UNIT-3

IoT Architecture and ProtocolsIoT

R

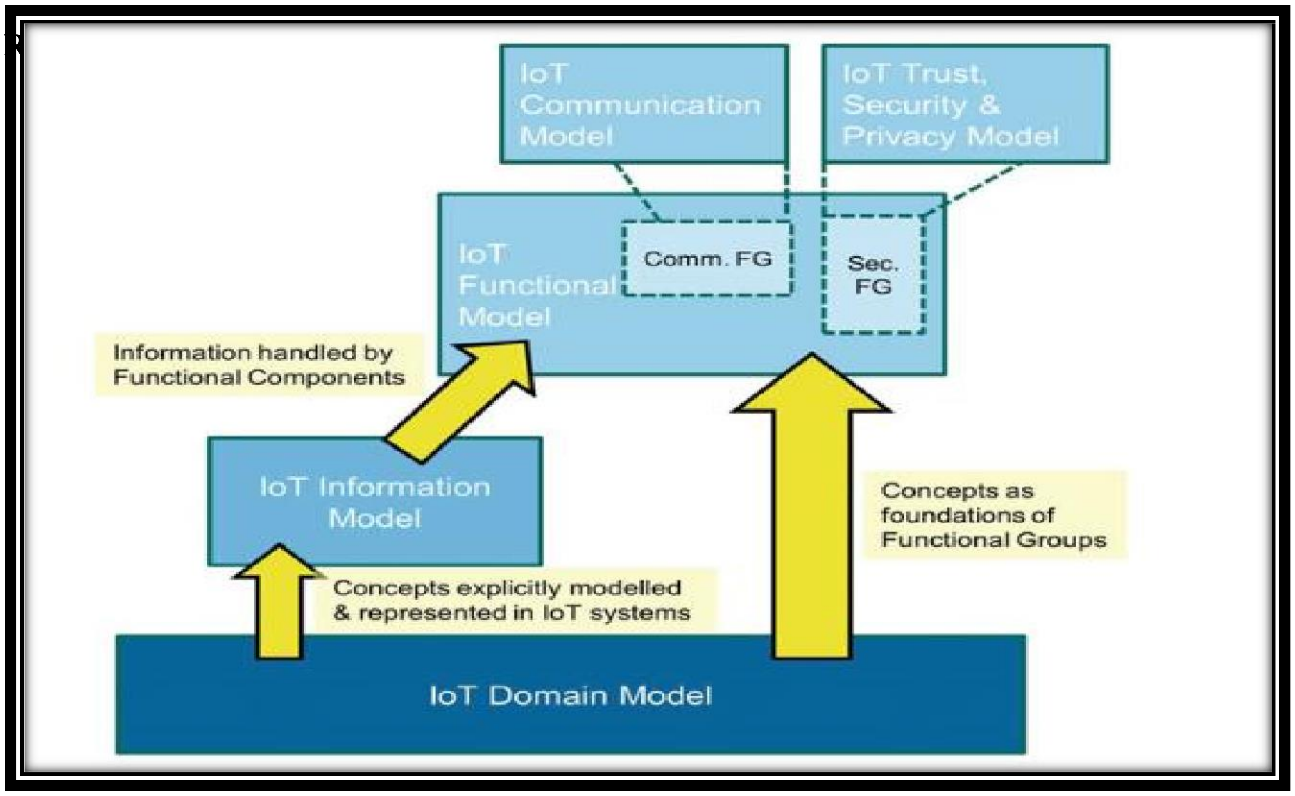

IoT Communication Model
IoT Trust, Security & Privacy Model
IoT Functional Model
Comm. FG
Sec. FG
Information handled by Functional Components
IoT Information Model
Concepts as foundations of Functional Groups
Concepts explicitly modelled & represented in IoT systems
IoT Domain Model

Fig. 1 Interaction of all sub-models in the IoT Reference Model.

Domain Model:

- The foundation of the IoT Reference Model is the IoT Domain Model, which introduces the main concepts of the Internet of Things like Devices, IoT Services and Virtual Entities (VE), and it also introduces relations between these concepts.
- Domain model defines the structure (e.g. relations, attributes) of IoT related information in an IoT system on a conceptual level without discussing how it would be represented.

Information Model:

- Based on the IoT Domain Model, the IoT Information Model has been developed. It defines the structure (e.g. relations, attributes) of IoT related information in an IoT system on a conceptual level without discussing how it would be represented
- The information pertaining to those concepts of the IoT Domain Model is modeled, which is explicitly gathered, stored and processed in an IoT system, e.g. information about Devices, IoT Services and Virtual Entities

Functional Model:

- The IoT Functional Model identifies groups of functionalities, of which most are grounded in key concepts of the IoT Domain Model.
- A number of these Functionality Groups (FG) build on each other, following the relations identified in the IoT Domain Model.
- The Functionality Groups provide the functionalities for interacting with the instances of these concepts or managing the information related to the concepts.
- e.g. information about Virtual Entities or descriptions of IoT Services
- The IoT Functional Model identifies groups of functionalities, of which most are grounded in key concepts of the IoT Domain Model.
- A number of these Functionality Groups (FG) build on each other, following the relations identified in the IoT Domain Model.
- The Functionality Groups provide the functionalities for interacting with the instances of these concepts or managing the information related to the concepts.
- e.g. information about Virtual Entities or descriptions of IoT Services
- The functionalities of the FGs that manage information use the IoT
- Information Model as the basis for structuring their information.
- IoT Functional Model identifies groups of functionalities, of which most are grounded in key concepts of the IoT Domain Model.
- A number of these Functionality Groups (FG) build on each other, following the relations identified in the IoT Domain Model.
- The Functionality Groups provide the functionalities for interacting with the instances of these concepts or managing the information related to the concepts, e.g. information about Virtual Entities or descriptions of IoT Services
- The functionalities of the FGs that manage information use the IoT Information Model as the basis for structuring their information.

Communication Model:

- The IoT Communication Model introduces concepts for handling the complexity of communication in heterogeneous IoT environments.
- Communication also constitutes one FG in the IoT Functional Model.

- A key functionality in any distributed computer system is the communication between the different components.
- One of the characteristics of IoT systems is often the heterogeneity of communication technologies employed, which often is a direct reflection of the complex needs such systems have to meet.

Trust, Security and Privacy Model:

- TSP are important in typical IoT use-case scenarios.
- Therefore, the relevant functionalities and their interdependencies and interactions are introduced in the IoT TSP Model.

Detailed Explanation for IoT reference model and sub models

Domain Model:

- Definition and Purpose
- The IoT-A project defines a domain model as a description of concepts belonging to a particular area of interest.
- The domain model also defines basic attributes of these Fig. 1 Interaction of all sub-models in the IoT Reference Model. The sub-models are explained in the text body 7 IoT Reference Model 115 concepts, such as name and identifier.
- The domain model defines relationships between concepts, for instance "Services expose Resources".
- Domain models also help to facilitate the exchange of data between domains.
- Domain model also provides a common lexicon and taxonomy of the IoT domain
- The main purpose of a domain model is to generate a common understanding of the target domain in question.
- Only with a common understanding of the main concepts it becomes possible to argue about architectural solutions and to evaluate them.
- The domain model is an important part of any reference model since it includes a definition of the main abstract concepts (abstractions), their responsibilities, and their relationships.

- For example, in the IoT domain, the device concept will likely remain relevant in the future, even if the types of devices used will change over time and/or vary depending on the application context.
- For instance, there are many technologies to identify objects: RFID, bar codes, image recognition etc.

