Department of Artificial Intelligence & Data Science

ANNAMACHARYA INSTITUTE OF TECHNOLOGY AND SCIENCES RAJAMPET
(An Autonomous Institution)

Title of the Course	Internet of Things
Category	PEC
Couse Code	20A307BT

Year	IV B.Tech
Semester	I Semester
Branch	AI&DS

Lecture Hours	Tutorial Hours	Practice Hours	Credits
3	0	0	3

Course Objectives:

- To understand the fundamentals of Internet of Things
- To learn about the IoT architecture and protocols
- To build a small low cost embedded system using Raspberry Pi & Arduino
- To apply the concept of Internet of Things in the real world scenario
- To Analyze applications of IoT in real time scenario

Unit 1 Introduction to IOT **9**

Internet of Things - Physical Design- Logical Design- IoT Enabling Technologies - IoT Levels & Deployment Templates - Domain Specific IoTs - IoT and M2M - IoT System Management with NETCONF-YANG- IoT Platforms Design Methodology.

Learning Outcomes: At the end of the unit, the student will be able to:

- Understand the concepts of Physical design, logical design and IoT enabling techniques. (L2)
- Demonstrate about domain specific IoTs, IoT System Management, IoT Platforms Design Methodology. (L2)

Unit 2 IOT Architecture **9**

M2M high-level ETSI architecture - IETF architecture for IoT - OGC architecture - IoT reference model - Domain model - Information model - Functional model - Communication model - IoT Reference architecture.

Learning Outcomes: At the end of the unit, the student will be able to:

- Understand different types of IoT architectures. (L2)
- Identify differences between different types of IoT models. (L3)

Unit 3 IoT Protocols **9**

Protocol Standardization for IoT – Efforts – M2M and WSN Protocols – SCADA and RFID Protocols – Unified Data Standards – Protocols – IEEE 802.15.4 – BACNet Protocol – Modbus– Zigbee Architecture – Network layer – 6LowPAN - CoAP – Security

Learning Outcomes: At the end of the unit, the student will be able to:

- Analyze different types of protocols used in IoT systems. (L4)
- Acquire knowledge about protocols used in wireless personal area networks. (L2)

Unit 4 Building IoT with Raspberry PI & ARDUINO **9**

Building IOT with RASPBERRY PI- IoT Systems - Logical Design using Python – IoT Physical Devices & Endpoints - IoT Device -Building blocks -Raspberry Pi Board - Linux on Raspberry Pi - Raspberry Pi Interfaces - Programming Raspberry Pi with Python - Other IoT Platforms – Arduino.

Learning Outcomes: At the end of the unit, the student will be able to:

- Understand how to develop IoT systems with Raspberry Pi using python programming. (L2)
- Develop IoT systems using IoT systems using Arduino. (L3)

Department of Artificial Intelligence & Data Science

Unit 5 Case Studies and Real-World Applications **9**

Real world design constraints - Applications - Asset management, Industrial automation, Smart grid, Commercial building automation, Smart cities - Participatory sensing - Data Analytics for IoT – Software & Management Tools for IoT Cloud Storage Models & Communication APIs - Cloud for IoT - Amazon Web Services for IoT.

Learning Outcomes: At the end of the unit, the student will be able to:

- Understand real world constraints and case studies. (L2)
- Design basic IoT application of different domains. (L3)

Prescribed Text Books:

1. Arshdeep Bahga, Vijay Madisetti, "Internet of Things – A hands-on approach", Universities Press, 2015
2. Olivier Hersent, David Boswarthick, Omar Elloumi, "The Internet of Things – Key applications and Protocols", Wiley, 2012

Reference Books:

1. Dieter Uckelmann, Mark Harrison, Michahelles, Florian (Eds), "Architecting the Internet of Things", Springer, 2011
2. Honbo Zhou,"The Internet of Things in the Cloud: A Middleware Perspective", CRC Press, 2012
3. Jan Holler, Vlasios Tsiatsis , Catherine Mulligan, Stamatis , Karnouskos, Stefan Avesand. David Boyle, "From Machine-to-Machine to the Internet of Things - Introduction to a New Age of Intelligence", Elsevier, 2014.

