MODULE - IV

IoT Architecture-State of the Art
Introduction, State of the art, **Architecture Reference Model-** Introduction, Reference Model and architecture, IoT reference Model.
IoT Reference Architecture: Introduction, Functional View, Information View, Deployment and Operational View, Other Relevant architectural views.

4.1 Introduction

An ARM is useful as a tool that establishes a common language across all the possible stakeholders of an M2M or IoT system. It can also serve as a starting point for creating concrete architectures of real systems when the relevant boundary conditions have been applied.

4.2 State of the art

Several Reference Architectures and Models exist both for M2M and IoT systems. The four most popular ones from which the specific Reference Architecture is discussed below

4.2.1 European Telecommunications Standards Institute M2M/oneM2M

The European Telecommunications Standards Institute (ETSI) in 2009 formed a Technical Committee (TC) on M2M topics aimed at producing a set of standards for communication among machines from an end to- end viewpoint.

- The ETSI M2M specifications are based on specifications from ETSI as well as other standardization bodies such as the IETF (Internet Engineering Task Force), 3GPP (3rd Generation Partnership Project), OMA (Open Mobile Alliance), and BBF (Broadband Forum).
- ETSI M2M produced the first release of the M2M standards in early 2012, while in the middle of 2012 seven of the leading Information and Communications Technology (ICT) standards organizations (ARIB, TTC, ATIS, TIA, CCSA, ETSI, TTA) formed a global organization called oneM2M Partnership Project (oneM2M) in order to develop M2M specifications, promote the M2M business, and ensure the global functionality of M2M systems.

4.2.1.1 ETSI M2M high-level architecture

The figure 4.1 shows the high-level ETSI M2M architecture.This high-level architecture is a combination of both a functional and topological view showing some functional groups (FG) clearly associated with pieces of physical infrastructure (e.g. M2M Devices, Gateways) while other functional groups lack specific topological placement.

There are two main domains, a network domain and a device and gateway domain. The boundary between these conceptually separated domains is the topological border between the physical devices and gateways and the physical communication infrastructure (Access network).

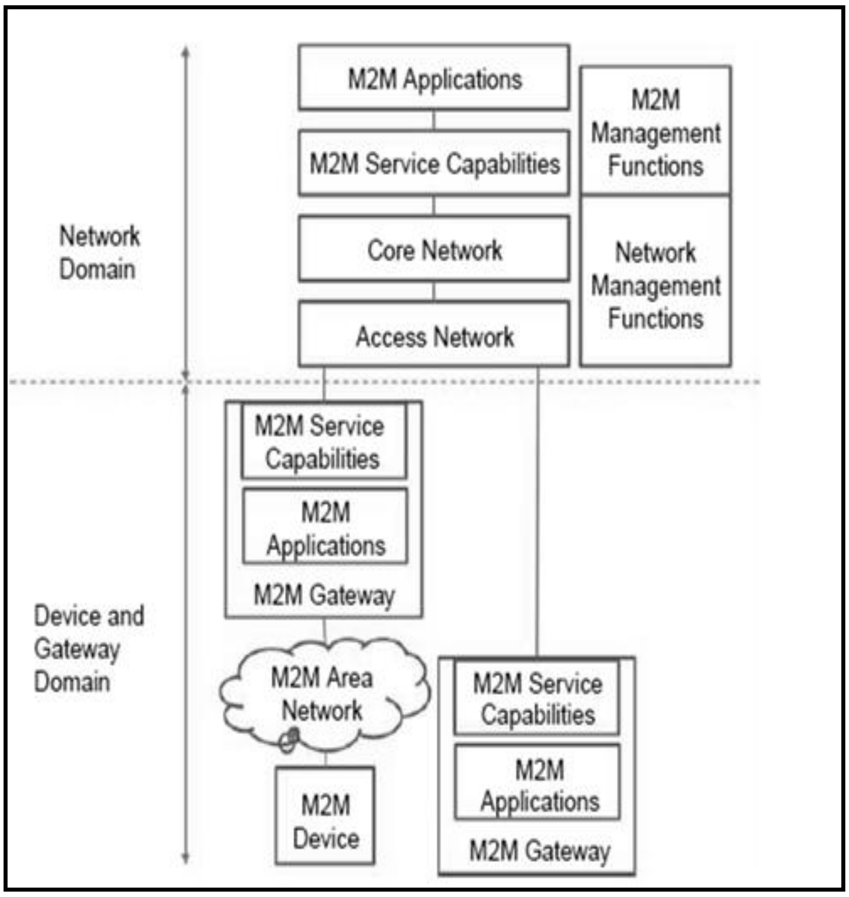

Figure 4.1: ETSI M2M High Level Architecture

The Device and Gateway Domain contains the following functional/topological entities:

• **M2M Device:** This is the device of interest for an M2M scenario, for example, a device with a temperature sensor. An M2M Device contains M2M Applications and M2M Service Capabilities. An M2M device connects to the Network Domain either directly or through an M2M Gateway:

- • Direct connection: The M2M Device is capable of performing registration, authentication, authorization, management, and provisioning to the Network Domain. Direct connection also means that the M2M device contains the appropriate physical layer to be able to communicate with the Access Network.

Through one or more M2M Gateway: This is the case when the M2M device does not have the appropriate physical layer, compatible with the Access Network technology, and therefore it needs a network domain proxy. Moreover, a number of M2M devices may form their own local M2M Area Network that typically employs a different networking technology from the Access Network. The M2M Gateway acts as a proxy for the Network Domain and performs the procedures of authentication, authorization, management, and provisioning. An M2M Device could connect through multiple M2M Gateways.

• **M2M Area Network:** This is typically a local area network (LAN) or a Personal Area Network (PAN) and provides connectivity between M2M Devices and M2M Gateways. Typical networking technologies are IEEE 802.15.1 (Bluetooth), IEEE 802.15.4 (ZigBee, IETF 6LoWPAN/ROLL/CoRE), MBUS, KNX (wired or wireless) PLC, etc.

• **M2M Gateway:** The device that provides connectivity for M2M Devices in an M2M Area Network towards the Network Domain. The M2M Gateway contains M2M Applications and M2M Service Capabilities. The M2M Gateway may also provide services to other legacy devices that are not visible to the Network Domain. The Network Domain contains the following functional/topological entities:

• **Access Network**: this is the network that allows the devices in the Device and Gateway Domain to communicate with the Core Network. Example Access Network Technologies are fixed (xDSL, HFC) and wireless (Satellite, GERAN, UTRAN, E-UTRAN W-LAN, WiMAX).

• **Core Network:** Examples of Core Networks are 3GPP Core Network and ETSI TISPAN Core Network. It provides the following functions:

- IP connectivity.
- Service and Network control.
- Interconnection with other networks.
- Roaming.

• **M2M Service Capabilities:** These are functions exposed to different M2M Applications through a set of open interfaces. These functions use underlying Core Network functions, and their objective is to abstract the network functions for the sake of simpler applications. More details about the specific service capabilities are provided later in the chapter.

• **M2M Applications:** These are the specific M2M applications (e.g. smart metering) that utilize the M2M Service Capabilities through the open interfaces.

• **Network Management Functions:** These are all the necessary functions to manage the Access and Core Network (e.g. Provisioning, Fault Management, etc.).

• **M2M Management Functions:** These are the necessary functions required to manage the M2M Service Capabilities on the Network Domain while the management of an M2M Device or Gateway is performed by specific M2M Service Capabilities. There are two M2M Management functions:

• **M2M Service Bootstrap Function (MSBF):** The MSBF facilitates the bootstrapping of permanent M2M service layer security credentials in the M2M Device or Gateway and the M2M Service Capabilities in the Network Domain. In the Network Service Capabilities Layer, the Bootstrap procedures perform, among other procedures, provisioning of an M2M Root Key (secret key) to the M2M Device or Gateway and the M2M Authentication Server (MAS).

• **M2M Authentication Server (MAS):** This is the safe execution environment where permanent security credentials such as the M2M Root Key are stored. Any security credentials established on the M2M Device or Gateway are stored in a secure environment such as a trusted platform module.

ETSI M2M functional architecture is that it focuses on the high-level specification of functionalities within the M2M Service Capabilities functional groups and the open interfaces between the most relevant entities, while avoiding specifying in detail the internals of M2M Service Capabilities.

The interfaces are specified in different levels of detail, from abstract to a specific mapping of an interface to a specific protocol (e.g. HTTP (Fielding 2000),IETF CoAP The most relevant entities in the ETSI M2M architecture are the M2M Nodes and M2M Applications. An M2M Node can be a Device M2M, Gateway M2M, or Network M2M Node Figure 4.2

An M2M Node is a logical representation of the functions on an M2M Device, Gateway, and Network that should at least include a Service Capability Layer (SCL) functional group.

An M2M Application is the main application logic that uses the Service Capabilities to achieve the M2M system requirements. The application logic can be deployed on a Device (Device Application, DA), Gateway (Gateway Application, GA) or Network (Network Application, NA).