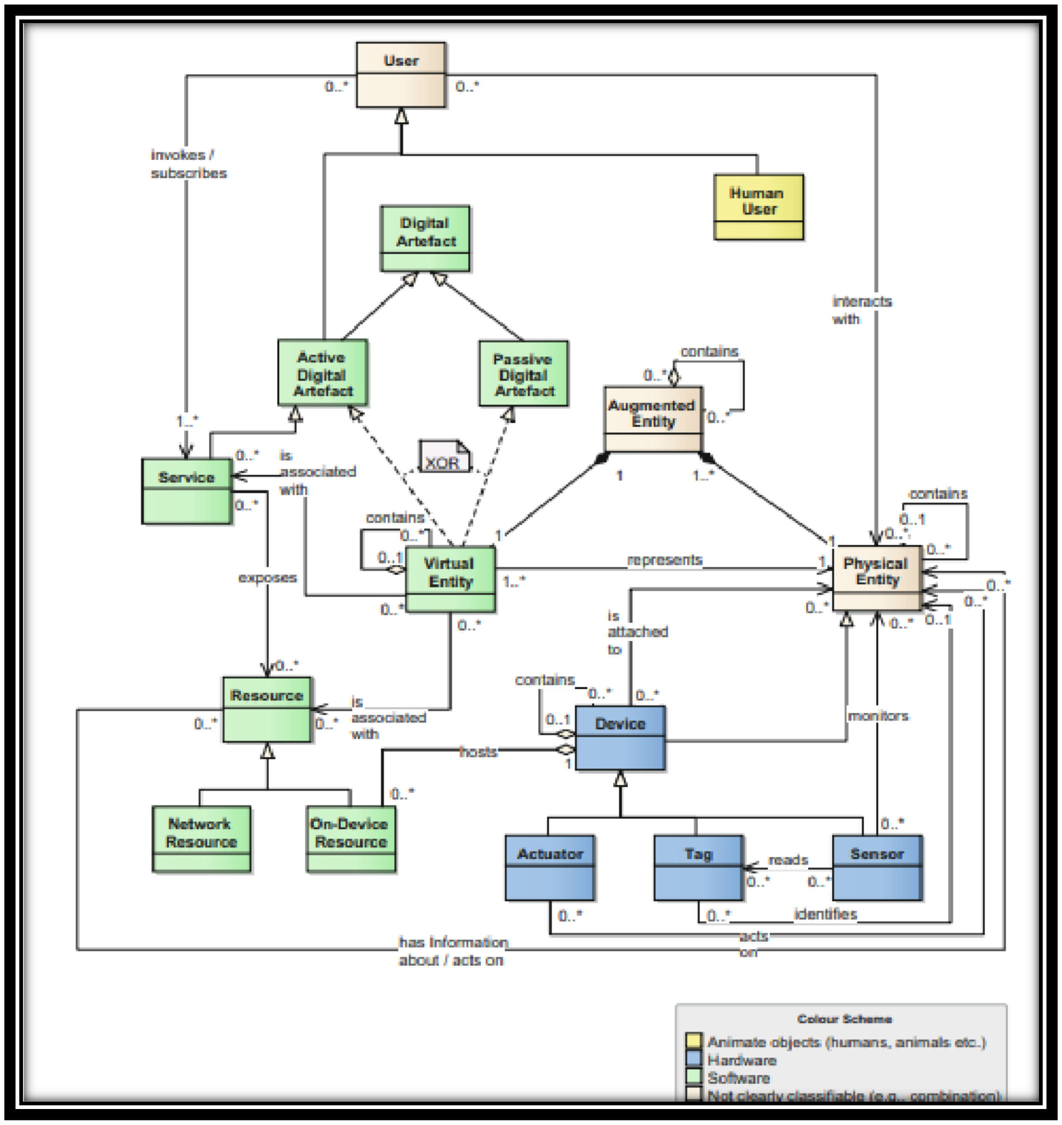


Fig 2.UML representation of the IoT Domain Model

Description of UML representation of IoT Domain Model

- **Sensors** provide information, knowledge, or data about the Physical Entity they monitor

An example for the latter is a face-recognition enabled camera. Information from sensors can be recorded for later retrieval

- **Tags** are used to identify Physical Entities, to which the Tags are usually physically attached..
- The identification process is called “reading”, and it is carried out by specific Sensor Devices, which are usually called readers.
- The primary purpose of Tags is to facilitate and increase the accuracy of the identification process.
- **Actuators** can modify the physical state of a Physical Entity, like changing the state (translate, rotate, stir, inflate, switch on/off,...) of simple Physical Entities or activating/deactivating functionalities of more complex ones.
- **Devices** are pieces of hardware, such as sensors, actuators, gadgets, appliances, or machines, that are programmed for certain applications and can transmit data over the internet or other networks
- **Device** (identifier, identifier + data, sensor data, or commands) and the communication topology (network, reader-tag, peer-to-peer, etc.).
- **Resource** Resources are software components that provide some functionality. The resources may perform any tasks that are useful for the system, network or end user applications.
- These are like **Network Resources, On-Device Resources**
- Resources can either run on a Device – hence called On-Device Resources – or they can run somewhere in the network (Network Resources).
- **On-Device Resources** are typically sensor Resources that provide sensing data or actuator Resources.
- **Network Resources** run on a dedicated server in the network or in the “cloud”, they do not rely on special hardware that allows direct connection to the physical world
- **Virtual Entities** are Digital Artefacts that can be classified as either active or passive.
- **Active Digital Artefacts (ADA)** are running software applications, agents or Services that may access other Services or Resources.
- **Passive Digital Artefacts (PDA)** are passive software elements such as database entries that can be digital representations of the Physical Entity.
- IoT **Devices** are pieces of hardware, such as sensors, actuators, gadgets, appliances, or machines, that are programmed for certain applications and can transmit data over the internet or other networks.

- **Service** – Services are the mechanism by which needs and capabilities are brought together. It represents a set of end-to-end services in which businesses contract with external providers to design, build, install and operate IoT solutions, including advisory consulting for IoT planning.
- **Services** provide the link between the IoT aspects of a system and other, non-IoT specific parts of an information system.
- **IoT Services** provide well-defined and standardised interfaces, hiding the complexity of accessing a variety of heterogeneous Resources.
- IoT Services can be classified by their level of abstraction are
- **Resource-level Services** expose the functionality, usually of a Device, by accessing its hosted Resources. These kinds of Services refer to a single Resource.
- **Virtual Entity-level Services** provide access to information at a Virtual Entitylevel.
- **Integrated Services** are the result of a Service composition of Resource-level or combinations of both Service abstractions.
- **Physical Entity** : Physical Entity is a discrete and identifiable entity in the physical environment (e.g. a room, a light, an appliance, a car, etc.).
- **Virtual Entity** : Virtual Entity is a representation of the Physical Entity in the digital world.
- A physical entity is represented by a virtual entity on the digital level.
- An **augmented entity** combines the two and stands for any combination of the two entities.
- **Users** inform about their needs and desires, and provide feedback within a networked intelligence to jointly improve their individual ability to rule the actuators of the system at their service.

Information Model:

- The IoT Information Model defines the structure (e.g. relations, attributes, services) of all the information for Virtual Entities on a conceptual level
- The representation of the information (e.g. binary, XML (Extensible Markup Language), RDF (Resource Description Framework) etc.)
- Information model defines structure of the information that is handled and processed in an IoT System.

- The main aspects are represented by the elements VirtualEntity, ServiceDescription and Association.
- Virtual Entity needs to have a unique identifier (identifier) or entity type (entityType), defining the type of the Virtual Entity representation, e.g. a human, a car or a temperature sensor.
- The IoT Information Model is a **meta-model** that defines the structure of key aspects of the information being managed in an IoT system