Course Outcomes:

At the end of the course, the student will be able to	Blooms Level of Learning
1. Analyze various protocols for IoT	L4
2. Develop Web services to access/control IoT devices.	L3
3. Design a portable IoT using Raspberry Pi.	L6
4. Deploy an IoT application and connect to the cloud.	L5
5. Analyze applications of IoT in real time scenario	L4

COs-POs-PSOs Mapping:

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
20A307BT.1	2	3	2	3	2	1	1	-	1	-	-	1	2	3	3
20A307BT.2	2	2	2	3	2	2	1	-	1	-	-	1	2	3	3
20A307BT.3	2	3	2	3	2	1	1	-	1	-	-	1	2	2	3
20A307BT.4	2	3	2	3	2	2	1	-	3	-	-	1	2	2	3
20A307BT.5	2	3	2	3	2	3	1	-	3	-	-	2	2	3	3

UNIT-I
INTRODUCTION OF IOT

IoT comprises things that have unique identities and are connected to internet. By 2020 there will be a total of 50 billion devices /things connected to internet. IoT is not limited to just connecting things to the internet but also allow things to communicate and exchange data.

DEFINITION:

A dynamic global n/w infrastructure with self configuring capabilities based on standard and interoperable communication protocols where physical and virtual-things have identities, physical attributes and virtual personalities and use intelligent interfaces, and are seamlessly integrated into information n/w, often communicate data associated with users and their environments.

Characteristics:

1) Dynamic & Self Adapting: IoT devices and systems may have the capability to dynamically adapt with the changing contexts and take actions based on their operating conditions, users context or sensed environment. Eg: the surveillance system is adapting itself based on context and changing conditions.
2) Self Configuring: allowing a large number of devices to work together to provide certain functionality.
3) Inter Operable Communication Protocols: support a number of interoperable communication protocols and can communicate with other devices and also with infrastructure.
4) Unique Identity: Each IoT device has a unique identity and a unique identifier (IP address).
5) Integrated into Information Network: that allow them to communicate and exchange data with other devices and systems.

Iot – key features

The most important features of IoT include artificial intelligence, connectivity, sensors,

active engagement, and small device use. A brief review of these features is given below

- **AI** – IoT essentially makes virtually anything “smart”, meaning it enhances every aspect of life with the power of data collection, artificial intelligence algorithms, and networks. This can mean something as simple as enhancing your refrigerator and cabinets to detect when milk and your favorite cereal run low, and to then place an order with your preferred grocer.
- **Connectivity** – New enabling technologies for networking, and specifically IoT networking, mean networks are no longer exclusively tied to major providers. Networks can exist on a much smaller and cheaper scale while still being practical. IoT creates these small networks between its system devices.
- **Sensors** – IoT loses its distinction without sensors. They act as defining instruments which transform IoT from a standard passive network of devices into an active system capable of real-world integration.
- **Active Engagement** – Much of today's interaction with connected technology happens through passive engagement. IoT introduces a new paradigm for active content, product, or service engagement.
- **Small Devices** – Devices, as predicted, have become smaller, cheaper, and more powerful over time. IoT exploits purpose-built small devices to deliver its precision, scalability, and versatility.

Iot – advantages

The advantages of IoT span across every area of lifestyle and business. Here is a list of some of the advantages that IoT has to offer –