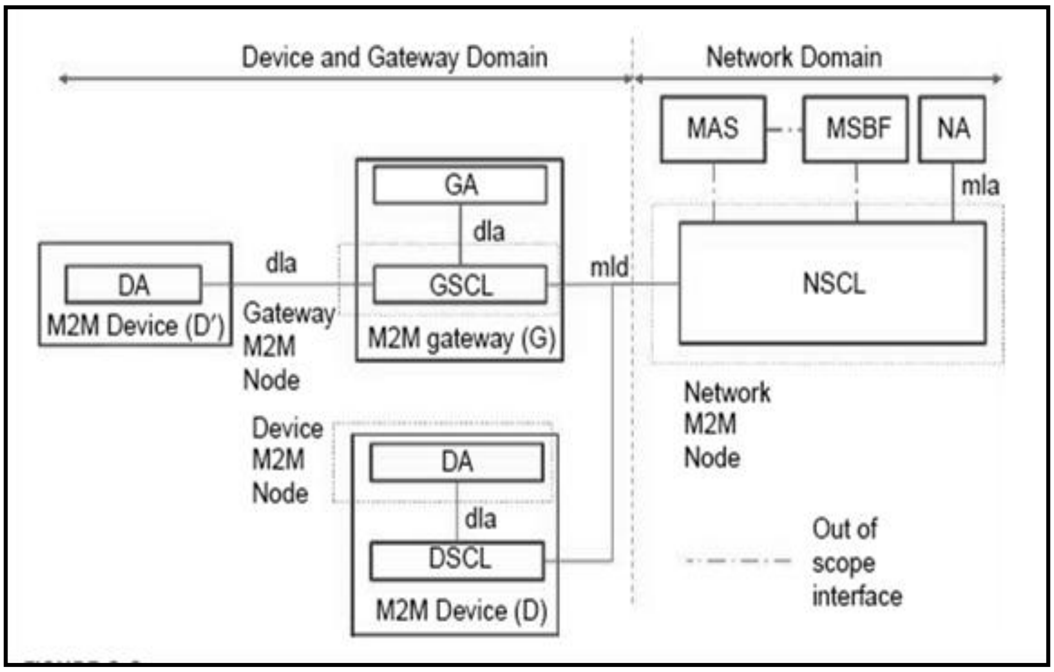

Figure 4.2: M2M service Capabilities, M2M Nodes and Open Interfaces

The SCL is a collection of functions that are exposed through the open interfaces or reference points mIa, dIa, and mId (ETSI M2M TC 2013b).Because the main topological entities that SCL can deploy are the Device,Gateway, and Network Domain, there are three types of SCL: DSCL (Device Service Capabilities Layer), GSCL (Gateway Service Capabilities Layer), and NSCL (Network Service Capabilities Layer).

SCL functions utilize underlying networking capabilities through technology-specific interfaces. For example, an NSCL using a 3GPP type of access network uses 3GPP communication services interfaces.

The ETSI M2M Service Capabilities are recommendations of functional groups for building SCLs,but their implementation is not mandatory, while the implementation of the interfaces mIa, dIa, and mId is mandatory for a compliant system.

4.2.1.2 ETSI M2M service capabilities

All the possible Service Capabilities (where "x" is Network, Gateway, and Device are shown in Figure 4.3:

1. Application Enablement (xAE). The xAE service capability is an application facing functionality and typically provides the implementation of the respective interface: NAE implements the mIa interface and the GAE and DAE implement the dIa interface.The xAE includes registration of applications (xA) to the respective xSCL; for example, a Network Application towards the NSCL.

2. **Generic Communication (xGC).** The NGC is the single point of contact for communication towards the GSCL and DSCL. It provides transport session establishment and negotiation of security mechanisms, potentially secure transmission of messages, and reporting of errors such as transmission errors. The GSC/DSC is the single point of contact for communication with the NSCL, and they both perform similar operations to the NGC (e.g. secure message transmissions to NSCL). The GSC performs a few more functions such as relaying of messages to/from NSCL from/to other SCs in the GSCL, and handles name resolution for the requests within the M2M Area Network.
3. **Reachability, Addressing, and Repository (xRAR).**

   This is one of the main service capabilities of the ETSI M2M architecture. The NRAR hosts mappings of M2M Device and Gateway names to reachability information, and scheduling information relating to reachability, such as whether an M2M Device is reachable between 10 and 11 o'clock.

   - It provides group management (creation/update/deletion) for groups of M2M Devices and Gateways,stores application (DA, GA, NA) data, and manages subscriptions to these data, stores registration information for NA, GSCL, and DSCL,and manages events (subscription notifications).
   - The GRAR provides similar functionality to the NRAR, such as maintaining mappings of the names of M2M Devices or groups to reachability information (routable addresses, reachability status, and reachability scheduling),storing DA, GA, NSCL registration information, storing DA, GA, NA,GSCL, NSCL data and managing subscriptions about them, managing groups of M2M Devices, and managing events. Similar to NRAR and GRAR, the DRAR stores DA, GA, NA, DSCL, and NSCL data and manages subscriptions about these data, stores DA registration and NSCL information, provides group management for groups of M2M Devices and event management.
4. **Communication Selection (xCS):** This capability allows each xSCL to select the best possible communication network when there is more than one choice or when the current choice becomes unavailable due to communication errors. The NCS provides

such a selection mechanism based on policies for reaching an M2M Device or Gateway, while the GCS/DCS provides a similar selection mechanism for reaching the NSCL.

5. **Remote Entity Management (xREM)**
   - The NREM provides management capabilities such as Configuration Management (CM) for M2M Devices and Gateways (e.g. installs management objects in device and gateways), collects performance management (PM) and Fault Management (FM) data and provides them to NAs or M2M Management Functions, performs device management to M2M Devices and Gateways such as firmware and software (application,SCL software) updates, device configuration, and M2M Area Network configuration.
   - The GREM acts as a management client for performing management operations to devices using the DREM and a remote proxy for NREM to perform management operations to M2M Devices in the M2M Area Network. Examples of proxy operations are mediation of NREM-initiated software updates, and handling management data flows from NREM to sleeping M2M Devices.The DREM provides the CM, PM, and FM counterpart on the device (e.g. start collecting radio link performance data) and provides the device-side software and firmware update support.

6. SECurity (xSEC). These capabilities provide security mechanisms such as M2M Service Bootstrap, key management, mutual authentication,and key agreement (NSEC performs mutual authentication and key agreement while the GSEC and DESC initiate the procedures), and potential platform integrity mechanisms.

7. History and Data Retention (xHDR). The xHDR capabilities are optional capabilities, in other words, they are deployed when required by operator policies. These capabilities provide data retention support to other xSCL capabilities (which data to retain) as well as messages exchanged over the respective reference points.

8. Transaction Management (xTM). This set of capabilities is optional and provides support for atomic transactions of multiple operations.

An atomic transaction involves three steps:

(a) propagation of a request to a number of recipients

(b) collection of responses, and

(c) commitment or roll back whether all the transactions successfully completed or not.

9. Compensation Broker (xCB). This capability is optional and provides support for brokering M2M-related requests and compensation between a Customer and a Service Provider. In this context a Customer and a Service Provider is an M2M Application.

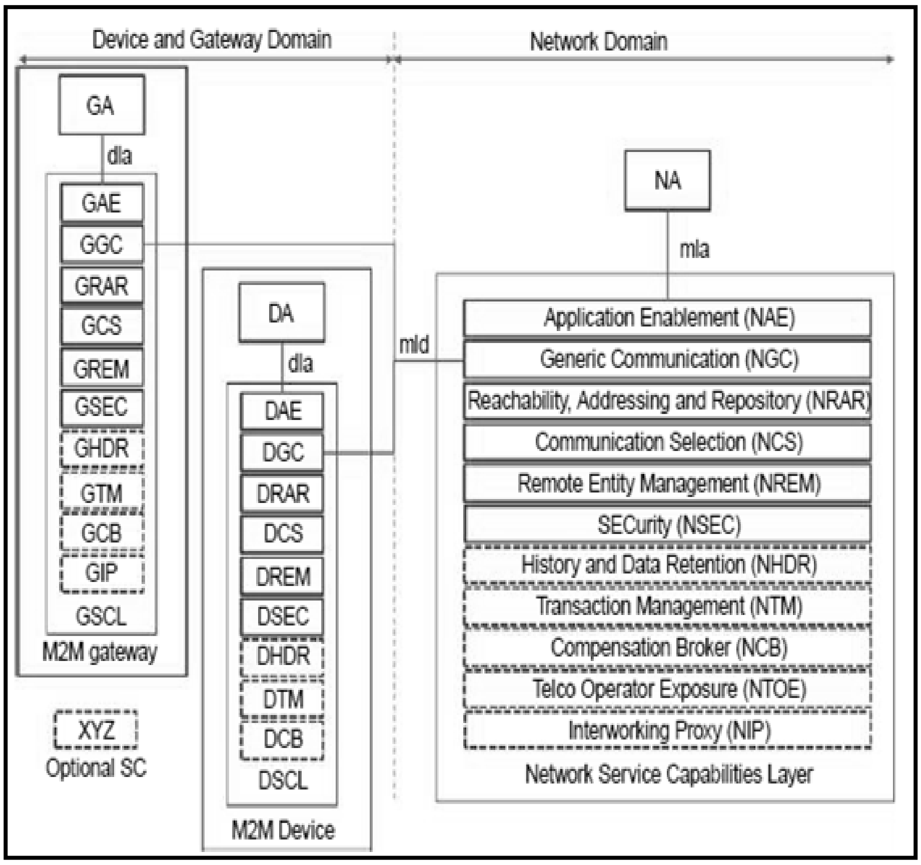
Device and Gateway Domain
Network Domain
GA
dIa
GAE
GGC
GRAR
GCS
GREM
GSEC
GHDR
GTM
GCB
GIP
GSCL
M2M gateway
XYZ
Optional SC
DA
dIa
DAE
DGC
DRAR
DCS
DREM
DSEC
DHDR
DTM
DCB
DSCL
M2M Device
mId
NA
mIa
Application Enablement (NAE)
Generic Communication (NGC)
Reachability, Addressing and Repository (NRAR)
Communication Selection (NCS)
Remote Entity Management (NREM)
SECurity (NSEC)
History and Data Retention (NHDR)
Transaction Management (NTM)
Compensation Broker (NCB)
Telco Operator Exposure (NTOE)
Interworking Proxy (NIP)
Network Service Capabilities Layer