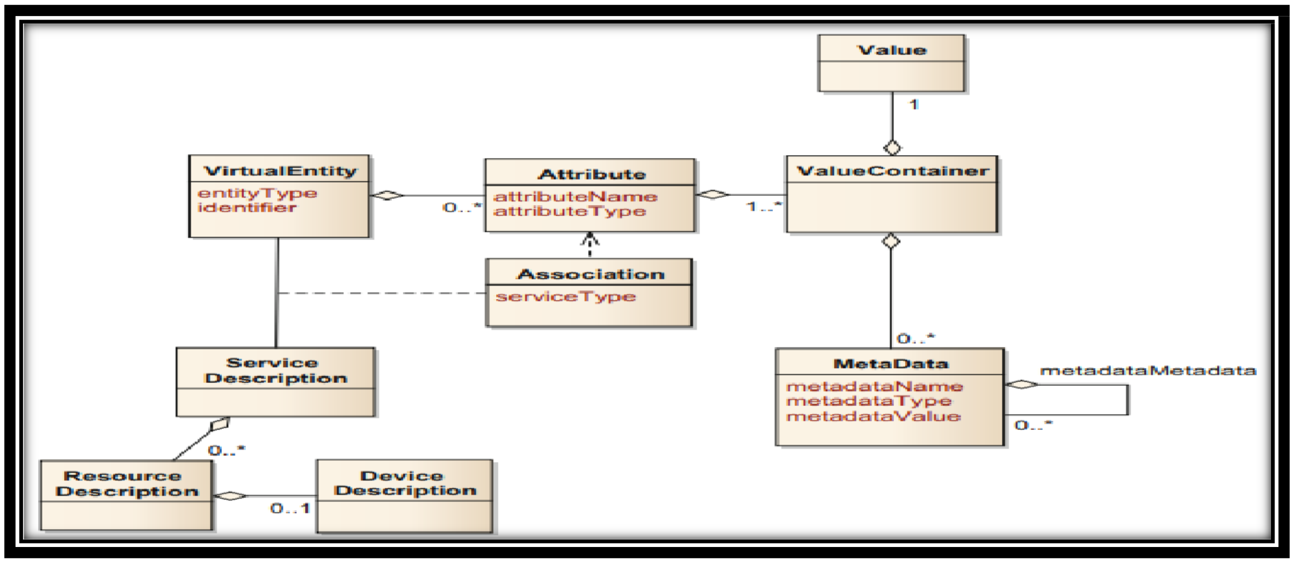


Fig 3. IoT Information Model

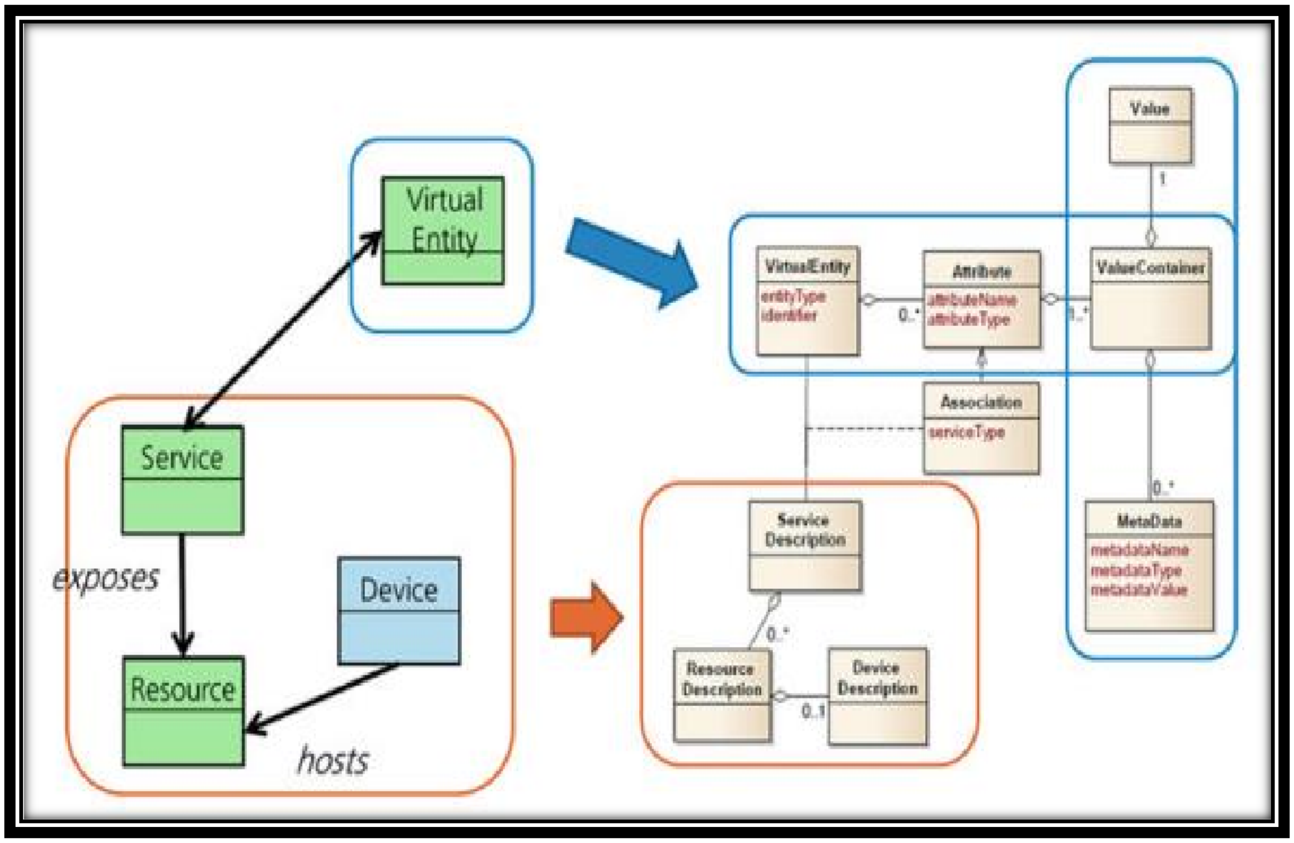


Fig. 4 Relation between the core concepts of the IoT Domain Model and IoT Information Model

- **Entity model**: The Entity Model specifies which attributes and features of real word objects are represented by the virtual counterpart, i.e. the Virtual Entity of the respective Physical Entity.
- **Resource model:** The Resource Model contains the information that is essential to identify Resources by a unique identifier and to classify Resources by their type, like sensor, actuator, processor or tag.
- **Service description model:** Services provide access to Resources and are used to access information or to control Physical Entities. A Service Description describes a Service, using for instance a service description language such as USDL (Unified Service Description Language)
- **Event Model:** Event models are quite essential in today's IoT architectures, e.g. in the EPCglobal Information Services. Normally events are used to track dynamic changes in a (software) system, showing who or what has triggered it and when, where and why the change occurred.

- EPC (Electronic Product Code)
- EPCglobal Architecture Framework is a collection of interrelated hardware, software, and data standards

Functional Model:

- The Functional Model is an abstract framework for understanding the main Functionality Groups (FG) and their interactions.
- This framework defines the common semantics of the main functionalities and will be used for the development of IoT-A compliant Functional Views.

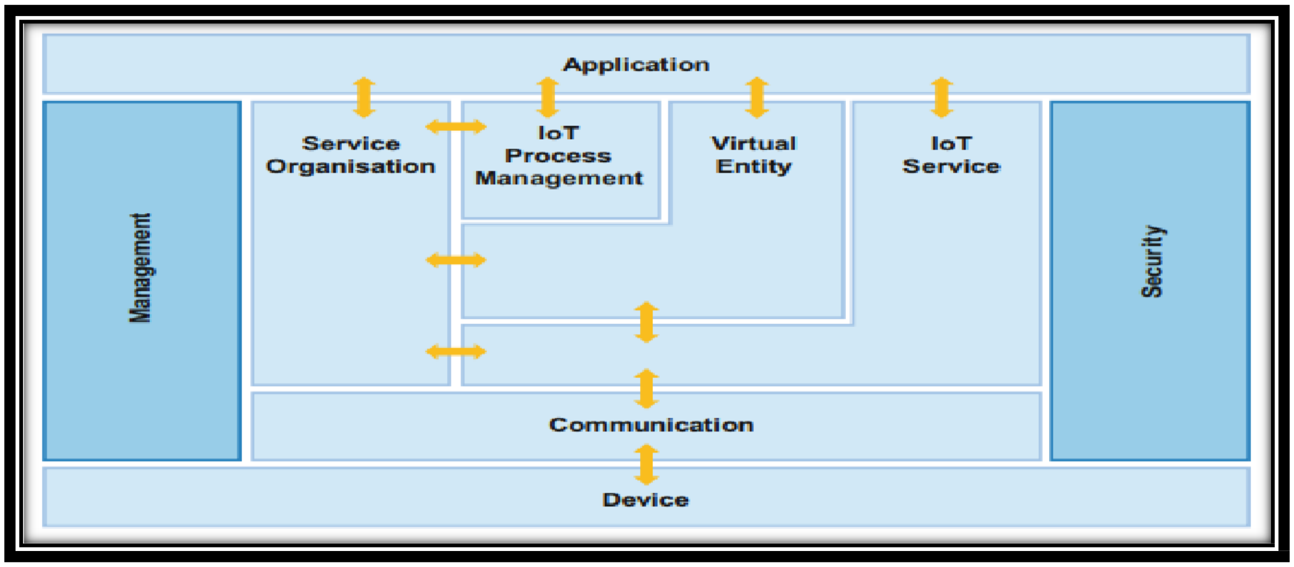


Fig 5. Functional Model

- The Functional Model contains seven longitudinal Functionality Groups (light blue) complemented by two transversal Functionality Groups (Management and Security, dark blue).
- Functional Model is a hierarchical model and the main interactions between the FG's are depicted with orange arrows.
- **The IoT Functional Model aims** at describing mainly the Functional Groups (FG) and their interaction with the ARM, while the Functional View of a Reference Architecture describes the functional components of an FG, interfaces, and interactions between the components.

- ***Device functional group***
- The Device FG contains all the possible functionality hosted by the physical Devices that are used for increment the Physical Entities.
- This Device functionality includes sensing, actuation, processing, storage, and identification components, the sophistication of which depends on the Device capabilities. The Functional View is typically derived from the Functional Model in conjunction with high-level requirements.
- ***Communication functional group***
- The Communication FG abstracts all the possible communication mechanisms used by the relevant Devices in an actual system in order to transfer information to the digital world components or other Devices.
- ***IoT Service functional group***
- The IoT Service FG corresponds mainly to the Service class from the IoT Domain Model, and contains single IoT Services exposed by Resources hosted on Devices or in the Network (e.g. processing or storage Resources).
- ***Virtual Entity functional group***
- The Virtual Entity FG corresponds to the Virtual Entity class in the IoT Domain Model, and contains the necessary functionality to manage associations between Virtual Entities with themselves as well as associations between Virtual Entities and related IoT Services, i.e. the Association objects for the IoT Information Model.
- Associations between Virtual Entities can be static or dynamic depending on the mobility of the Physical Entities related to the corresponding Virtual Entities.
- ***IoT Service Organization functional group***
- The purpose of the IoT Service Organisation FG is to host all functional components that support the composition and orchestration of IoT and Virtual Entity services.
- Moreover, this FG acts as a service hub between several other functional groups such as the IoT Process Management FG.
- for example, service requests from Applications or the IoT Process Management are directed to the Resources implementing the necessary Services.