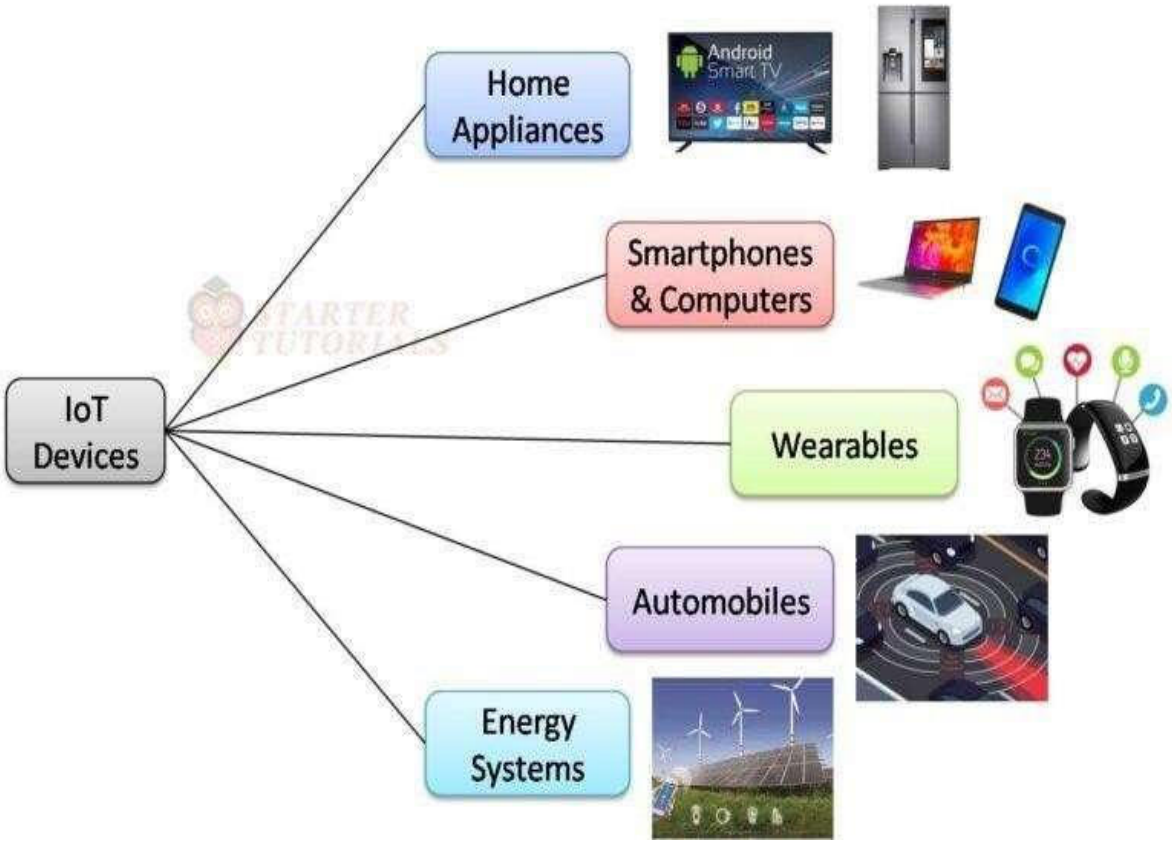
- Improved Customer Engagement – Current analytics suffer from blind-spots and significant flaws in accuracy; and as noted, engagement remains passive. IoT completely transforms this to achieve richer and more effective engagement with audiences.
- Technology Optimization – The same technologies and data which improve the customer experience also improve device use, and aid in more potent improvements to technology. IoT unlocks a world of critical functional and field data.
- Reduced Waste – IoT makes areas of improvement clear. Current analytics give

us superficial insight, but IoT provides real-world information leading to more effective management of resources.

- Enhanced Data Collection – Modern data collection suffers from its limitations and its design for passive use. IoT breaks it out of those spaces, and places it exactly where humans really want to go to analyze our world. It allows an accurate picture of everything. IoT – Disadvantages Though IoT delivers an impressive set of benefits, it also presents a significant set of challenges. Here is a list of some its major issues –
- Security – IoT creates an ecosystem of constantly connected devices communicating over networks. The system offers little control despite any security measures. This leaves users exposed to various kinds of attackers.
- Privacy – The sophistication of IoT provides substantial personal data in extreme detail without the user's active participation.
- Complexity – Some find IoT systems complicated in terms of design, deployment, and maintenance given their use of multiple technologies and a large set of new enabling technologies.
- Flexibility – Many are concerned about the flexibility of an IoT system to integrate easily with another. They worry about finding themselves with several conflicting or locked systems.
- Compliance – IoT, like any other technology in the realm of business, must comply with regulations. Its complexity makes the issue of compliance seem incredibly challenging when many consider standard software compliance a battle.