Figure 4.3: M2M Capabilities for different M2M Nodes.

10.Telco Operator Exposure (NTOE). This is also an optional capability and provides exposure of the Core Network service offered by a Telecom Network Operator.

11. Interworking Proxy (xIP). This capability is an optional capability and provides mechanisms for connecting non-ETSI M2M Devices and Gateways to ETSI SCLs. NIP provides mechanisms for non-ETSI M2M Devices and Gateways to connect to NSCL while GIP provides the functionality for non-compliant M2M Devices to connect to GSCL via the reference point dIa, and the DIP provides the necessary mechanisms to connect non-compliant devices to DSCL via the dIa reference point.

4.2.1.3 ETSI M2M interfaces

The main interfaces mIa, dIa, and mId (ETSI M2M TC 2013b) can be briefly described as follows:

• **mIa:** This is the interface between a Network Application and the Network Service Capabilities Layer (NSCL). The procedures supported by this interface are (among others) registration of a Network Application to the NSCL, request to read/write information to NSCL, GSCL, or DSCL, request for device management actions (e.g. software updates), subscription and notification of specific events.

• **dIa:** This is the interface between a Device Application and (D/G)SCL or a Gateway Application and the GSCL. The procedures supported by this interface are (among others) registration of a Device/Gateway Application to the GSCL, registration of a Device Application to the DSCL, request to read/write information to NSCL, GSCL, or DSCL, subscription and notification of specific events.

• **mId:** This is the interface between the Network Service Capabilities Layer (NSCL) and the GSCL or the DSCL. The procedures supported by this interface are (among others) registration of a Device/Gateway SCL to the NSCL, request to read/write information to NSCL, GSCL,or DSCL, subscription and notification of specific events.

4.2.1.4 ETSI M2M resource management

The ETSI M2M architecture assumes that applications (DA, GA, NA) exchange information with SCLs by performing CRUD (Create, Read,Update, Delete) operations on a number of Resources following the RESTful (Representational State Transfer) architecture paradigm. One of the principles of this paradigm is that representations of uniquely addressed resources are transferred from the entity that hosts these resources to the requesting entity. In the ETSI M2M architecture, all the state information maintained in the SCLs is modeled as a resource structure that architectural entities operate on. A very simplified view of the resource structure is a collection of containers of information structured in hierarchical manner following a corresponding hierarchy of a unique naming structure.

In addition to the CRUD operations, ETSI M2M defines two more operations: NOTIFY and EXECUTE. The NOTIFY operation is triggered upon a change in the representation of a resource, and results in a notification sent to the entity that originally subscribed to monitor changes to the resource in question. This operation is not an orthogonal operation to the CRUD set, but can be implemented by an UPDATE operation from the resource host towards the requesting entity.

The EXECUTE operation is not orthogonal as well, but can be implemented by an UPDATE operation with no parameters from the requesting entity to a specific resource.

When a requesting entity issues an EXECUTE operation towards a specific resource, the specific resource executes a specific task.

The following example in Figure 4.4 demonstrates how an ETSI M2M entity communicates with another entity using the CRUD and NOTIFIES operations. Assume that a device application (DA) is programmed to send a sensor measurement to a network application (NA). The DA using the DSCL updates the representation of a specific resource (Ra) residing on the NSCL (steps 1 and 2 in Figure 4.4). The NA has configured the NSCL to be notified when the specific resource is updated, in which case the NA reads the updated representation (steps 4 and 5, Figure 4.4).

The root of the hierarchical resource tree is the ,sclBase. resource, and contains all the other resources hosted by the SCL. The root has a unique identifier. In case the RESTful architecture is implemented in a real system by using web resources, the ,sclBase. has an absolute URI (Universal Resource Identifier), for example, “http://m2m.operator1.com/ some/path/to/base”. The top-level structure of the ,sclBase. resource is shown in Figure 4.5. The different fonts used in this figure 4.5 denote

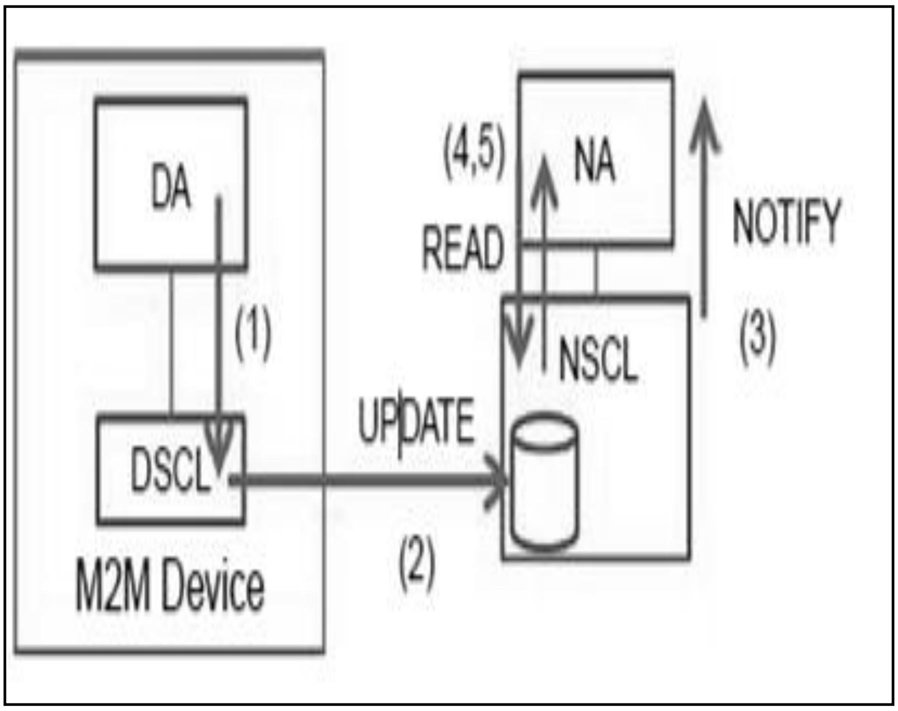

Figure 4.4 : Communication Between DA and NA Using the SCLs.

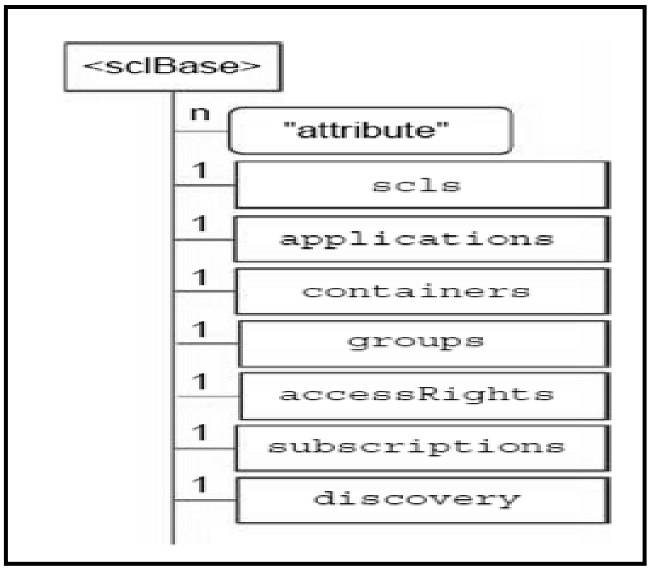

Figure 4.5: The Top-Level Structure of the, sclBase Resource.

different information semantics. A term between the symbols ",” and “.” denotes an arbitrary resource name; for example, ,sclBase. In **Figure 4.5.** A term within quotes (“”) denotes a placeholder for one or more fixed names. In this specific case, “attribute” represents a member of a fixed list of attributes for the resource ,sclBase.. A term annotated in Courier New font such as scls denotes a literal resource name used by the specification as is. The ,sclBase. is structured as a tree with different branches, each of which is annotated with a designation of its cardinality. For example, the ,sclBase. contains n attributes, one scls resource, one applications resource, etc.