- ***IoT Process Management functional group***
- The IoT Process Management FG is a collection of functionalities that allows smooth integration of IoT-related services (IoT Services, Virtual Entity Services, Composed Services) with the Enterprise (Business) Processes.
- ***Management functional group***
- The Management FG includes the necessary functions for enabling fault and performance monitoring of the system, configuration for enabling the system to be flexible to changing User demands, and accounting for enabling subsequent billing for the usage of the system.
- Support functions such as management of ownership, administrative domain, rules and rights of functional components, and information stores are also included in the Management FG.
- ***Security functional group***
- The Security FG contains the functional components that ensure the secure operation of the system as well as the management of privacy.
- The Security FG contains components for Authentication of Users (Applications, Humans), Authorisation of access to Services by Users, secure communication (ensuring integrity and confidentiality of messages) between entities of the system such as Devices, Services, Applications, and last but not least, assurance of privacy of sensitive information relating to Human Users.
- ***Application functional group***
- The Application FG is just a placeholder that represents all the needed logic for creating an IoT application.
- The applications typically contain custom logic tailored to a specific domain such as a Smart Grid

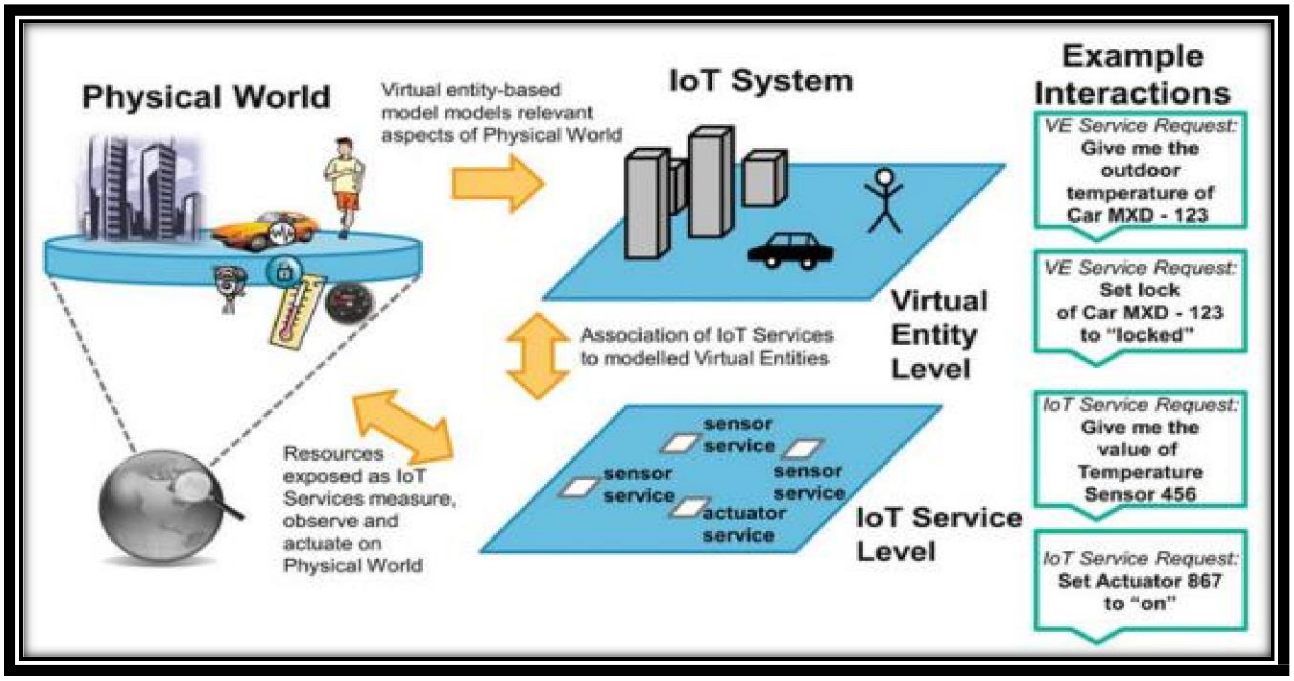

Physical World
Virtual entity-based model models relevant aspects of Physical World
IoT System
Example Interactions
VE Service Request: Give me the outdoor temperature of Car MXD - 123
VE Service Request: Set lock of Car MXD - 123 to "locked"
Virtual Entity Level
Association of IoT Services to modelled Virtual Entities
IoT Service Request: Give me the value of Temperature Sensor 456
sensor service
sensor service
sensor service
actuator service
Resources exposed as IoT Services measure, observe and actuate on Physical World
IoT Service Level
IoT Service Request: Set Actuator 867 to "on"

Fig 6. IoT-Service and Virtual-Entity abstraction levels

Communication Model:

- IoT Communication Model aims at defining the main communication paradigms for connecting elements, as defined in the IoT Domain Model.
- Communication Model that leverages on the ISO OSI 7-layer model for networks
- IoT Communication Model, it is important to identify the communication system elements and/or the communicating Users among those defined in the IoT Domain Model.

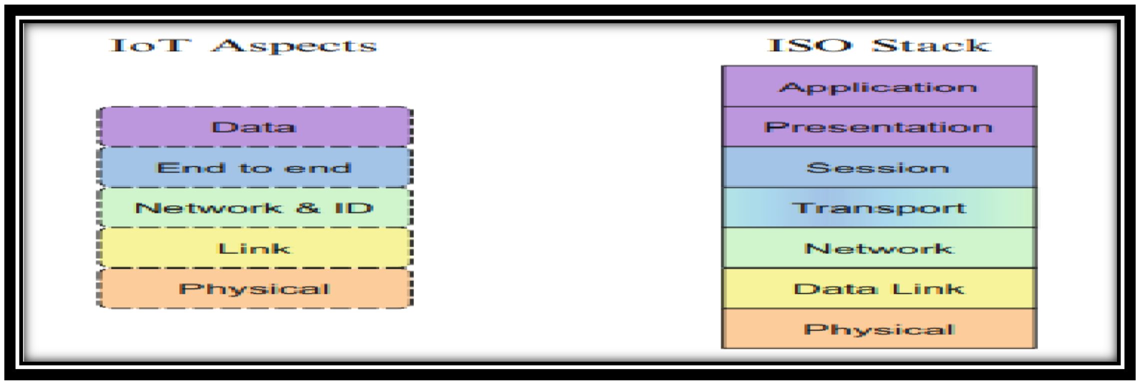


Fig 7. The interoperability aspects of the IoT Communication Model (left) compared to the ISO/OSI

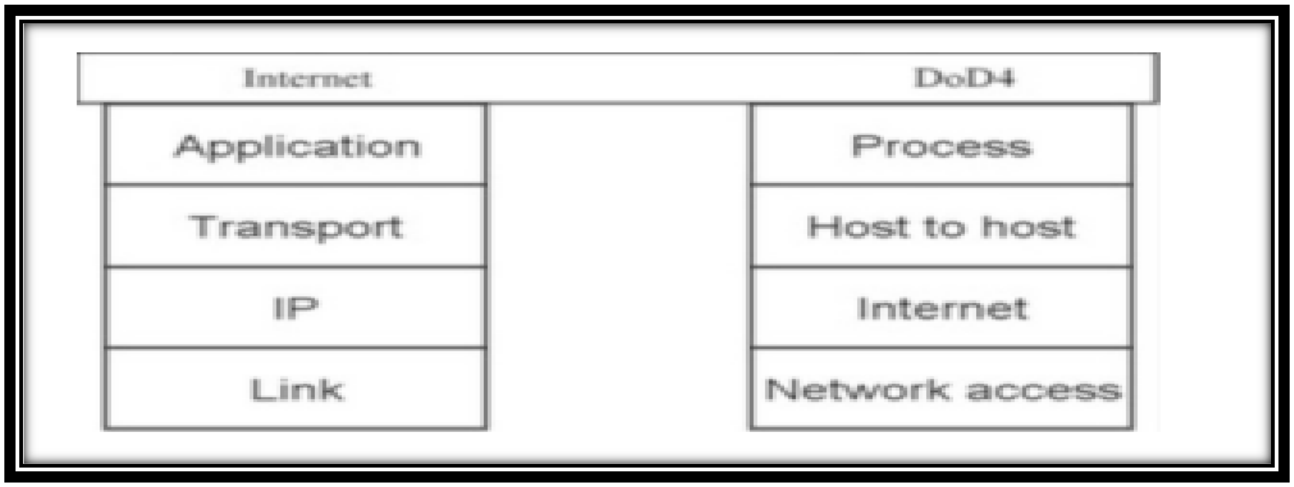


Fig 8. Four layers Internet stack (left) and DoD4 (Department of Defense) stack (right)