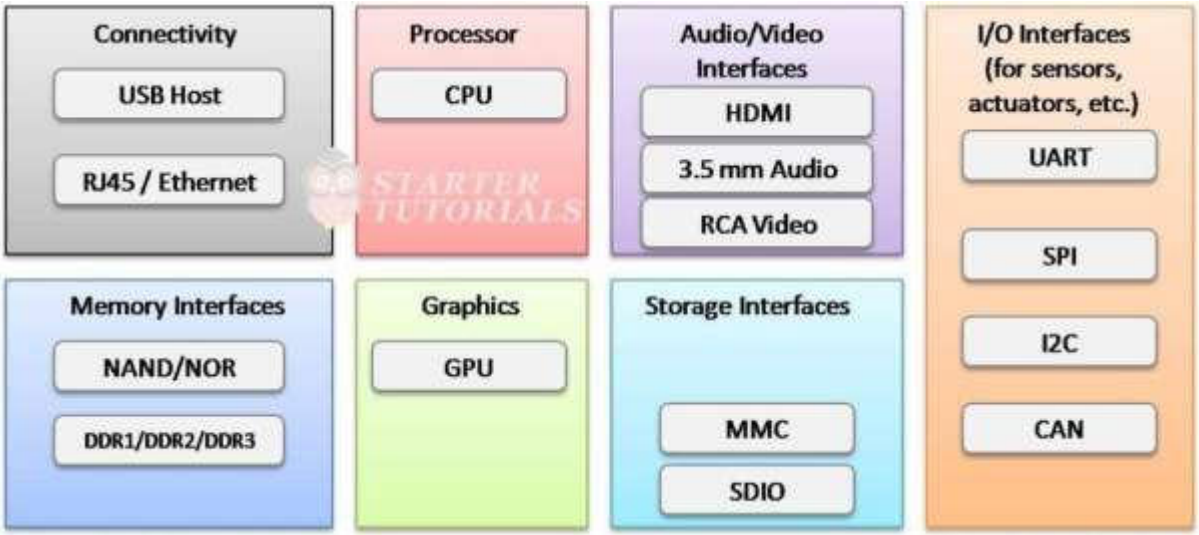
Applications of iot:

1) Home
2) Cities
3) Environment
4) Energy
5) Retail
6) Logistics
7) Agriculture
8) Industry
9) Health & Life Style



PHYSICAL DESIGN OF IOT

1) Things in IoT:



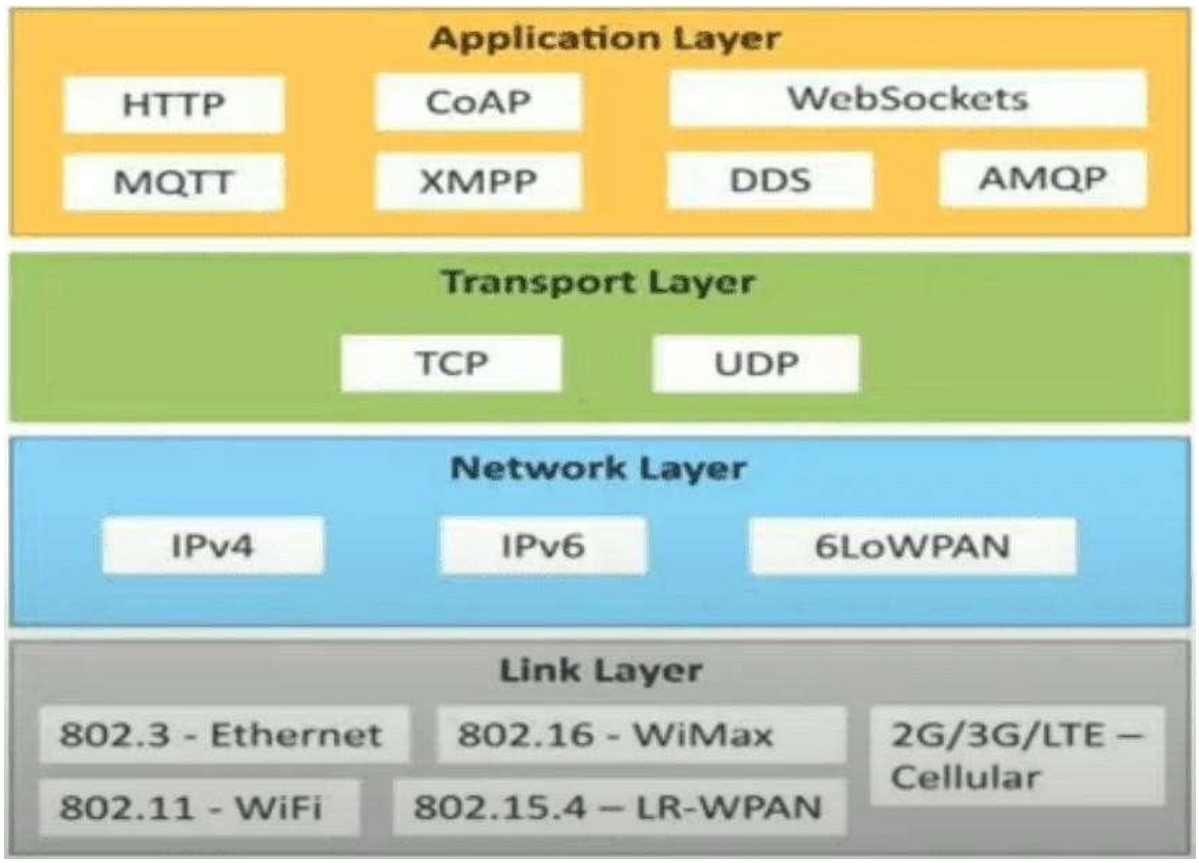
The things in IoT refers to IoT devices which have unique identities and perform remote sensing, actuating and monitoring capabilities. IoT devices can exchange

data with other connected devices applications. It collects data from other devices and process data either locally or remotely.

An IoT device may consist of several interfaces for communication to other devices both wired and wireless. These includes (i) I/O interfaces for sensors, (ii) Interfaces for internet connectivity (iii) memory and storage interfaces and (iv) audio/video interfaces.

2) IoT Protocols:

a) Link Layer : Protocols determine how data is physically sent over the network_s physical layer or medium. Local network connect to which host is attached. Hosts on the same link exchange data packets over the link layer using link layer protocols. Link layer determines how packets are coded and signaled by the h/w device over the medium to which the host is attached.



Protocols:

- 802.3-Ethernet: IEEE802.3 is collection of wired Ethernet standards for the link layer.

Eg: 802.3 uses co-axial cable; 802.3i uses copper twisted pair connection; 802.3j uses fiber optic connection; 802.3ae uses Ethernet over fiber.

- 802.11-WiFi: IEEE802.11 is a collection of wireless LAN(WLAN) communication standards including extensive description of link layer. Eg: 802.11a operates in 5GHz band, 802.11b and 802.11g operates in 2.4GHz band, 802.11n operates in 2.4/5GHz band, 802.11ac operates in 5GHz band, 802.11ad operates in 60Ghzband.
- 802.16 - WiMax: IEEE802.16 is a collection of wireless broadband standards including exclusive description of link layer. WiMax provide data rates from 1.5 Mb/s to 1Gb/s.
- 802.15.4-LR-WPAN: IEEE802.15.4 is a collection of standards for low rate wireless personal area network(LR-WPAN). Basis for high level communication protocols such as ZigBee. Provides data rate from 40kb/s to250kb/s.
- 2G/3G/4G-Mobile Communication: Data rates from 9.6kb/s(2G) to up to100Mb/s(4G).

b) Network/Internet Layer:

Responsible for sending IP datagrams from source n/w to destination n/w. Performs the host addressing and packet routing. Datagrams contains source and destination address.

