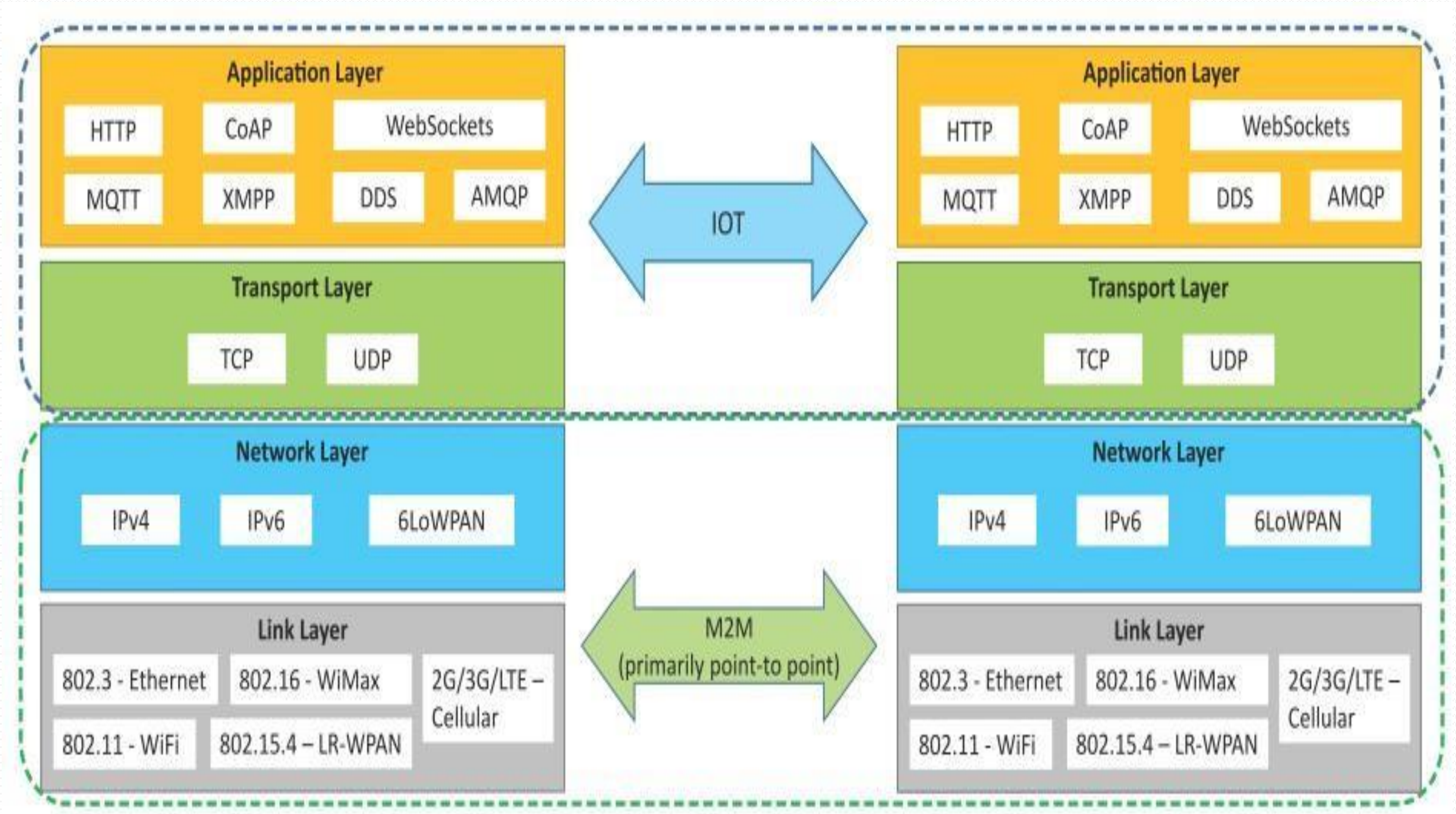
Difference between IoT and M2M

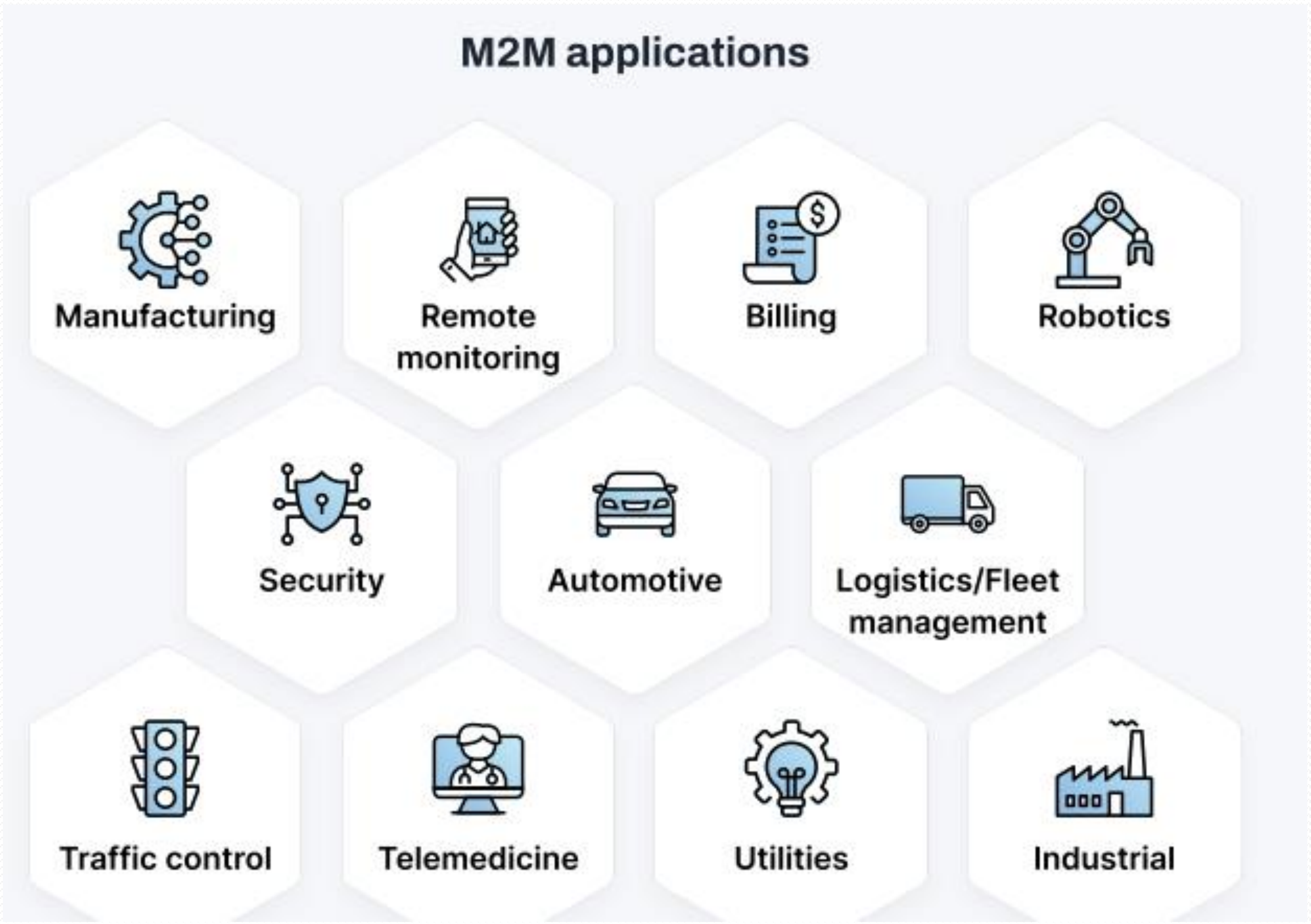
- Communication Protocols
  - M2M and IoT can differ in how the communication between the machines or devices happens.
  - M2M uses either proprietary or non-IP based communication protocols for communication within the M2M area networks.
- Machines in M2M vs Things in IoT
  - The "Things" in IoT refers to physical objects that have unique identifiers and can sense and communicate with their external environment (and user applications) or their internal physical states.
  - M2M systems, in contrast to IoT, typically have homogeneous machine types within an M2M area network.

Difference between IoT and M2M

- Hardware vs Software Emphasis
  - While the emphasis of M2M is more on hardware with embedded modules, the emphasis of IoT is more on software.
- Data Collection & Analysis
  - M2M data is collected in point solutions and often in on-premises storage infrastructure.
  - In contrast to M2M, the data in IoT is collected in the cloud (can be public, private or hybrid cloud).
- Applications
  - M2M data is collected in point solutions and can be accessed by on-premises applications such as diagnosis applications, service management applications, and on- premisis enterprise applications.
  - IoT data is collected in the cloud and can be accessed by cloud applications such as analytics applications, enterprise applications, remote diagnosis and management applications, etc.

Communication in IoT vs M2M




M2M applications
Manufacturing