- ***Communication Model***
- ***Safety:*** The IoT Reference Model can only provide IoT-related guidelines for ensuring a safe system to the extent possible and controllable by a sys- tem designer. Eg: smart grid.
- ***Privacy***
- Because interactions with the physical world may often include humans, protecting the User privacy is of utmost importance for an IoT system. The IoT-A Privacy Model depends on the following functional components: Identity Management, Authentication, Authorisation, and Trust & Reputation

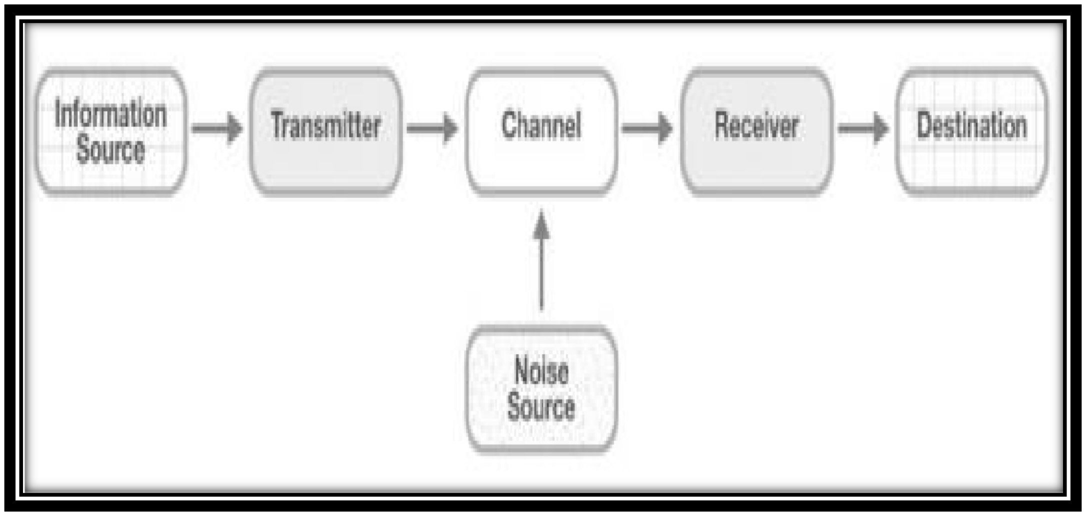


Fig 9. Schematic diagram of a general communication system

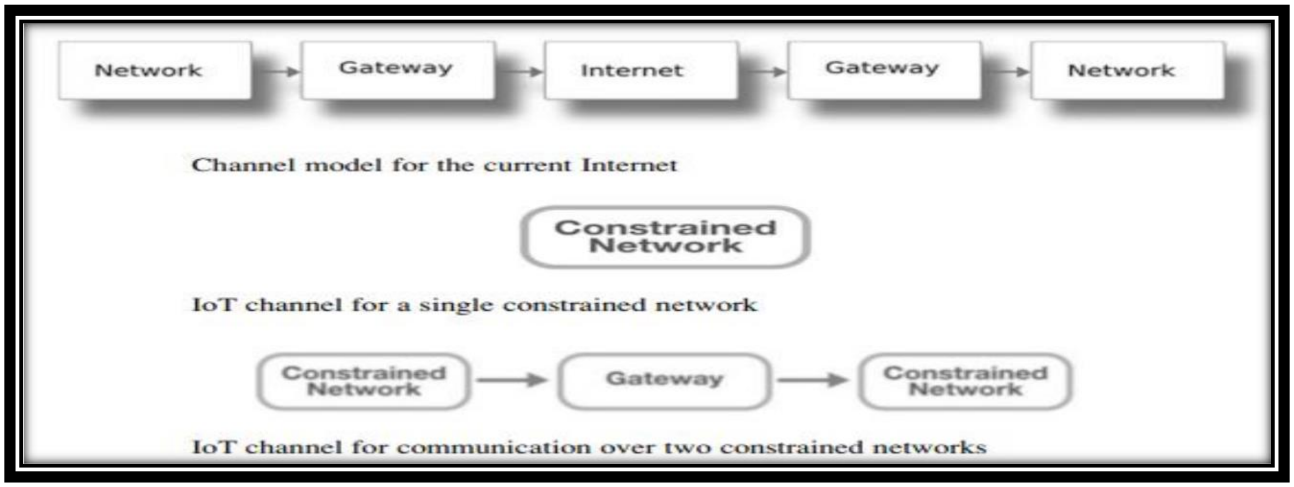

Network
Gateway
Internet
Gateway
Network
Channel model for the current Internet
Constrained Network
IoT channel for a single constrained network
Constrained Network
Gateway
Constrained Network
IoT channel for communication over two constrained networks

Fig 10. IoT channel for communication over two constrained networks

- A constrained network is composed of a significant portion of constrained nodes. Mostly, these constrained node networks are deployed in the edge network of an IoT system.

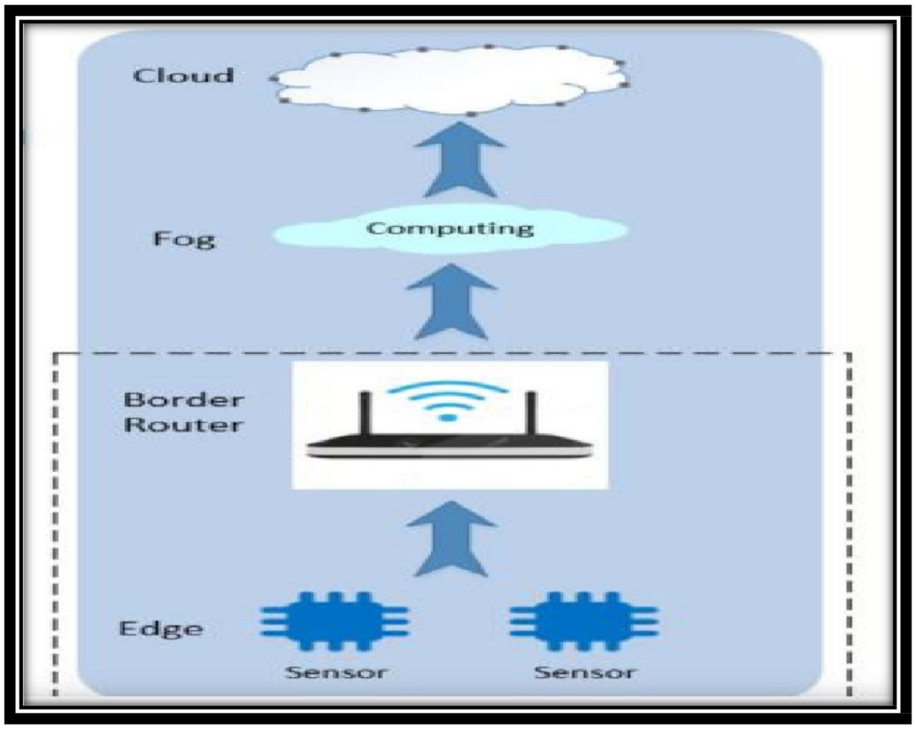


Fig 11. Constrained Networks

Trust, Security, Privacy

- **Trust:** Trust Model that provides data integrity and confidentiality, and endpoint authentication and non-repudiation between any two system-entities that interact with each other.

Trust Model Mandatory Aspects

- The Trust-Model domains
- Trust-evaluation mechanisms
- Behavior policies
- Trust anchor
- Federation of trust
- M2M support
- **Security:** IoT Security is the act of securing Internet devices and the networks they're connected to from threats and breaches by protecting, identifying, and monitoring risks all while helping fix vulnerabilities from a range of devices that can pose security risks to your business.
- Security reference model is made of three layers:
- The Service Security layer
- The Communication Security layer
- The Application Security layer.
- **Privacy:** Because interactions with the physical world may often include humans, protecting the User privacy is of utmost importance for an IoT system.
- The IoT-A Privacy Model depends on the following functional components: Identity Management, Authentication, Authorisation, and Trust & Reputation
- Privacy Model aim is
- To describe the mechanisms – e.g. access policies, encryption/decryption algorithms, security mechanisms based on credentials, and so on – that prevent data of a subject to be used improperly.
- ***Safety:*** The IoT Reference Model can only provide IoT-related guidelines for ensuring a safe system to the extent possible and controllable by a system designer. Eg: smart grid.