4.2.2 International Telecommunication Union Telecommunication sector view

The Telecommunication sector of the International Telecommunication Union (ITU-T) has been active on IoT standardization since 2005 with the Joint Coordination Activity on Network Aspects of Identification Systems (JCA-NID), which was renamed to Joint Coordination Activity on IoT (JCA-IoT) in 2011. During the same year apart from this coordination activity on IoT, ITU-T formed the specific IoT Global Standards Initiative (IoT-GSI) activity in order to address specific IoT-related issues. The latest ITU-T Recommendation, Y.2060 (ITU-T 2013) (Vermesan & Friess 2013), provides an overview of the IoT space with respect to ITU-T. This recommendation describes a high-level overview of the IoT domain model and the IoT functional model as a set of Service Capabilities similar to ETSI-M2M.

The ITU-T IoT domain model includes a set of physical devices that connect directly or through gateway devices to a communication network that allows them to exchange information with other devices, services, and applications.

ITU-T formed the specific IoT Global Standards Initiative (IoT-GSI) activity in order to address specific IoT-related issues. The latest ITU-T Recommendation, Y.2060 (ITU-T 2013) (Vermesan & Friess 2013), provides an overview of the IoT space with respect to ITU-T. This recommendation describes a high-level overview of the IoT domain model and the IoT functional model as a set of Service Capabilities similar to ETSI-M2M.

The ITU-T IoT domain model includes a set of physical devices that connect directly or through gateway devices to a communication network that allows them to exchange information with other devices, services, and applications.

If the ITU-T and ETSI M2M models/architectures are compared, the two approaches are very similar in terms of Service Capabilities for M2M.However, the ITU-T IoT domain model with physical and virtual things,and the physical and virtual world model, shows the influences of more modern IoT architectural models and references such as the IoT-A.

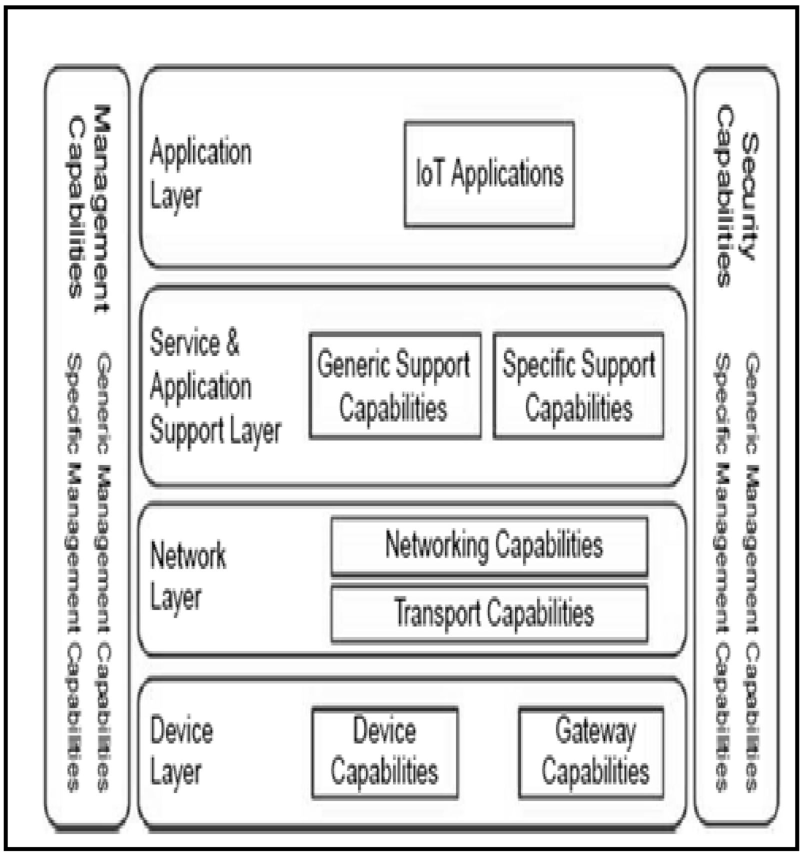
Management Capabilities
Generic Management Capabilities
Specific Management Capabilities
Application Layer
IoT Applications
Service & Application Support Layer
Generic Support Capabilities
Specific Support Capabilities
Network Layer
Networking Capabilities
Transport Capabilities
Device Layer
Device Capabilities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

Figure 4.6: ITU-T IoT Reference Model.

4.2.3 Internet Engineering Task Force architecture fragments

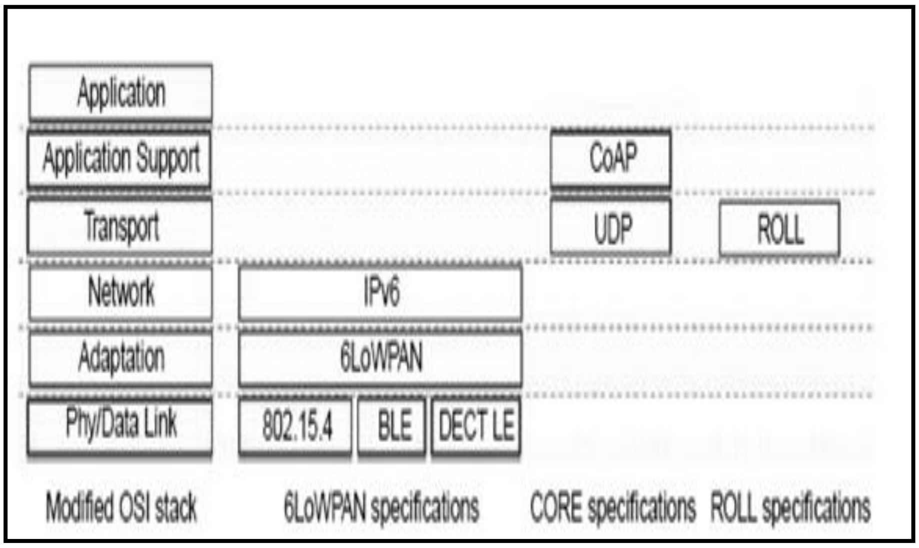

Figure 4.7: IETF Working Groups and Specification Scope

The modified Open Systems Interconnection (OSI) model in which several layers are merged because of the implementation on constrained devices.This is illustrated in Figure 4.7 as one layer called Application Support which includes the Presentation and Session Layers combined.

Moreover, one intermediate layer is introduced: the Adaptation Layer positioned between the Physical/Data Link and the Network Layer and whose main function is to adapt the Network Layer packets to Phy/Link layer packets among others.

An example of an adaptation layer is the 6LoWPAN layer designed to adapt IPv6 packets to IEEE 8021.5.4/Bluetooth Low Energy (BLE)/DECT Low Energy packets. An example of an Application Support Layer is IETF Constrained Application Protocol (CoAP), which provides reliability and RESTful operation support to applications.

Apart from the core of the specifications, the IETF CoRE workgroup includes several other draft specifications that sketch parts of an architecture for IoT. The CoRE Link Format specification describes a discovery method for the CoAP resources of a CoAP server.

The IETF CoRE working group has also produced a draft specification for a Resource Directory. A Resource Directory is a CoAP server resource (/rd) that maintains a list of resources, their corresponding server contact information (e.g. IP addresses or fully qualified domain name, or FQDN), their type, interface, and other information similar to the information that the CoRE Link Format document specifies (Figure 4.8a).

An RD plays the role of a rendezvous mechanism for CoAP Server resource descriptions, in other words, for devices to publish the descriptions of the available resources and for CoAP clients to locate resources that satisfy certain criteria such as specific resource types.

A Mirror Server (Vial 2012) is a rendezvous mechanism for CoAP Server resource presentations. A Mirror Server is a CoAP Server resource (/ms) that maintains a list of resources and their cached representations (Figure 4.8b). A CoAP Server registers its resources to the Mirror Server, and upon registration a new mirror server resource is created on the Mirror Server with a container (mirror representation) for the original server representation. The original CoAP Server updates the mirror representation either periodically or when the representation changes. A CoAP Client that retrieves the mirror representation receives the latest updated representation from the original CoAP Server.
The Mirror Server is useful when the CoAP Server is not always available for direct access.