Remote monitoring
Billing
Robotics
Security
Automotive
Logistics/Fleet management
Traffic control
Telemedicine
Utilities
Industrial

Primary Goal of M2M Communication

- To enable machines to interact and share information autonomously, thereby improving operational efficiency, reducing costs, and enhancing productivity.
- By facilitating seamless connectivity between devices, M2M empowers organizations to remotely monitor and manage assets, collect valuable data insights, and automate routine tasks.
- Whether it's tracking inventory levels in a warehouse, environmental monitoring in agricultural fields, or optimizing energy usage in buildings with smart metering solutions, *M2M communication enables smarter decision-making and proactive interventions*.

M2M Protocols

- At the core of the success of M2M communication within the broader landscape of IoT lies in the protocols governing device interaction and data exchange.
- These protocols, adapted for diverse applications and network environments, are pivotal in ensuring interoperability, reliability, and security.
- Among the notable M2M protocols are
  - MQTT (Message Queuing Telemetry Transport)
  - CoAP (Constrained Application Protocol)
  - AMQP (Advanced Message Queuing Protocol)

1.MQTT

- Message Queuing Telemetry Transport
- A messaging protocol that allows devices to communicate over the internet and is commonly used in the Internet of Things (IoT)
- It's designed to be lightweight, easy to implement, and efficient, making it ideal for use in resource-constrained environments.