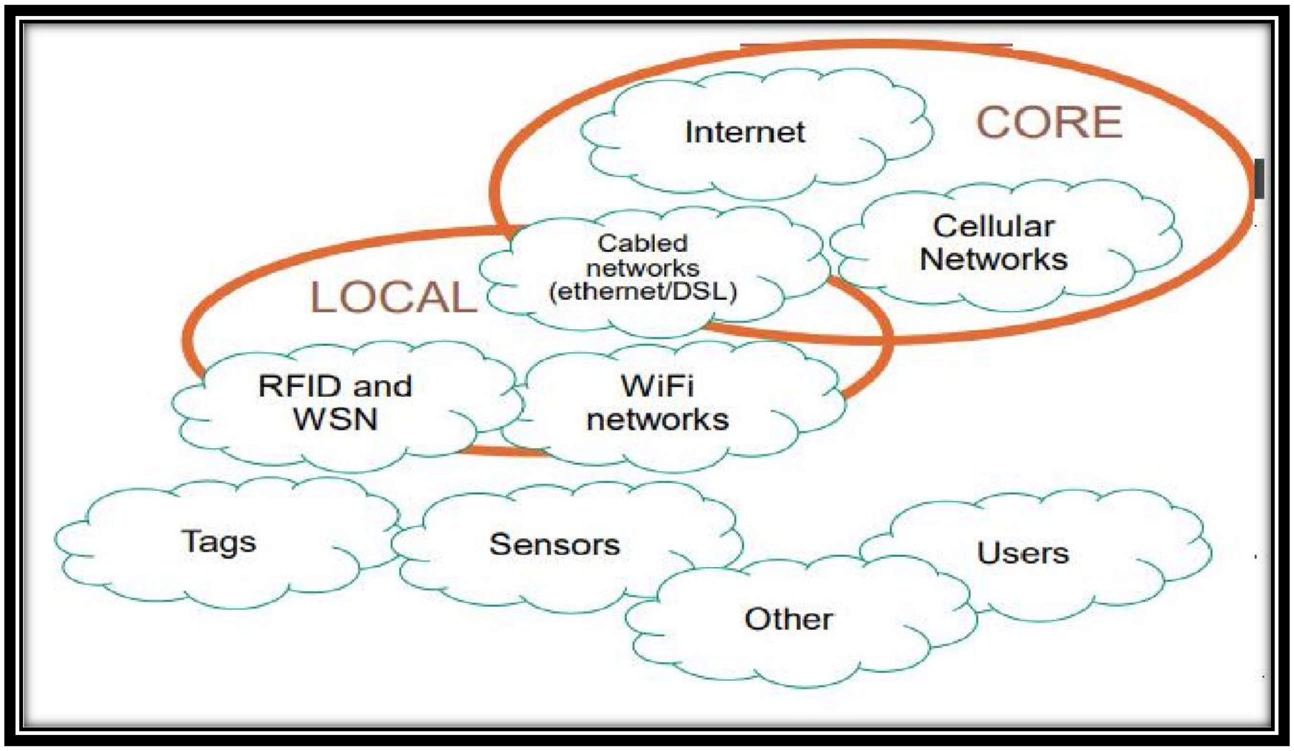


Fig 12. Deployment & Operation

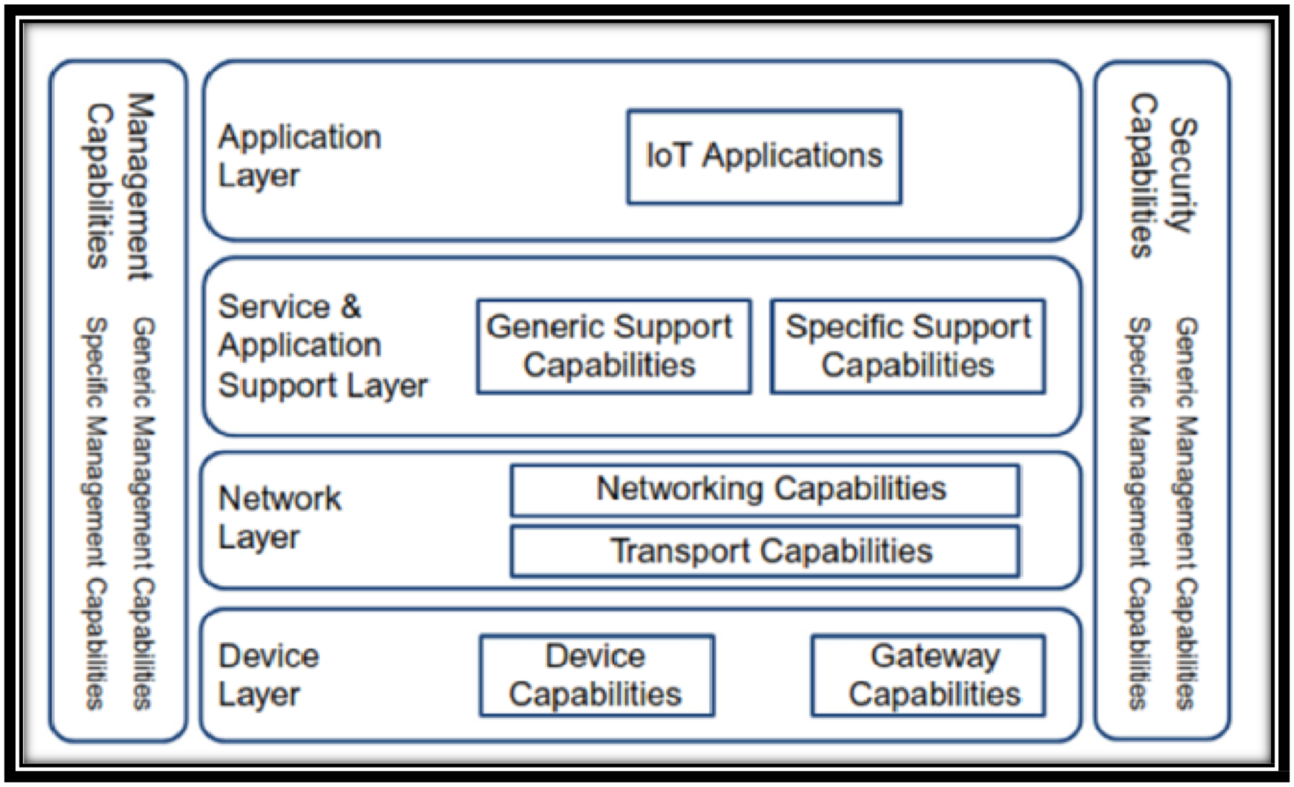
Conclusion:

- In this Section we discussed the IoT Reference Model.
- The IoT Reference Model defines the basic concepts, models, terminology, and relationships in the IoT ARM.
- It demonstrates our thinking, rationale and design space for structuring the domain of the Internet of Things.
- It also proposes the Functional Groups that we deem relevant for IoT architectures, as outlined in the IoT Functional Model

IoT Architecture Reference Model:

- **IoT reference model is a four-layer model with associated management and security capabilities. The four layers of the model are as follows: application layer, service support and application support layer, network layer, and the device layer.**

Architecture Reference Model



Management Capabilities
Generic Management Capabilities
Specific Management Capabilities
Application Layer
IoT Applications
Service & Application Support Layer
Generic Support Capabilities
Specific Support Capabilities
Network Layer
Networking Capabilities
Transport Capabilities
Device Layer
Device Capabilities
Gateway Capabilities
Security Capabilities
Generic Management Capabilities
Specific Management Capabilities

Fig 13. ITU-T , IoT Architecture Reference Model

- **Application Layer:**
- The Application Layer is the simplest layer to explain.
- This layer contains the IoT applications.
- Those IoT applications depend on IoT system and its end users (e.g. smart transport, e-health, smart agriculture, etc.)
- **Service Support and Application Support Layer:**
- The Service Support and Application Support Layer consists of two capability groups:
- Generic and specific support capabilities.

- Generic support capabilities are common capabilities that can be used by different IoT applications, e.g. data processing or data storage.
- Specific support capabilities are particular capabilities that catered for the requirements of specific applications.
- **Network Layer:**
- The Network Layer also has two following types of capabilities:
- Networking and Transport capabilities.
- **Networking capabilities** provide access and transport resource control functions, mobility management or authentication, authorization, and accounting (AAA).
- **Transport capabilities** provide connectivity for the transport of service and application-specific data, as well as the transport of IoT-related control and management information
- **Device Layer:**
- Device Layer capabilities are categorized into two groups:
- Device and Gateway capabilities.
- **Device capabilities** include direct interaction with the communication network. This is for the simplest form of IoT systems where end devices in IoT systems collect data and upload collected data directly to the communication network.
- Also, vice versa, the end devices are capable of receiving information (e.g., commands) form the core of the communication network.
- **Gateway capabilities** include multiple interface support.
- This feature is needed because end devices can use a variety of short-range technologies for data collections, such as: CAN bus, ZigBee, Bluetooth, Bluetooth Low Energy or Wi-Fi;
- Long-range technologies for data transfer such as the public switched telephone network (PSTN), 2G/3G/4G/5G mobile networks
- Long-term evolution networks (LTE)
- Variety of emerging Low-power wide-area network (LPWAN) technologies such as LoRa, LoRaWAN SigFox, NB-IoT, LTE-M, etc.
- IoT **Management capabilities** cover fault management, configuration management, accounting management, performance management, and security management.