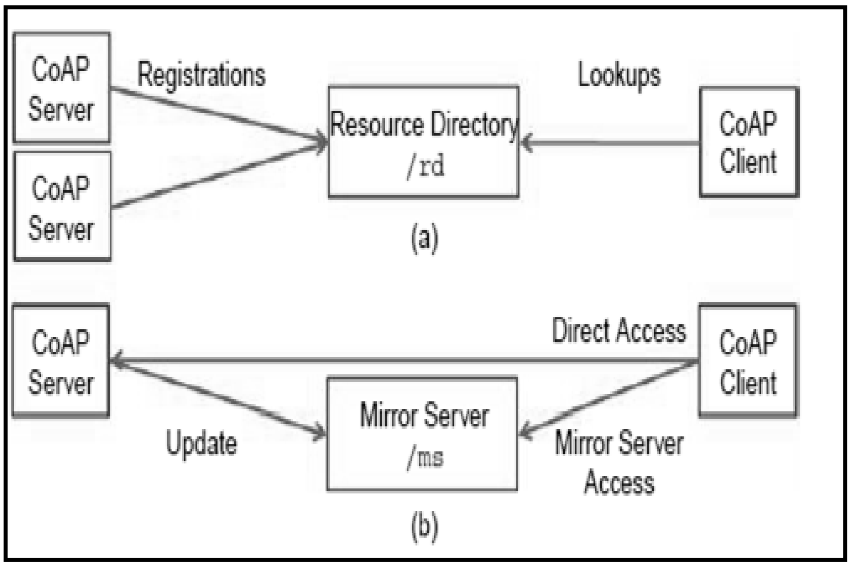

Figure 4.8: IETF CoRE Functional Components a) Resource Directory b) Mirror Server

The figure 4.9 The interworking issues appear when an HTTP Client accesses a CoAP Server through an HTTP-CoAP proxy or when a CoAP Client accesses an HTTP Server through a CoAP-HTTP proxy (Figure 4.9a).
The mapping process is not straightforward for a number of reasons. The main is the different transport protocols used by the HTTP and CoAP: HTTP uses TCP while CoAP uses UDP. The guidelines focus more on the HTTP-to-CoAP proxy and recommend addressing schemes (e.g. how to map a CoAP resource address to an HTTP address),mapping between HTTP and

CoAP response codes, mapping between different media types carried in the HTTP/CoAP payloads, etc.

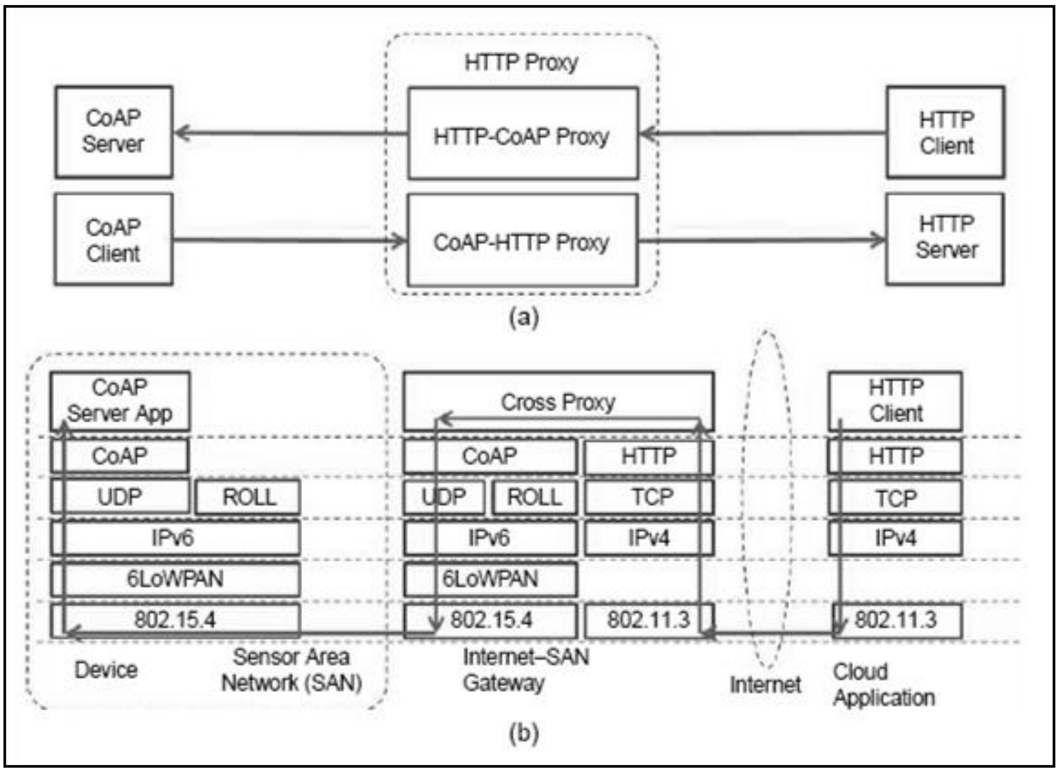

Figure 4.9: IETF CoRE HTTP Proxy

a) Possible configurations b) example layer interaction upon a request from HTTP client to a CoAP server via a HTTP Proxy.

- As an example, consider the case that an HTTP Client sends an HTTP request to a CoAP server (Figure 4.9a) through a Gateway Device hosting an HTTP-CoAP Cross Proxy.
- The Gateway Device connects to the Internet via an Ethernet cable using a LAN, and on the CoAP side the CoAP server resides on a Sensor/Actuator (SAN) based on the IEEE 802.15.4 PHY/MAC.
- The HTTP request needs to include two addresses, one for reaching the Cross Proxy and one for reaching the specific CoAP Server in the SAN. The default recommended address mapping is to append the CoAP resource address (e.g. coap://s.coap.example.com/foo)to the Cross proxy address (e.g. http://p.example.com/.well-known/core/), resulting in http://p.example.com/.well-known/core/coap://s.coap.example.com/foo.

- The request is in plain text format and contains the method (GET). It traverses the IPv4 stack of the client, reaches the gateway, traverses the IPv4 stack of the gateway and reaches the Cross proxy.
- The request is translated to a CoAP request (binary format) with a destination CoAP resource coap://s.coap.example.com/foo, and it is dispatched in the CoAP stack of the gateway, which sends it over the SAN to the end device.A response is sent from the end device and follows the reverse path in the SAN order to reach the gateway.

4.2.4 Open Geospatial Consortium architecture

The Open Geospatial Consortium (OGC 2013) is an international industry consortium of a few hundred companies, government agencies, and universities that develops publicly available standards that provide geographical information support to the Web, and wireless and location-based services.

The functionality that is targeted by OGC SWE includes:

• Discovery of sensor systems and observations that meet an application's criteria.

• Discovery of a sensor's capabilities and quality of measurements.

• Retrieval of real-time or time-series observations in standard encodings.

• Tasking of sensors to acquire observations.

• Subscription to, and publishing of, alerts to be issued by sensors or sensor services based upon certain criteria.

OGC SWE includes the following standards:

- SensorML and Transducer Model Language (TML), which include a model and an XML schema for describing sensor and actuator systems and processes; for example, a system that contains a temperature sensor measuring temperature in Celsius, which also involves a process for converting this measurement to a measurement with Fahrenheit units.
- Observations and Measurements (O&M), which is a model and an XML schema for describing the observations and measurements for a sensor (Observations and Measurements, O&M).
- SWE Common Data model for describing low-level data models (e.g. serialization in XML) in the messages exchanged between OGC SWE functional entities.

- Sensor Observation Service (SOS), which is a service for requesting,filtering, and retrieving observations and sensor system information.This is the intermediary between a client and an observation repository or near real-time sensor channel.
- Sensor Planning Service (SPS), which is a service for applications requesting a user-defined sensor observations and measurements acquisition. This is the intermediary between the application and a sensor collection system.
- PUCK, which defines a protocol for retrieving sensor metadata for serial port (RS232) or Ethernet-enabled sensor devices.
- An example of how these standards relate to each other is shown in Figure 4.10. Because OGC follows the SOA paradigm, there is a registry (CAT) that maintains the descriptions of the existing OGC services ,including the Sensor Observation and Sensor Planning Services. Upon installation the sensor system using the PUCK protocol retrieves the SensorML description of sensors and processes, and registers them with the Catalog so as to enable the discovery of the sensors and processes by client applications. The Sensor System also registers to the SOS and the SOS registers to the Catalog. A client application #1 requests from the Sensor Planning Service that the Sensor System be tasked to sample its

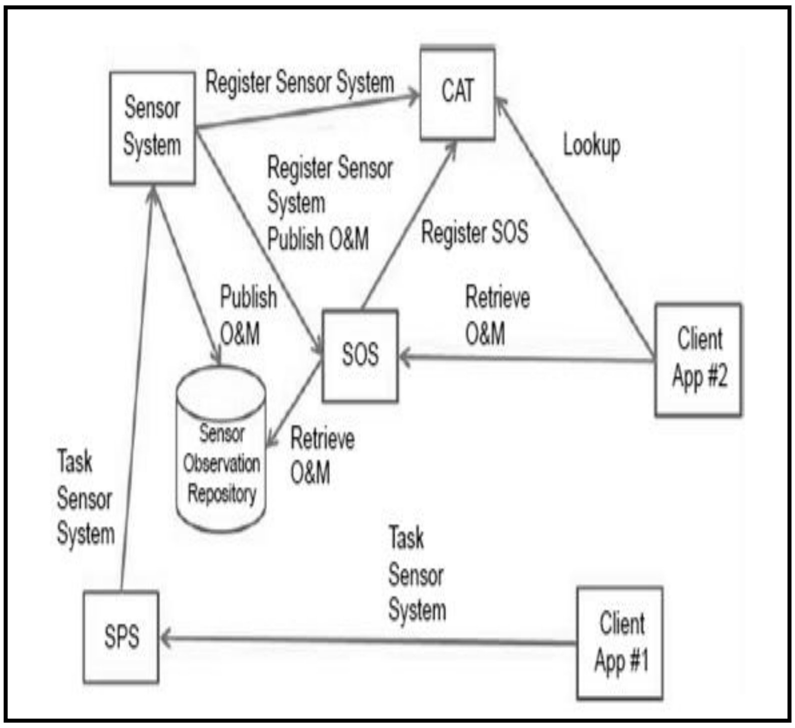

Figure 4.10: OGC functional architecture and interactions.