Key Features Of MQTT

- **Origin** : MQTT was originally named MQ TT, or MQ Telemetry Transport, and was developed in the early 1990s.
- **Publish/subscribe** : MQTT uses a publish/subscribe (Pub/Sub) model, where a sender (publisher) and receiver (subscriber) communicate through topics. The MQTT broker filters messages and distributes them to the subscribers.
- **Low overhead** : MQTT has a small code footprint and low power consumption, making it ideal for devices with limited processing power and battery life.
- **Binary message format** : MQTT uses a binary message format for communication, which is different from other protocols like HTTP or SMTP that use text-based formats.
- **Security** : MQTT supports SSL/TLS, which allows clients to log in using a certificate. There are multiple TCP channels for different security levels, including unencrypted, encrypted, and encrypted with a client certificate.

MQTT

- Notable for its lightweight and efficient messaging
- Extensively utilized in IoT applications where bandwidth and power limitations are prevalent
- Publish-subscribe architecture allows devices to subscribe to topics and receive relevant messages
  - suitable for scenarios requiring real-time data updates and event-triggered communication

What is MQTT? [AWS….]

- A standards-based messaging protocol, or set of rules, used for machine-to-machine communication.
- Smart sensors, wearables, and other Internet of Things (IoT) devices typically have to transmit and receive data over a resource-constrained network with limited bandwidth.
- These IoT devices use MQTT for data transmission, as it is easy to implement and can communicate IoT data efficiently.
- MQTT supports messaging between devices to the cloud and the cloud to the device.

Why is the MQTT protocol important?

Benefits:

- **Lightweight and efficient**
  - MQTT implementation on the IoT device requires minimal resources, so it can even be used on small microcontrollers. For example, a minimal MQTT control message can be as little as two data bytes. MQTT message headers are also small so that you can optimize network bandwidth.
- **Scalable**
  - MQTT implementation requires a minimal amount of code that consumes very little power in operations. The protocol also has built-in features to support communication with a large number of IoT devices. Hence, you can implement the MQTT protocol to connect with millions of these devices.
- **Reliable**
  - Many IoT devices connect over unreliable cellular networks with low bandwidth and high latency. MQTT has built-in features that reduce the time the IoT device takes to reconnect with the cloud. It also defines three different quality-of-service levels to ensure reliability for IoT use cases— at most once (0), at least once (1), and exactly once (2).
- **Secure**
  - MQTT makes it easy for developers to encrypt messages and authenticate devices and users using modern authentication protocols, such as OAuth, TLS1.3, Customer Managed Certificates, and more.
- **Well-supported**
  - Several languages like Python have extensive support for MQTT protocol implementation. Hence, developers can quickly implement it with minimal coding in any type of application.

PRINCIPLE BEHIND MQTT

- *Space decoupling*
  - The publisher and subscriber are not aware of each other’s network location and do not exchange information such as IP addresses or port numbers.
- *Time decoupling*
  - The publisher and subscriber don’t run or have network connectivity at the same time.
- *Synchronization decoupling*
  - Both publishers and subscribers can send or receive messages without interrupting each other. For example, the subscriber does not have to wait for the publisher to send a message.

MQTT Components

1. **MQTT client**
   - Any device from a server to a microcontroller that runs an MQTT library
   - If the client is sending messages, it acts as a publisher, and if it is receiving messages, it acts as a receiver.
   - Basically, any device that communicates using MQTT over a network can be called an MQTT client device.
2. **MQTT broker**
   - The backend system which coordinates messages between the different clients
   - Responsibilities of the broker include receiving and filtering messages, identifying clients subscribed to each message, and sending them the messages.
   - It is also responsible for other tasks such as:
     - Authorizing and authenticating MQTT clients
     - Passing messages to other systems for further analysis
     - Handling missed messages and client sessions
3. **MQTT connection**
   - Clients and brokers begin communicating by using an MQTT connection.
   - Clients initiate the connection by sending a *CONNECT* message to the MQTT broker.
   - The broker confirms that a connection has been established by responding with a *CONNACK* message.
   - Both the MQTT client and the broker require a TCP/IP stack to communicate. Clients never connect with each other, only with the broker.

How does MQTT work?

1. An MQTT client establishes a connection with the MQTT broker.
2. Once connected, the client can either publish messages, subscribe to specific messages, or do both.
3. When the MQTT broker receives a message, it forwards it to subscribers who are interested.

MQTT topic

- The term ‘topic’ refers to keywords the MQTT broker uses to filter messages for the MQTT clients.
- Topics are organized hierarchically, similar to a file or folder directory.
  - For example, consider a smart home system operating in a multilevel house that has different smart devices on each floor.
  - In that case, the MQTT broker may organize topics as:

ourhome/groundfloor/livingroom/light

ourhome/firstfloor/kitchen/temperature

MQTT publish

- MQTT clients publish messages that contain the topic and data in byte format.
- The client determines the data format such as text data, binary data, XML, or JSON files.
- For example, a lamp in the smart home system may publish a message *on* for the topic

  livingroom/light

MQTT subscribe

- MQTT clients send a *SUBSCRIBE* message to the MQTT broker, to receive messages on topics of interest.
- This message contains a unique identifier and a list of subscriptions.
- For example, the smart home app on your phone wants to display how many lights are on in your house.
- It will subscribe to the topic *light* and increase the counter for all *on* messages.

What is MQTT over WSS?

- MQTT over WebSockets (WSS) is an MQTT implementation to receive data directly into a web browser.
- The MQTT protocol defines a JavaScript client to provide WSS support for browsers.
- In this case, the protocol works as usual but it adds additional headers to the MQTT messages to also support the WSS protocol.
- Think of it as the MQTT message payload wrapped in a WSS envelope.