- Again, two groups can be defined.
- Essential **Generic management capabilities** in the IoT include device management (remote device activation and de-activation, diagnostics, firmware and/or software updating, device working status management), local network topology management (traffic and congestion management).
- **Specific management** capabilities are closely related to application-specific requirements, e.g., smart grid power transmission line monitoring requirements, etc.
- **Security capabilities** consist of Generic and Specific capabilities.
- **Generic security capabilities** are independent of applications, and their functions depend on the layer.
- At the level of the Application Layer, they include authorization, authentication, application data confidentiality, and integrity protection, privacy protection, security audit, and anti-virus functions.
- At the Network Layer, they include authorization, authentication, user and signaling data confidentiality, and signaling integrity protection.
- Finally, at the Device Layer capabilities include authentication, authorization, device integrity validation, access control, data confidentiality, and integrity protection.
- **Specific security capabilities** are closely coupled with application-specific requirements, e.g., mobile payment, security requirements

6LoWPAN:

- 6LoWPAN is an IPv6 protocol, and It's extended from is IPv6 over Low Power Personal Area Network.
- This protocol works on Wireless Personal Area Network i.e., WPAN.
- WPAN is a Personal Area Network (PAN) where the interconnected devices are centered around a person's workspace and connected through a wireless medium.
- IPv6 is Internet Protocol Version 6 is a network layer protocol that allows communication to take place over the network. It is faster and more reliable and provides a large number of addresses.
- 6LoWPAN has very low cost, short-range, low memory usage, and low bit rate.

- It comprises an Edge Router and Sensor Nodes. Even the smallest of the IoT devices can now be part of the network, and the information can be transmitted to the outside world as well. For example, LED Streetlights.

Edge Router and Sensor Nodes

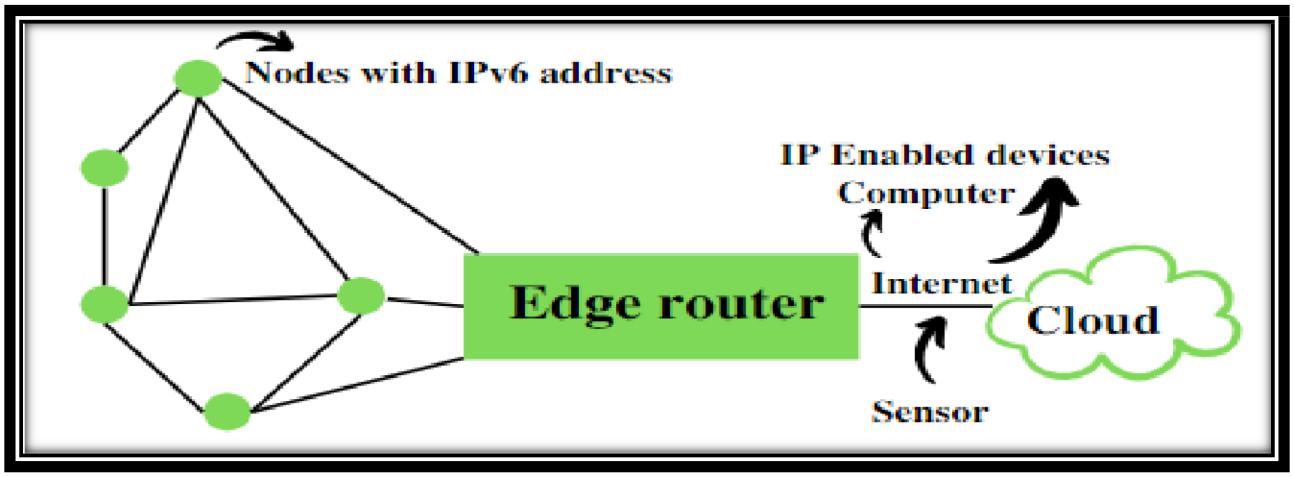


Fig 14. Edge Router and Sensor Nodes

- It is a technology that makes the individual nodes IP enabled.
- 6LoWPAN can interact with 802.15.4 devices and also other types of devices on an IP Network. For example, Wi-Fi.
- It uses AES 128 link layer security, which AES is a block cipher having key size of 128/192/256 bits and encrypts data in blocks of 128 bits each. This is defined in IEEE 802.15.4 and provides link authentication and encryption. **(AES – Advanced Encryption Techniques)**
- **Basic Requirements of 6LoWPAN:**
- The device should be having sleep mode in order to support the battery saving.
- Minimal memory requirement.
- Routing overhead should be lowered.
- **Features of 6LoWPAN:**
- It is used with IEEE 802.15,.4 in the 2.4 GHz band.
- Outdoor range: ~200 m (maximum)
- Data rate: 200kbps (maximum)

- Maximum number of nodes: ~100
- **Advantages of 6LoWPAN:**
- 6LoWPAN is a mesh network that is robust, scalable, and can heal on its own.
- It delivers low-cost and secure communication in IoT devices.
- It uses IPv6 protocol and so it can be directly routed to cloud platforms.
- It offers one-to-many and many-to-one routing.
- In the network, leaf nodes can be in sleep mode for a longer duration of time.
- **Disadvantages of 6LoWPAN:**
- It is comparatively less secure than Zigbee.
- It has lesser immunity to interference than that Wi-Fi and Bluetooth.
- Without the mesh topology, it supports a short range.
- **Applications of 6LoWPAN:**
- It is a wireless sensor network.
- It is used in home-automation,
- It is used in smart agricultural techniques, and industrial monitoring.
- It is utilised to make IPv6 packet transmission on networks with constrained power and reliability resources possible.

RPL (IPv6 Routing protocol)

- **RPL** stands for **Routing Protocol for Low Power and Lossy Networks** for heterogeneous traffic networks.
- It is a routing protocol for Wireless Networks.
- This protocol is based on the same standard as by Zigbee and 6 Lowpan is IEEE 802.15.4 It holds both many-to-one and one-to-one communication.
- The RPL protocol is implemented using the Contiki Operating system.
- This Operating System majorly focuses on IoT devices, more specifically Low Power Wireless IoT devices.
- It is an Open source Model and was first bought into the picture by Adam Dunkels.

- The RPL protocol mostly occurs in wireless sensors and networks
- The RPL protocol is implemented using the Contiki Operating system.
- This Operating System majorly focuses on IoT devices, more specifically Low Power Wireless IoT devices.
- It is an Open source Model and was first bought into the picture by Adam Dunkels.
- The RPL protocol mostly occurs in wireless sensors and networks

CoAP (Constrained Application Protocol):

- Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things.
- **CoAP (Constrained Application Protocol)** is a session layer protocol that provides the RESTful (HTTP) interface between HTTP client and server.
- CoAP enables low-power sensors to use RESTful services while meeting their low power constraints.
- This protocol is specially built for IoT systems primarily based on HTTP protocols.
- CoAP is designed to enable simple, constrained devices to join the IoT even through constrained networks with low bandwidth and low availability.

CoAP Architecture:

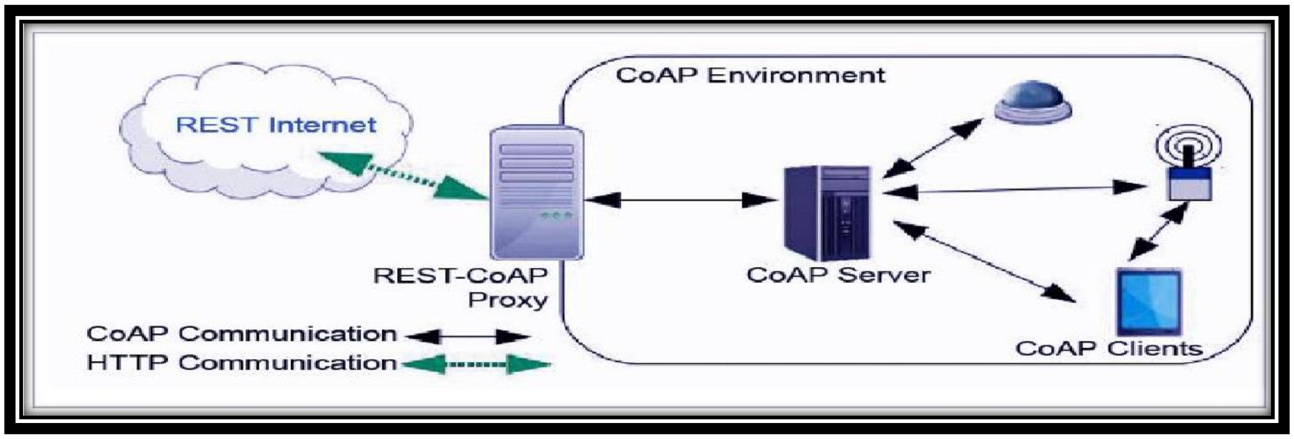


Fig 15. CoAP Architecture

This network is used within the limited network or in a constrained environment. The whole architecture of CoAP consists of CoAP client, CoAP server, REST CoAP proxy, and REST internet.