4.3 Introduction

This chapter provides an overview of the Architecture Reference Model (ARM) for IoT, including descriptions of the domain, information, and functional models. Chapter 8 then outlines the Reference Architecture.

An ARM consists of two main parts: a Reference model and Reference architecture. The foundation of an IoT Reference Architecture description is an IoT reference model. A reference model describes the domain using a number of sub-models (**Figure 4.11**). The domain model of an architecture model captures the main concepts or entities in the domain in question, in this case M2M and IoT. When these common language references are established , the domain model adds descriptions about the relationship between the concepts.

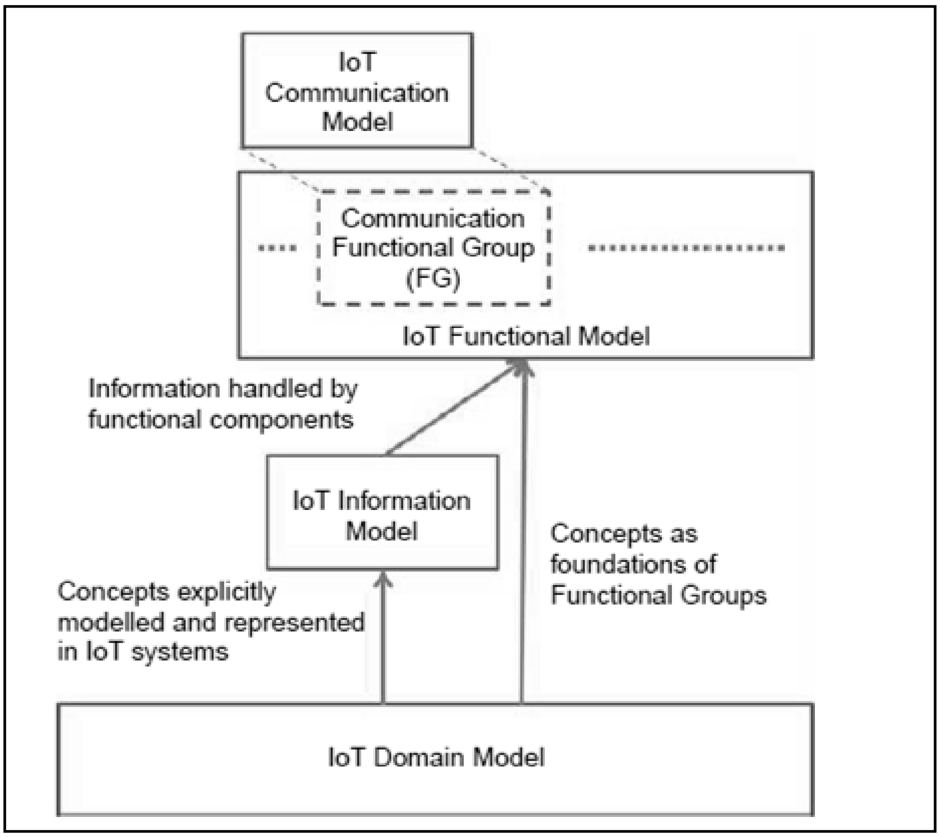
IoT Communication Model
Communication Functional Group (FG)
IoT Functional Model
Information handled by functional components
IoT Information Model
Concepts as foundations of Functional Groups
Concepts explicitly modelled and represented in IoT systems
IoT Domain Model

Figure 4.11: IoT Reference Model.

These concepts and relationships serve the basis for the development of an information model because a working system needs to capture and process information about its main entities and their interactions.

A working system that captures and operates on the domain and information model contains concepts and entities of its own, and these needs to be described in a separate model, the functional model. An M2M and IoT system contain communicating entities, and therefore the corresponding communication model needs to capture the communication interactions of these entities. These are a few examples of sub-models that we use in this chapter for the IoT reference model.

Apart from the reference model, the other main component of an ARM is the Reference Architecture. A System Architecture is a communication tool for different stakeholders of the system. Developers, component and system managers, partners, suppliers, and customers have different views of a single system based on their requirements and their specific interactions with the system.

The task becomes more complex when the architecture to be described is on a higher level of abstraction compared with the architecture of real functioning systems. The high-level abstraction is called Reference Architecture as it serves as a reference for generating concrete architectures and actual systems, as shown in the Figure 4.12

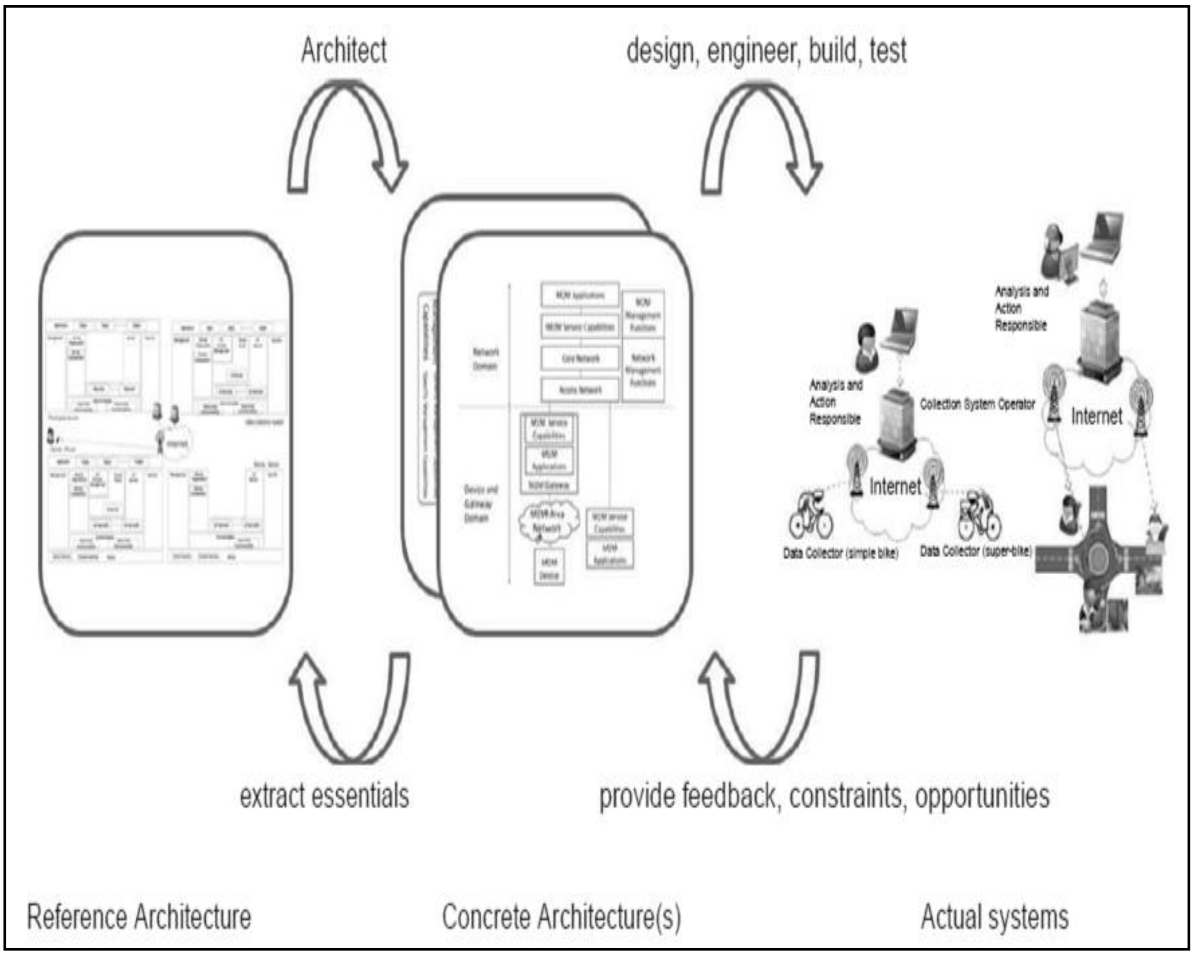
Architect
design, engineer, build, test
extract essentials
provide feedback, constraints, opportunities
Reference Architecture
Concrete Architecture(s)
Actual systems
Analysis and Action Responsible
Collection System Operator
Internet
Data Collector (simple bike)
Data Collector (super-bike)

Figure 4.12: From reference to concrete architectures and actual systems.