2.CoAP : What Is The CoAP Protocol?

- Constrained Application Protocol
- A specialized internet application protocol for constrained devices.
- It was designed to allow small, low-power devices to join the Internet of Things (IoT).
- The protocol allows these devices to communicate with the wider Internet using minimal resources.
- A small base specification that can be extended with additional functionality when needed.
- It operates over UDP and provides a request/response interaction model between application endpoints, enabling interoperability among different types of devices.
- Highly reliable, with mechanisms in place to ensure message delivery, even in cases of limited network connectivity or device power.
  - This makes it suitable for IoT devices, which often operate in challenging network environments.

Key Features of CoAP

- RESTful Architecture
- Built-In Discovery
- Asynchronous Message Exchanges
- Optional Reliability with Confirmable Messages

RESTful Architecture

- CoAP uses a RESTful (Representational State Transfer) architecture.
- It follows a set of constraints that allow it to operate efficiently over a large, distributed network.
- In a RESTful system, data and functionality are considered resources, and these resources are accessed using a standard, uniform interface.

Built-In Discovery

- The CoAP protocol’s built-in discovery mechanism allows devices to discover resources on other devices without requiring any prior knowledge of their existence.
- This is especially useful in IoT networks, where devices may be constantly joining and leaving the network.
- The built-in discovery feature in CoAP is achieved through the use of a well-known 'core' resource that provides a list of available resources on a device.
- This resource can be queried by other devices on the network, allowing them to discover what resources are available and how to interact with them.

Asynchronous Message Exchanges

- CoAP supports asynchronous message exchanges, which is crucial for IoT networks where devices may not always be connected or available.
- With asynchronous message exchanges, a device can send a request to another device and then continue with other tasks without waiting for a response.
- The response can be processed once it arrives, even if delayed.
- This feature uses a message identifier in each CoAP message, which allows a device to match responses with requests.
- This, in conjunction with the ability to retransmit lost messages, ensures a high level of reliability in message exchanges.

Optional Reliability with Confirmable Messages

- CoAP offers optional reliability through the use of confirmable messages.
- When a device sends a confirmable message, it expects an acknowledgement from the recipient.
- If no acknowledgement is received within a certain time, the message is retransmitted.
- This feature allows CoAP to provide reliable communication in environments where network connectivity is unreliable.
- Devices can ensure that critical messages are received and processed.

Use Cases of CoAP

- Smart Home Automation
- Industrial IoT
- Wearables and Healthcare
- Energy Management

Pros of CoAP Protocol

- Lightweight
- Fast
- Efficient Encoding
- Stateless Communication

Cons of CoAP Protocol

- **Less Mature than Alternatives :**
  - less mature than some of its alternatives, such as HTTP and MQTT.
  - This means that there are fewer resources available for developers, such as libraries and tools, which can make the development process more challenging.
- **NAT Traversal :**
  - Network Address Translation
  - Because it uses UDP, which does not establish a connection before sending data, CoAP can have issues with NAT traversal, as the router may not know where to send the response.
  - To overcome this issue, the CoAP protocol needs to use techniques such as UDP hole punching, which can be complex and resource-intensive.
- **Fragmentation:**
  - occurs when a message is too large to fit into a single packet and needs to be divided into smaller fragments.
  - This can increase the complexity of the protocol and decrease its efficiency.
  - the loss of a single fragment can result in the loss of the entire message. This can be particularly problematic in unreliable networks, where packet loss is common.

CoAP vs. MQTT

Feature	MQTT	CoAP
Purpose	Messaging and communication in IoT	Designed for resource-constrained devices in IoT
Transport Protocol	TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
Communication Style	Publish/Subscribe model	Request/Response model
Header Size	2 bytes fixed header	4 bytes fixed header
Payload Format	Supports binary and text payloads	Supports binary and text payloads
QoS (Quality of Service)	Levels 0, 1, and 2 for message delivery	Reliability through "confirmable" and "non-confirmable" messages
Message Types	Publish, Subscribe, Connect, Disconnect, etc.	GET, POST, PUT, DELETE, etc.

CoAP vs. MQTT

Feature	MQTT	CoAP
Resource Discovery	Not inherent, requires additional mechanisms	Built-in resource discovery through CoRE Link Format
Security	Supports SSL/TLS for encryption and authentication	Datagram Transport Layer Security (DTLS) for secure communication
Connection Overhead	Maintains persistent connections	Lightweight connection setup
Scalability	Well-suited for large-scale deployments	Designed for constrained devices and networks
Header Compression	No built-in header compression	Uses CoAP-specific header compression
Message Compression	Supports message payload compression	Supports message payload compression
Use Cases	Wide range of IoT applications	Constrained devices with limited resources

3. AMQP (Advanced Message Queuing Protocol)

- AMQP is the Internet Protocol for Messaging
- The Advanced Message Queuing Protocol (AMQP) is an open standard for passing business messages between applications or organizations.
- It connects systems, feeds business processes with the information they need and reliably transmits onward the instructions that achieve their goals.

What is AMQP?

- Advanced Message Queuing Protocol
- It is a protocol that is used for communication between applications.
- It is a lightweight, protocol that supports the applications for data transfer.
- This protocol is used for its scalability and modularity with the technologies.