Protocols:

- **IPv4:** Internet Protocol version4 is used to identify the devices on a n/w using a hierarchical addressing scheme. 32 bit address. Allows total of 2**32addresses.
- **IPv6:** Internet Protocol version6 uses 128 bit address scheme and allows 2**128 addresses.
- **6LOWPAN:(**IPv6overLowpowerWirelessPersonalAreaNetwork)operates in 2.4 GHz frequency range and data transfer 250 kb/s.

c) Transport Layer:

Provides end-to-end message transfer capability independent of the underlying

n/w. Set up on connection with ACK as in TCP and without ACK as in UDP.

Provides functions such as error control, segmentation, flow control and congestion control.

Protocols:

- **TCP:** Transmission Control Protocol used by web browsers(along with HTTP and HTTPS), email(along with SMTP, FTP). Connection oriented and stateless protocol. IP Protocol deals with sending packets, TCP ensures reliable transmission of protocols in order. Avoids n/w congestion and congestion collapse.
- **UDP:** User Datagram Protocol is connectionless protocol. Useful in time sensitive applications, very small data units to exchange. Transaction oriented and stateless protocol. Does not provide guaranteed delivery.

d) Application Layer: Defines how the applications interface with lower layer protocols to send data over the n/w. Enables process-to-process communication using ports.

Protocols:

- **HTTP:** Hyper Text Transfer Protocol that forms foundation of WWW. Follow request- response model Stateless protocol.
- **CoAP:** Constrained Application Protocol for machine-to-machine (M2M) applications with constrained devices, constrained environment and constrained n/w. Uses client- server architecture.
- **WebSocket**: allows full duplex communication over a single socket connection.
- **MQTT:** Message Queue Telemetry Transport is light weight messaging protocol based on publish-subscribe model. Uses client server architecture. Well suited for constrained environment.
- **XMPP:** Extensible Message and Presence Protocol for real time communication and streaming XML data between network entities. Support client-server and server-server communication.

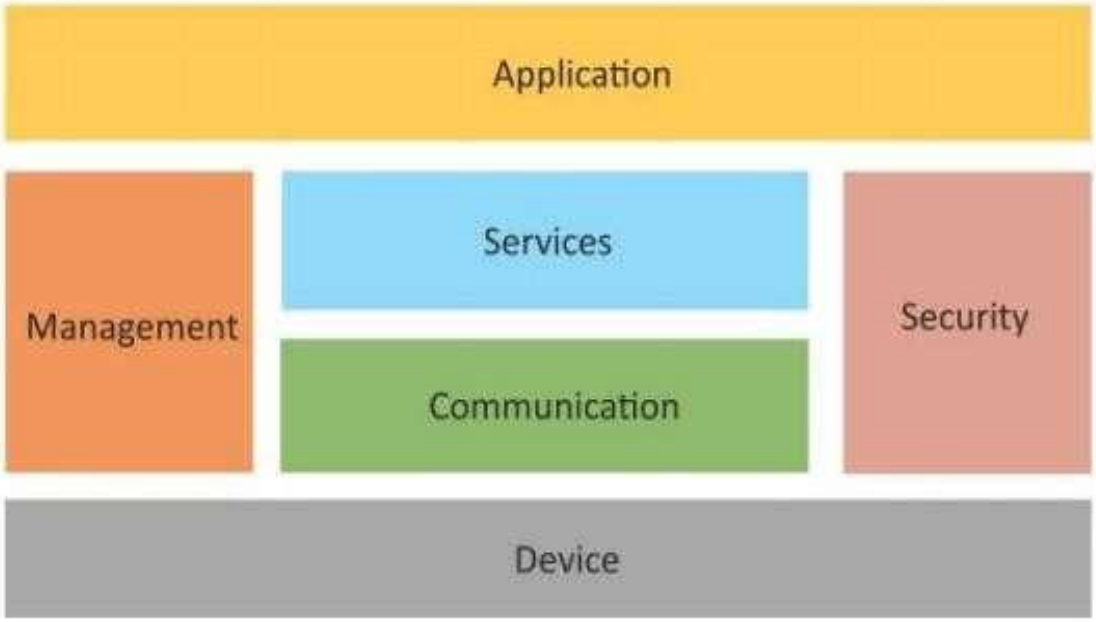
- **DDS:** Data Distribution Service is data centric middleware standards for device-to-device or machine-to-machine communication. Uses publish-subscribe model.
- **AMQP:** Advanced Message Queuing Protocol is open application layer protocol for business messaging. Supports both point-to-point and publish-subscribe model.