- Concrete architectures are instantiations of rather abstract and high-level Reference Architectures.
- A Reference Architecture captures the essential parts of an architecture, such as design principles, guidelines, and required parts (such as entities), to monitor and interact with the physical world for the case of an IoT Reference Architecture.
- A concrete architecture can be further elaborated and mapped into real world components by designing, building, engineering, and testing the different components of the actual system. As the figure implies, the whole process is iterative, which means that the actual deployed system in the field provides invaluable feedback with respect to the design and engineering choices, current constraints of the system, and potential future opportunities that are fed back to the concrete architectures. The general essentials out of multiple concrete architectures can then be aggregated, and contribute to the evolution of the Reference Architecture.
- The IoT architecture model is related to the IoT Reference Architecture as shown in Figure 4.13. This figure shows two facets of the IoT ARM: (a)how to actually create an IoT ARM, and (b) how to use it with respect to building actual systems.

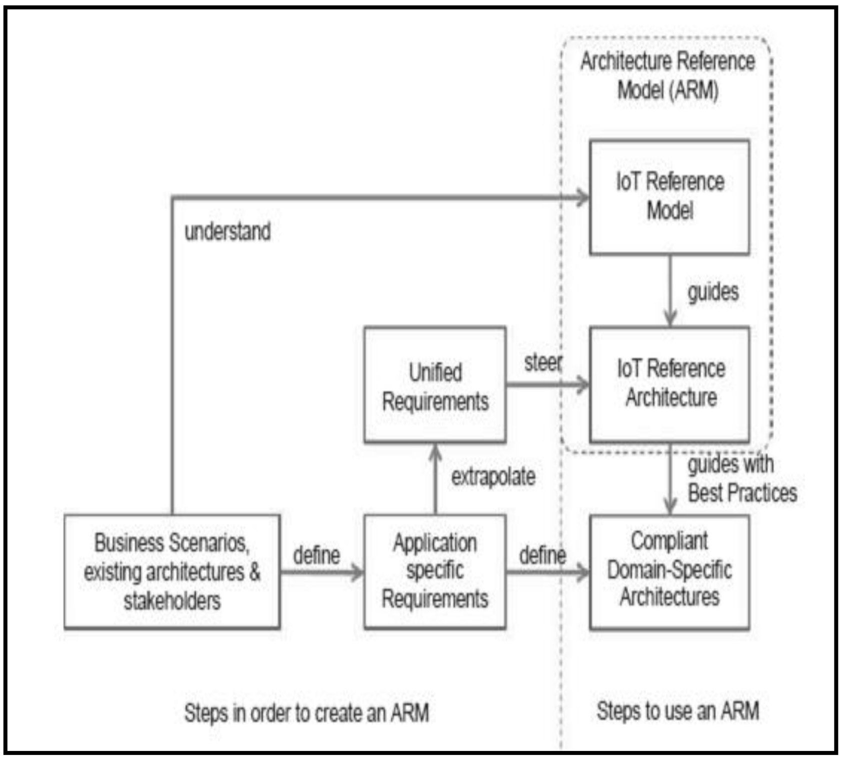
Architecture Reference Model (ARM)
IoT Reference Model
understand
guides
Unified Requirements
steer
IoT Reference Architecture
extrapolate
guides with Best Practices
Business Scenarios, existing architectures & stakeholders
define
Application specific Requirements
define
Compliant Domain-Specific Architectures
Steps in order to create an ARM
Steps to use an ARM

Figure 4.13: IoT Reference Model and Reference Architecture dependencies.

4.4 IOT REFERENCE MODEL

4.4.1 IOT DOMAIN MODEL

- The domain model captures the basic attributes of the main concepts and the relationship between these concepts.
- A domain model also serves as a tool for human communication between people working in the domain in question and between people who work across different domains.
- A domain model also serves as a tool for human communication between people working in the domain in question and between people who work across different domains.

4.4.1.1 Model notation and semantics

- For the purposes of the description of the domain model, we use the Unified Modeling Language (UML).Class diagrams in order to present the relationships between the main concepts of the IoT domain model.
- The Class diagrams consist of boxes that represent the different classes of the model connected with each other through typically continuous lines or arrows, which represent relationships between the respective classes.
- Each class is a descriptor of a set of objects that have similar structure, behavior, and relationships.
- A class contains a name (e.g. Class A in Figure 4.14) and a set of attributes and operations.

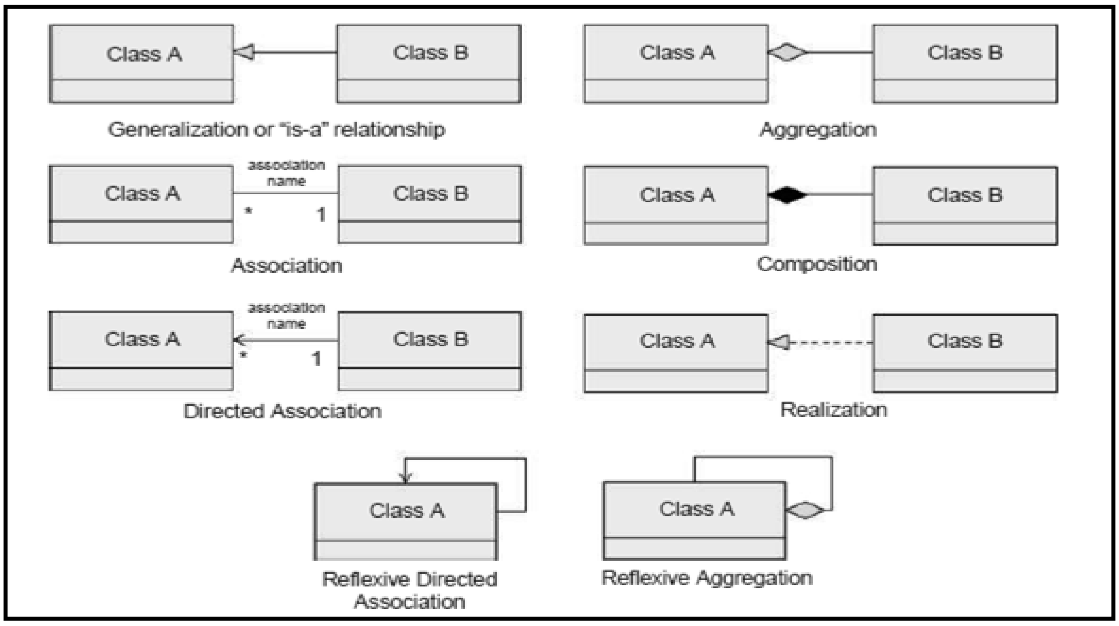

Figure 4.14: UML Class diagram main modeling concepts.

- For the description of the IoT domain model, we will use only the class name and the class attributes, and omit the class operations.
- Notation-wise this is represented as a box with two compartments, one containing the class name and the other containing the attributes. However, for the IoT domain model description, the attribute compartment will be empty in order not to clutter the complete domain model.
- The following modeling relationships between classes (Figure 4.14) are needed for the description of the IoT Domain Model: Generalization/ Specialization, Aggregation and Reflexive Aggregation, Composition, Directed Association and Reflexive Directed Association, and Realization.
- The Generalization/Specialization relationship is represented by an arrow with a solid line and a hollow triangle head. Depending on the starting point of the arrow, the relationship can be viewed as a generalization or specialization.
- The Aggregation relationship is represented by a line with a hollow diamond in one end and represents a whole-part relationship or a containment relationship and is often called a “has-a” relationship. The class that touches the hollow diamond is the whole class while the other class is the part class.
- The Composition relationship is represented by a line with a solid black diamond in one end, and also represents a whole-part relationship or a containment relationship. The class that touches the solid black diamond is the whole class while the other class is the part class.

4.4.1.2 Main concepts

The IoT is a support infrastructure for enabling objects and places in the physical world to have a corresponding representation in the digital world.The reason why we would like to represent the physical world in the digital world is to remotely monitor and interact with the physical world using software.

Let’s illustrate this concept with an example (Figure 4.15).Imagine that we are interested in monitoring a parking lot with 16 parking spots. The parking lot includes a payment station for drivers to pay for the parking spot after they park their cars. The parking lot also includes an electronic road sign on the side of the street that shows in real-time the number of empty spots.

Frequent customers also download a smart phone application that informs them about the availability of a parking spot before they even drive on the street where the parking lot is located. In order to realize such a service, the relevant physical objects as well as their properties need to be captured and translated to digital objects such as variables, counters, or database objects so that software can operate on these objects and achieve the desired effect, i.e. detecting when someone parks without paying, informing drivers about the availability of parking spots, producing statistics about the average occupancy levels of the parking lot, etc.