Components of AMQP

- **Exchanges:** The exchange is responsible for fetching messages and properly arranging them in the appropriate queue
- **Channel:** A channel is a multiplexed virtual connection between AMQP peers that is built into an existing connection.
- **Message Queue:** It is a unique entity that connects messages to their resources or points.
- **Binding:** Bindings are a set of predetermined instructions for queuing and exchanging. It manages message transmission and delivery.
- **Virtual Host:** Vhost is a platform that provides isolation capabilities within the broker. Multiple vhosts may be functional at the same time, depending on the users and their access rights.
- **Layers of AMQP**

Layers of AMQP

- **Function Layer:** The function layer handles basic file transfer transactions, message queues, access, and control streaming.
- **Transport layer:** Framing content data representation and error management.

AMQP Key Capabilities

AMQP connects across:

- Organizations – applications in different organizations
- Technologies – applications on different platforms
- Time – systems don’t need to be available simultaneously
- Space – reliably operate at a distance, or over poor networks

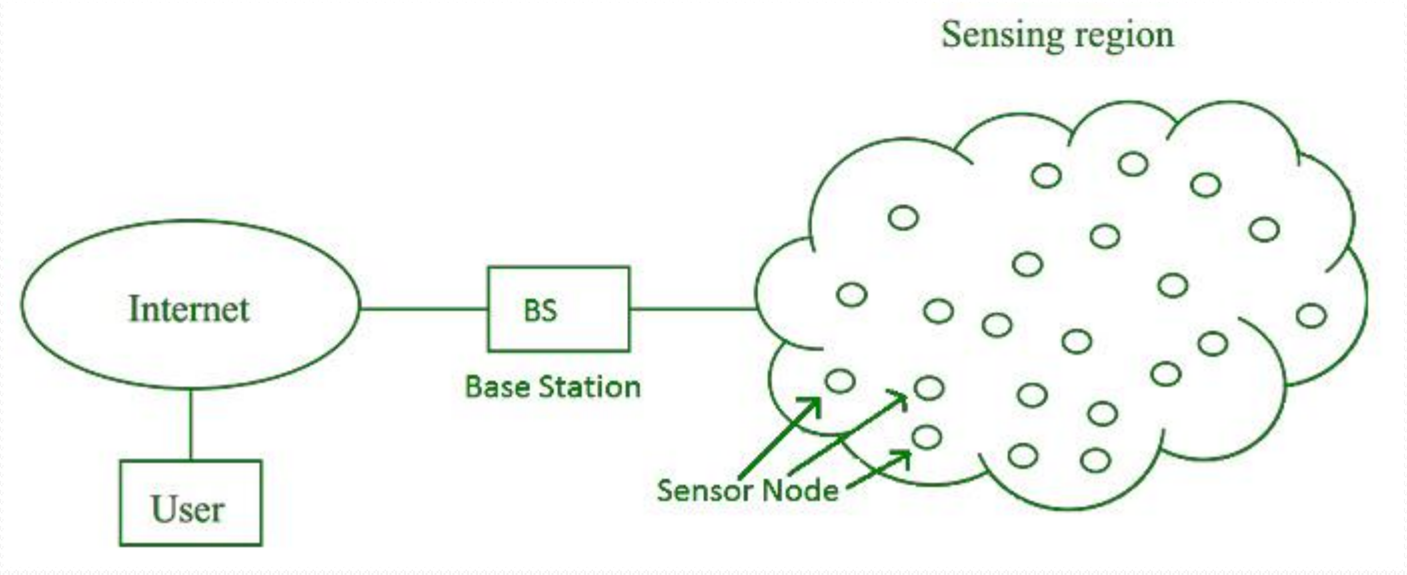
AMQP Key Features

- Security
- Reliability
- Interoperability
- Standard
- Open

Wireless Sensor Network (WSN)

- An infrastructure-less wireless network that is deployed in a large number of wireless sensors in an ad-hoc manner that is used to monitor the system, physical, or environmental conditions.
- Sensor nodes are used in WSN with the onboard processor that manages and monitors the environment in a particular area.
- They are connected to the Base Station which acts as a processing unit in the WSN System.
- The base Station in a WSN System is connected through the Internet to share data. WSN can be used for processing, analysis, storage, and mining of the data.

Wireless Sensor Network Architecture



Wireless Sensor Network Architecture

A Wireless Sensor Network (WSN) architecture is structured into three main layers:

- **Physical Layer**: This layer connects sensor nodes to the base station using technologies like radio waves, infrared, or Bluetooth. It ensures the physical communication between nodes and the base station.
- **Data Link Layer**: Responsible for establishing a reliable connection between sensor nodes and the base station. It uses protocols such as IEEE 802.15.4 to manage data transmission and ensure efficient communication within the network.
- **Application Layer**: Enables sensor nodes to communicate specific data to the base station. It uses protocols like ZigBee to define how data is formatted, transmitted, and received, supporting various applications such as environmental monitoring or industrial control.

- These layers work together to facilitate the seamless operation and data flow within a Wireless Sensor Network, enabling efficient monitoring and data collection across diverse applications.