The data is sent from CoAP clients (such as smartphones, RFID sensors, etc.) to the CoAP server and the same message is routed to REST CoAP proxy.

The REST CoAP proxy interacts outside the CoAP environment and uploads the data over REST internet.

* REST : Representational State Transfer

MQTT (Message Queue Telemetry Transport):

- **MQTT (Message Queue Telemetry Transport)** is a messaging protocol which was introduced by IBM in 1999.
- It was initially built for monitoring sensor node and faraway tracking in IoT.
- Its suits are small, cheap, low-memory and low-power devices.
- MQTT provides embedded connectivity between applications and middleware in one side and another side it connects networks and communicators.

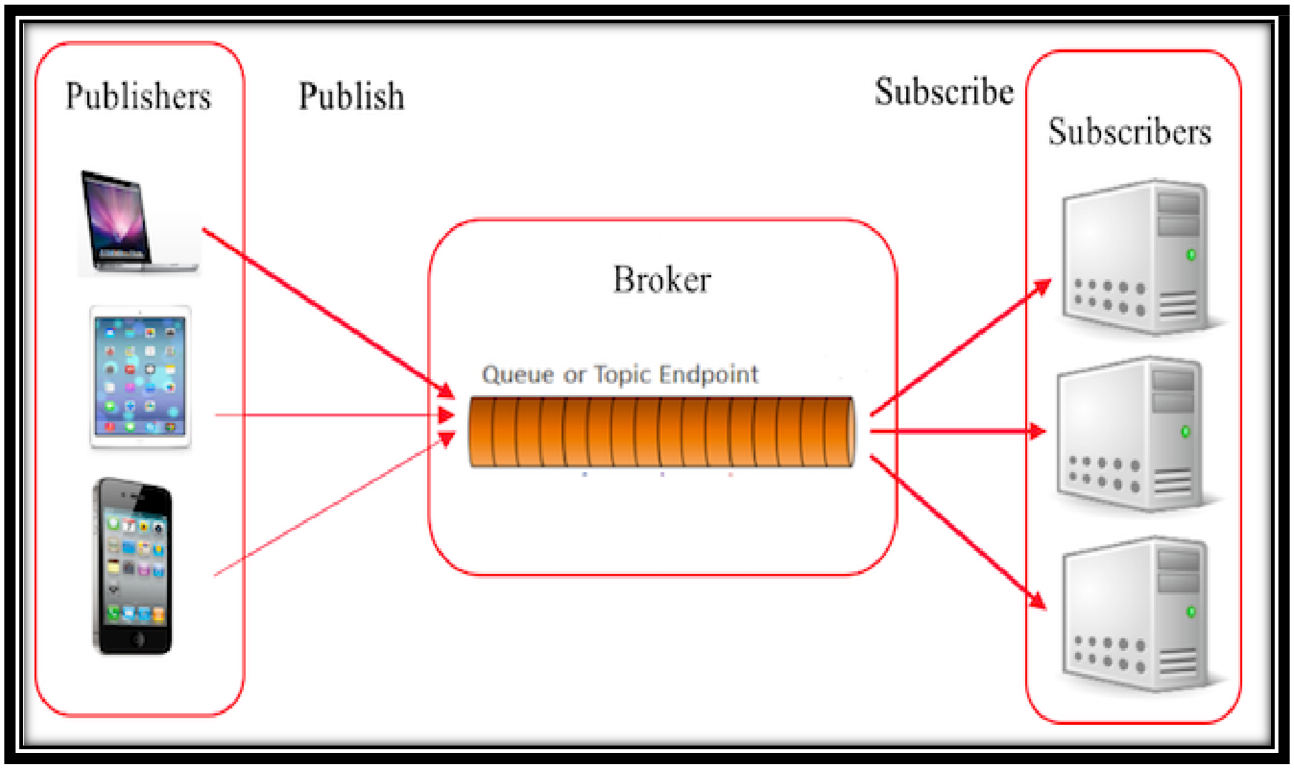


Fig 16. MQTT Architecture

- MQTT protocol is based on publish/subscribe architecture.

- The publish/subscribe architecture consists of three major components: publishers, subscribers, and a broker.
- According to IoT point of view, **publishers are lightweight sensor devices** that send their data to connected broker and goes back to sleep whenever possible.
- **Subscribers are applications**, which are interested in a certain topic or sensory data, so they are connected to brokers to be informed whenever new data are received.
- The broker receives the sensory data and filters them in different topics and sends them to subscribers according to interest in the topics.

IoT frameworks

- An IoT framework can be defined as a set of protocols, tools, and standards that provide a specific structure for developing and deploying IoT applications and services.
- In other words, an IoT framework gives you the basics for building your own application.
- An IoT framework provides you with a solid foundation to build your application.
- When you use a framework, you get access to a range of features and capabilities for the development of IoT solutions, applications, and smart connected products.

IoT Framework Design Goals

- IoT frameworks have four primary design goals:

1. Reduce development time and bring IoT solutions to market sooner
2. Reduce apparent complexity of deploying and operating an IoT network
3. improve application portability and interoperability
4. improve serviceability, reliability, and maintainability.

- There are both open-source and proprietary IoT frameworks out there.
- Open-source usually indicates that the source code is available for everyone to use and improve.
- On the other hand, tech giants, such as Amazon Web Services, Google Cloud, and Cisco, offer proprietary, paid IoT platforms.

Popular open-source IoT frameworks :

- **DeviceHive, Mainflux, Thinger.io, Kaa enterprise IoT platform**

- **DeviceHive:** DeviceHive is an open-source IoT data platform that helps you build innovative and scalable IoT/M2M solutions.
- You can use the framework to accelerate your IoT development and build secure and cost-effective IoT solutions and applications for a wide range of industries.
- DeviceHive is a microservice-based framework that supports Python, Node.js, and Java client libraries.
- **Mainflux:** Mainflux is a secure, scalable, and open-source framework to develop and manage Internet of Things solutions.
- Depending on the project needs, development teams can deploy Mainflux across on-premises, cloud, or hybrid environments.
- Mainflux enables developers to integrate new functionalities and features into their projects using its easy-to-use APIs (Application Programming Interfaces)
- **Thinger.io:**Thinger.io is a cloud-based IoT framework that lets IoT developers connect thousands of devices in a matter of minutes.
- According to the official website, the goal behind building and improving this platform is to democratize the use of IoT, making it accessible to the whole world and streamlining the development of large-scale IoT projects.
- **Kaa enterprise IoT :** Based on a flexible microservices architecture, the Kaa enterprise IoT platform provides everything that you need to develop an enterprise-grade IoT application - from device connection and management to data collection, IoT dashboards, and analytics.
- Depending on your project needs, you can deploy the Kaa framework on-prem, in the cloud, or across both in a hybrid model.

Proprietary IoT frameworks : Amazon Web Services IoT, Azure IoT, Cisco IoT solutioins

- **Amazon Web Services IoT :** AWS Internet of Things falls under the category of proprietary frameworks.
- It provides a broad and deep range of services and solutions to help IoT teams build intelligent IoT solutions with artificial intelligence (AI) and machine learning (ML) capabilities.
- With AWS IoT, an IoT team can easily connect and manage billions of devices and safeguard their device data with preventative mechanisms like encryption and access control.

- IoT application development companies use AWS IoT offerings to complete complex IoT projects and streamline data analysis for industrial, commercial, customer, and automotive workloads.
- Additionally, the platform provides a highly secure and elastic infrastructure for IoT teams to scale easily and reliably.

Proprietary IoT frameworks :

- **Azure IoT :** Azure IoT helps businesses build, deploy, and manage IoT applications at scale.
- IoT development agencies across the world tap into Azure's IoT capabilities to develop IoT applications and solutions for manufacturing, retail, energy, healthcare, and logistics.
- The cloud services platform provides development teams with intelligent edge-to-cloud technologies and an ecosystem of thousands of partners to enable businesses to solve industry-specific business challenges.
- **Cisco IoT solutions :** Cisco's end-to-end IoT solutions are applied across industries such as manufacturing, utilities, smart cities, ports and terminals, rail, roadways, etc.
- IoT teams use Cisco's IoT solutions to connect, manage, and scale assets, applications, and data in real-time to drive business results.

Thing Speak

- ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualization of live data, and send alerts.
- **Collect** : Send sensor data privately to the cloud.
- **Analyze**: Analyze and visualize your data with MATLAB.
- **Act** : Trigger a reaction.
- ThingSpeak Features:

1. Collect data in private channels
2. Share data with public channels
3. RESTful and MQTT APIsMATLAB analytics and visualizations
4. Event scheduling

5. Alerts
6. App integrations

ThingSpeak Works with:

- MATLAB® & Simulink®
- Arduino®
- Particle devices
- ESP8266 and ESP32 Modules
- Raspberry Pi™
- LoRaWAN®
- Things Network
- Senet
- Libelium
- Beckhoff