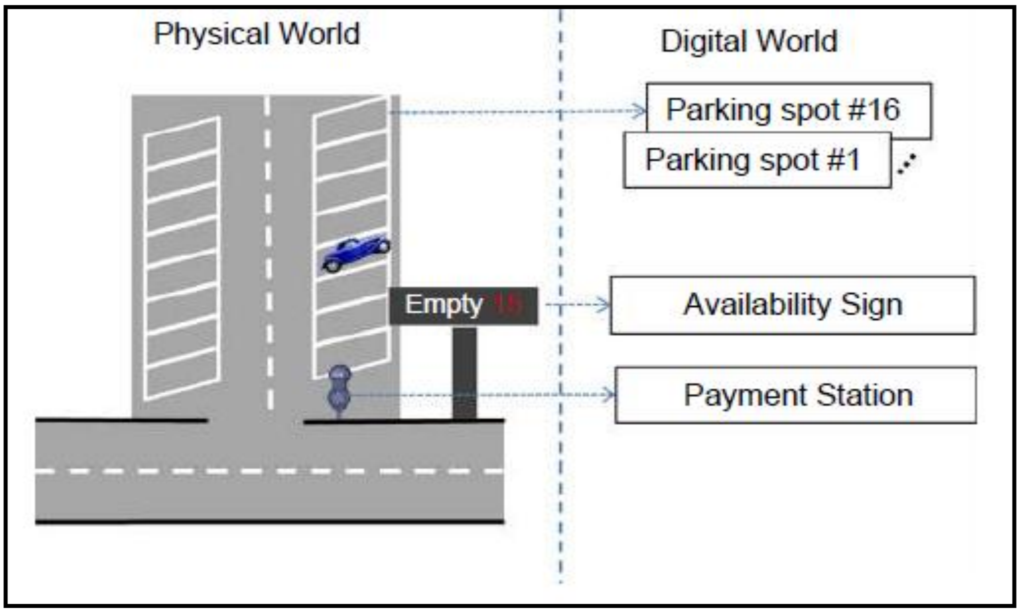

Figure 4.15: Physical vs. Virtual World.

For these purposes, the parking lot as a place is instrumented with parking spot sensors (e.g. loops), and for each sensor, a digital representation is created (Parking spot #1_#16). In the digital world, a parking spot is a variable with a binary value ("available" or "occupied"). The parking lot payment station also needs to be represented in the digital world in order to check if a recently parked car owner actually paid the parking fee. Finally, the availability sign is represented to the digital world in order to allow notification to drivers that an empty lot is full for maintenance purposes, or even to allow maintenance personnel to detect when the sign is malfunctioning.

As interaction with the physical world is the key for the IoT; it needs to be captured in the domain model (Figure 4.16). The first most fundamental interaction is between a human or an application with the physical A User can be a Human User, and the interaction can be physical (e.g. parking the car in the parking lot).

The physical interaction is the result of the intention of the human to achieve a certain goal (e.g. park the car). In other occasions, a Human world object or place. Therefore, a User and a Physical Entity are two concepts that belong to the domain model.

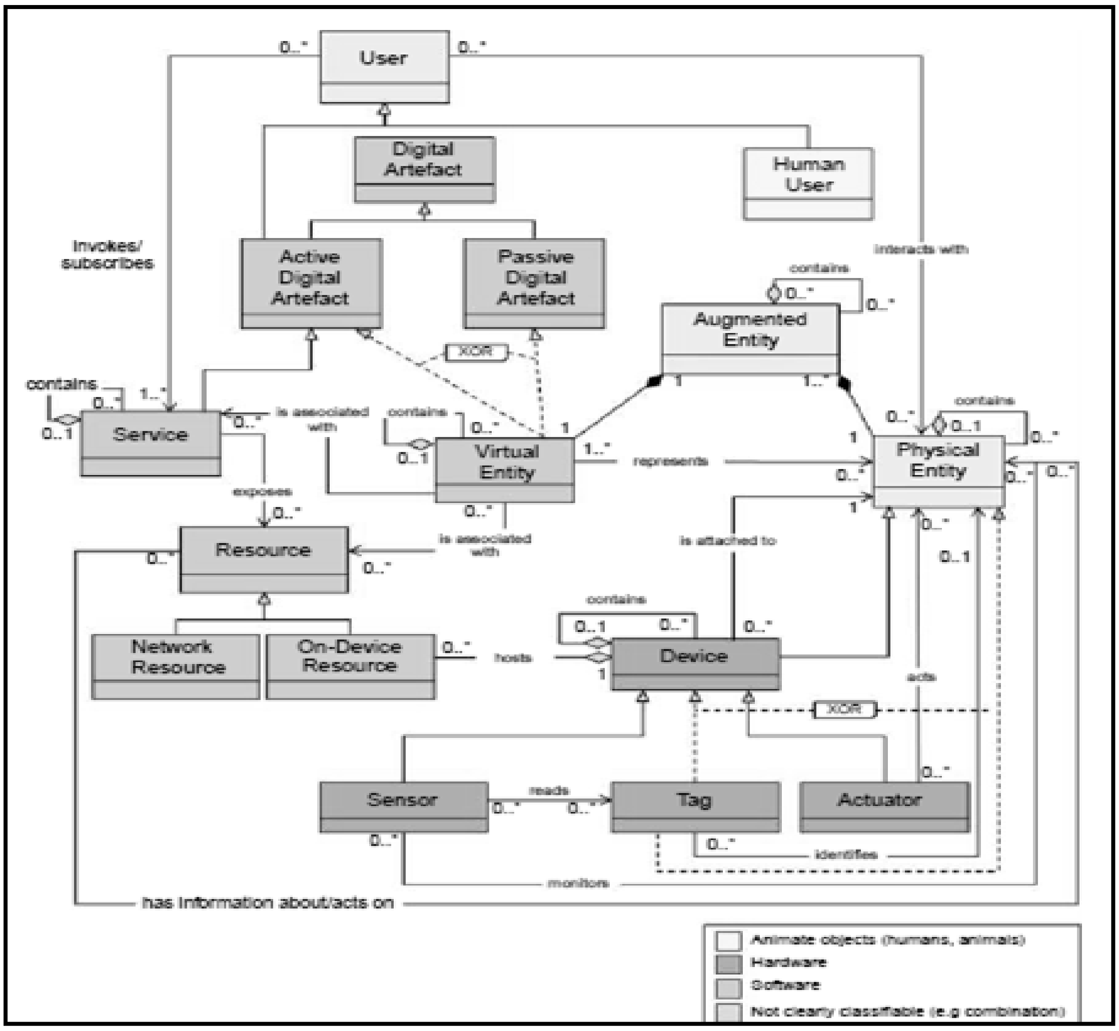

Figure 4.15: IoT Domain Model.

A Physical Entity, as the model shows, can potentially contain other physical entities; for example, a building is made up of several floors, and each floor has several rooms.

The objects, places, and things represented as Physical Entities are the same as Assets mentioned earlier in the book. According to the Oxford Dictionary, an Asset .is an item or property that is regarded as having value”; therefore, the term Asset is more related to the business aspects of IoT. Because the domain model is a technical tool, we use the term Physical Entity instead of Asset.

- A Physical Entity is represented in the digital world as a Virtual Entity.
- A Virtual Entity can be a database entry, a geographical model (mainly for places), an image or avatar, or any other Digital Artifact.
- One Physical Entity can be represented by multiple Virtual Entities, each serving a different purpose, e.g. a database entry of a parking spot denoting the spot

availability,and an (empty/full) image of a parking spot on the monitor of the parking lot management system.

- Each Virtual Entity also has a unique identifier for making it addressable among other Digital Artifacts. A Virtual Entity representation contains several attributes that correspond to the Physical Entity current state.
- The Virtual Entity representation and the Physical Entity actual state should be synchronized whenever a User operates on one or the other, if of course that is physically possible.

For the IoT Domain Model, three kinds of Device types are the most important:

1. Sensors: These are simple or complex Devices that typically involve a transducer that converts physical properties such as temperature into electrical signals. These Devices include the necessary conversion of analog electrical signals into digital signals, e.g. a voltage level to a 16-bit number, processing for simple calculations, potential storage for intermediate results, and potentially communication capabilities to transmit the digital representation of the physical property as well

receive commands. A video camera can be another example of a complex sensor that could detect and recognize people.

2. Actuators: These are also simple or complex Devices that involve a transducer that converts electrical signals to a change in a physical property (e.g. turn on a switch or move a motor). These Devices also include potential communication capabilities, storage of intermediate commands, processing, and conversion of digital signals to analog electrical signals.

3. Tags: Tags in general identify the Physical Entity that they are attached to. In reality, tags can be Devices or Physical Entities but not both, as the domain model shows. An example of a Tag as a Device is a Radio Frequency Identification (RFID) tag, while a tag as a Physical Entity is a paper-printed immutable barcode or Quick Response (QR) code. Either electronic Devices or a paper-printed entity tag contains a unique identification that can be read by optical means (bar codes or QR codes) or radio signals (RFID tags). The reader Device operating on a tag is typically a sensor, and sometimes a sensor and an actuator combined in the case of writable RFID tags.