Advantages of WSN

- **Low cost:** WSNs consist of small, low-cost sensors that are easy to deploy, making them a cost-effective solution for many applications.
- **Wireless communication:** WSNs eliminate the need for wired connections, which can be costly and difficult to install. Wireless communication also enables flexible deployment and reconfiguration of the network.
- **Energy efficiency:** WSNs use low-power devices and protocols to conserve energy, enabling long-term operation without the need for frequent battery replacements.
- **Scalability:** WSNs can be scaled up or down easily by adding or removing sensors, making them suitable for a range of applications and environments.
- **Real-time monitoring:** WSNs enable real-time monitoring of physical phenomena in the environment, providing timely information for decision making and control.

Disadvantages of WSN

- **Limited range:** The range of wireless communication in WSNs is limited, which can be a challenge for large-scale deployments or in environments with obstacles that obstruct radio signals.
- **Limited processing power:** WSNs use low-power devices, which may have limited processing power and memory, making it difficult to perform complex computations or support advanced applications.
- **Data security:** WSNs are vulnerable to security threats, such as eavesdropping, tampering, and denial of service attacks, which can compromise the confidentiality, integrity, and availability of data.
- **Interference:** Wireless communication in WSNs can be susceptible to interference from other wireless devices or radio signals, which can degrade the quality of data transmission.
- **Deployment challenges:** Deploying WSNs can be challenging due to the need for proper sensor placement, power management, and network configuration, which can require significant time and resources.
  - while WSNs offer many benefits, they also have limitations and challenges that must be considered when deploying and using them in real-world applications.

M2M and WSN Protocols

- Most M2M applications are developed today in a highly customized fashion
- High-level M2M architecture from M2M Standardization Task Force (MSTF) does include fixed & other non cellular wireless networks
  - Means it’s generic, holistic IoT architecture even though it is M2M architecture
- M2M and IoT sometimes are used interchangeably in the United States

M2M and WSN Protocols

- Other M2M standards activities include:
    - Data transport protocol standards - M2MXML, JavaScript Object Notation (JSON), BiTXML, WMMP, MDMP
    - Extend OMA [Open Mobile Alliance] DM [Device Management] to support M2M devices protocol management objects
    - M2M device management, standardize M2M gateway
    - M2M security and fraud detection
    - Network API's M2M service capabilities
    - Remote management of device behind gateway/firewall
    - Open REST-based API for M2M applications

Internet Protocol for Smart Objects

- The IPSO Alliance is an open, informal and thought-leading association of like-minded organizations and individuals that **promote the value of using the Internet Protocol for the networking of Smart Objects.**
- **IP Stack** can easily run on tiny, battery operated embedded devices as it is long-lived and stable technology.
- The role of the alliance is to ensure how IPv4, IPv6, and 6LoWPAN are used, deployed and provided to all potential users.

Internet Protocol for Smart Objects

- Mobile IP is an approach by IETF (Internet Engineering Task Force) which manages the movement of mobile devices over IPV4 and IPV6

M2M Standardization Efforts

M2M Standardization Task Force (MSTF) coordinate the efforts of individual **standards development organizations** (SDO)for M2M Applications

These **define a conceptual framework** for M2M applications and **specify a service layer** that will enable application developers to create applications that operate transparently across different vertical domains

M2M Standardization Efforts

M2M standards activities include the following

- Use JSON as Data Transport Format
- Resolve IP addressing issues for devices IPV6
- Use Open REST- based API for M2M applications
- Remote management of devices behind a gateway or firewall be done
- Fix the charging standars

WSN Standardization Efforts

There are number of standardization bodies in the field of WSNs

The IEEE focuses on the physical and MAC layers

IETF works on layers 3 and above.

WSN Standardization Efforts

IEEE 1451 is a set of smart transducer interface standards **developed by** the **IEEE Instrumentation and Measurement Society's Sensor Technology Technical Committee** that **describe** a set of open, common, network- **independent communication interfaces** for connecting sensors or actuators) to microprocessors, instrumentation systems, and control/field networks

The goal of the IEEE 1451 family of standards is to **allow the access of transducer data through a common set of interfaces whether the transducers are connected to systems or networks via a wired or wireless means**

WSN Standardization Efforts …IEEE 1451 Activities

1451.0-2007 Common Functions, Communication Protocols

1451.1-1999 Network Capable Application Processor Information Model

1451.2-1997 Transducer to Microprocessor Communication Protocols

1451.3-2003 Digital Communication Formats for Distributed Multi- drop Systems

WSN Standardization Efforts … IEEE 1451 Activities

1451.4-2004 Mixed- mode Communication Protocols

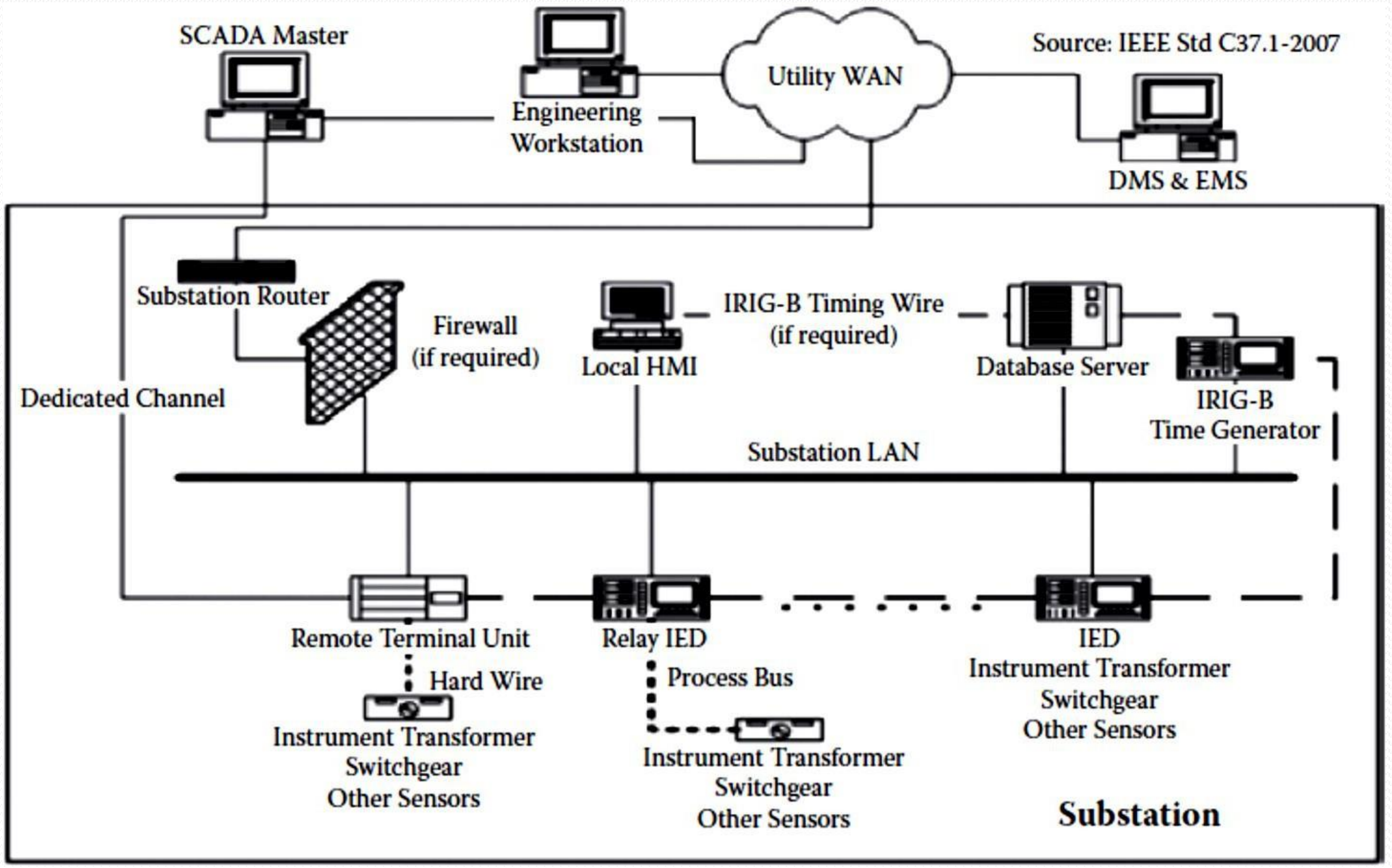
1451.5-2007 Wireless Communication Protocols

1451.7-2010 Transducers to Radio Frequency Identification (RFID) Systems Communication Protocols

SCADA and RFID Protocols

- Supervisory Control And Data Acquisition
- One of the IoT pillars to represent the whole industrial automation arena
- IEEE created standard specification called Std C37.1™, for SCADA & automation systems in 2007
- In recent years, network-based industrial automation has greatly evolved
- With the use of intelligent electronic devices (IEDs), or IoT devices in our terms, in substations and power stations

SCADA and RFID Protocols



SCADA and RFID Protocols

- The processing is now distributed
- Functions that used to be done at control center can now be done by IED i.e. M2M between devices
- Due to restructuring of electric industry, traditional vertically integrated electric utilities are replaced by many entities such as
  - GENCO (Generation Company),
  - TRANSCO (Transmission Company),
  - DISCO (Distribution Company),
  - ISO (Independent System Operator), etc.

Issues with IoT Standardization

- It should be noted that not everything about standardization is positive
- Standardization is like a double-edged sword:
  - Critical to market development
  - But it may threaten innovation and inhibit change when standards are accepted by the market
- Standardization and innovation are like yin & yang
- They could be contradictory to each other in some cases, even though this observation is debatable

Issues with IoT Standardization

- Different consortia, forums and alliances have been doing standardization in their own limited scope
- For example, 3GPP covers only cellular wireless networks while EPCglobal's middleware covers only RFID events
- Even within same segment, there are more than one consortium or forum doing standardization without enough communication with each other
- Some are even competing with each other

Issues with IoT Standardization

- Some people believe that the IoT concept is well established
- However, some gray zones remain in the definition, especially which technology should be included
- Following two issues for IoT standardization in particular and ICT standardization in general may never have answers:

Issues with IoT Standardization

1. ICT standardization is a highly decentralized activity. How can the individual activities of the network of extremely heterogeneous standards-setting bodies be coordinated?
2. It will become essential to allow all interested stakeholders to participate in the standardization process toward the IoT and to voice their respective requirements and concerns. How can this be achieved?

Unified Data Standards

- Already discussed about two pillars of the Internet
- HTML/HTTP combination of data format and exchange protocol is the foundation pillar of WWW
- Described great number of data standards and protocols proposed for four pillar domains of IoT
- Many issues still impede the development of IoT and especially WoT vision

Unified Data Standards

Unified Data Standards

- Many standardization efforts have been trying to define unified data representation, protocol for IoT
- Before IoT, Internet was actually an Internet of documents or of multimedia documents
- Two pillars of Internet including HTML/HTTP turned the Internet into WWW
- We need to turn the IoT into the WoT
- What will it take to make this to happen?

Unified Data Standards

- *Do we need a new HTML/HTTP-like standard for MTC and WoT? If there is no need to reinvent the wheel, what extensions do we need to build on top of HTML/HTTP or HTML5?*
- *Browser is intended for humans, so do we need new browser for machines to make sense of ocean of machine-generated data? If not, what extensions do we need to make to the existing browsers?*

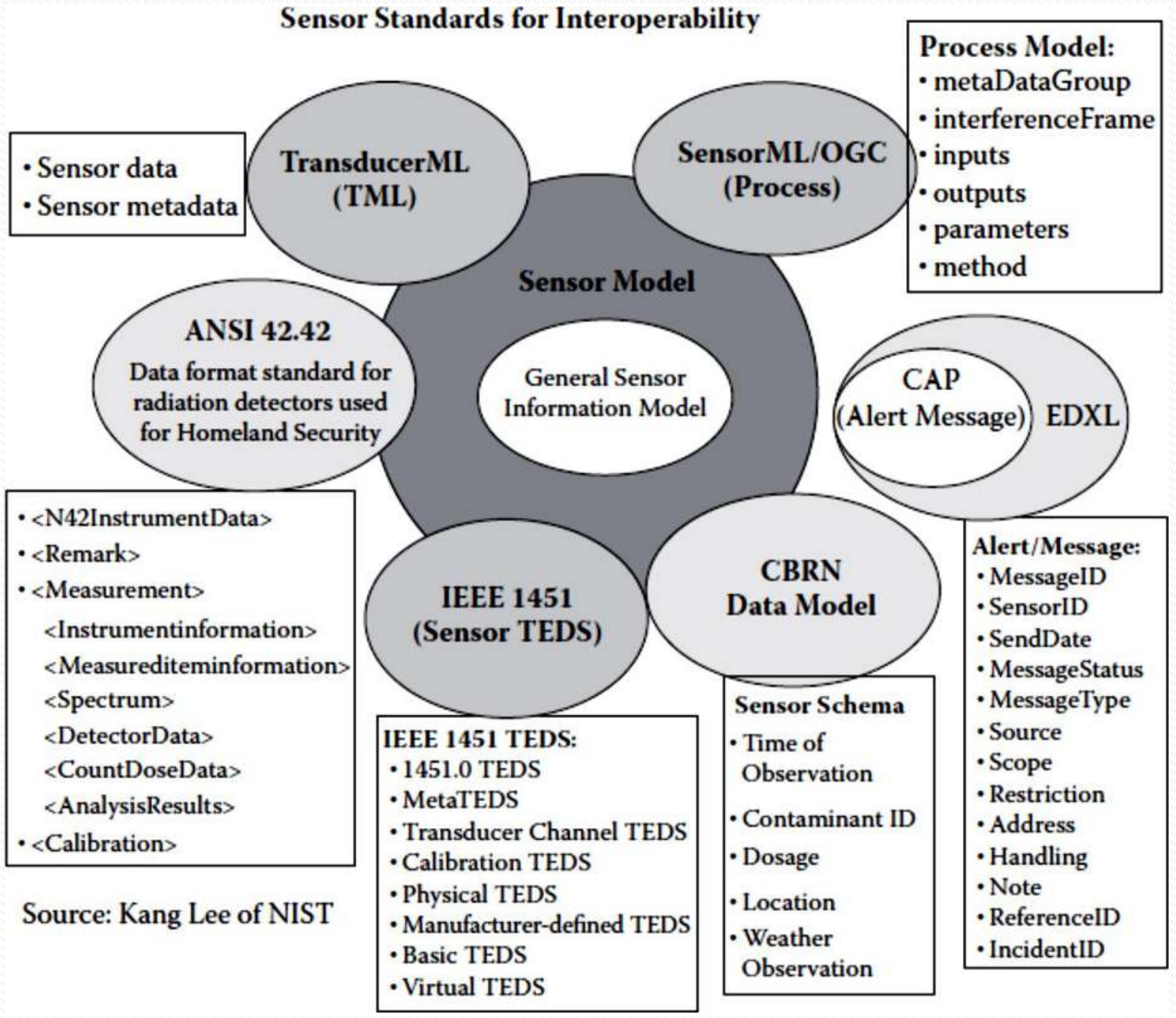
Unified Data Standards

- *Today, most new protocols are built on top of XML. For OS there must be XML-based data format standards or a metadata standard to represent the machine-generated data (MGD). Is it possible to define such a metadata standard that covers everything?*

Unified Data Standards

- There are many different levels of protocols
- But the ones that most directly relate to business and social issues are the ones closest to the top
- so-called application protocols such as HTML/HTTP for the web
- Web has always been visual medium, but restricted
- Until recently, HTML developers were limited to CSS & JavaScript in order to produce animations
- Or they would have to rely on a plug-in like Flash

Unified Data Standards



19/09/2024
9 am – 10 am