LOGICAL DESIGN OF IOT

Refers to an abstract represent of entities and processes without going into the low level specifies of implementation.

1) IoT Functional Blocks 2) IoT Communication Models 3) IoT Comm. APIs

- **IoT Functional Blocks:** Provide the system the capabilities for identification, sensing, actuation, communication and management.



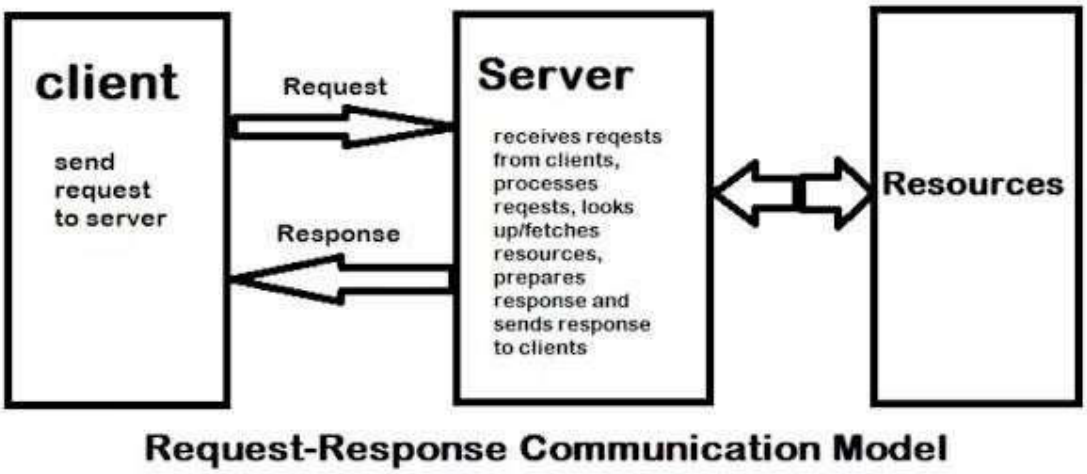
- **Device:** An IoT system comprises of devices that provide sensing, actuation, monitoring and control functions.
- **Communication:** handles the communication for IoTsystem.
- Services: for device monitoring, device control services, data publishing services and services for device discovery.

- **Management:** Provides various functions to govern the IoT system.

- **Security:** Secures IoT system and priority functions such as authentication ,authorization, message and context integrity and data security.
- **Application:** IoT application provide an interface that the users can use to control and monitor various aspects of IoT system.
- **IoT Communication Models**:

1) Request-Response 2) Publish-Subscibe3)Push-Pull4) ExclusivePair

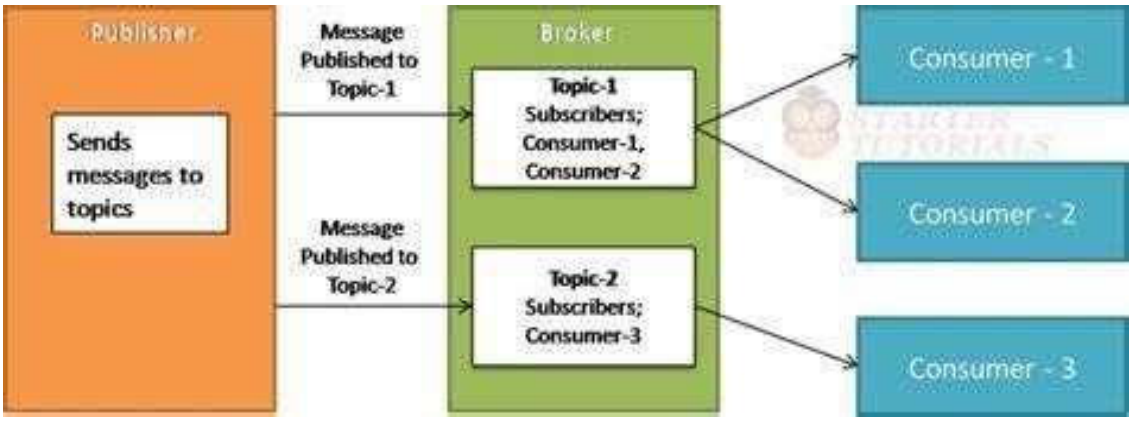
1) Request-Response Model:



Request-Response Communication Model

In which the client sends request to the server and the server replies to requests. Is a stateless communication model and each request-response pair is independent of others.

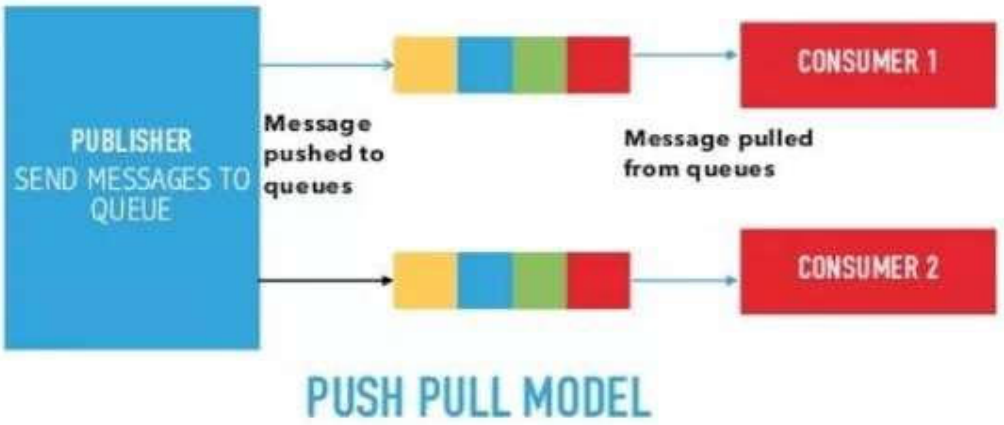
2) Publish-Subscibe Model:



Involves publishers, brokers and consumers. Publishers are source of data. Publishers send data to the topics which are managed by the broker. Publishers are not aware of the consumers. Consumers subscribe to the topics which are managed by the broker. When the broker receives data for a topic from the publisher, it sends the data to all the subscribed consumers.

3) Push-Pull Model:

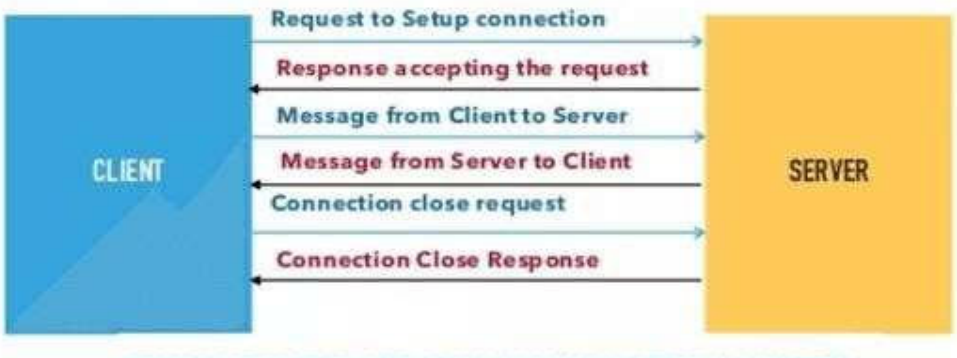
in which data producers push data to queues and consumers pull data from the queues. Producers do not need to aware of the consumers. Queues help in decoupling the message between the producers and consumers.



PUSH PULL MODEL

4) Exclusive Pair:

is bi-directional, fully duplex communication model that uses a persistent connection between the client and server. Once connection is set up it remains open until the client send a request to close the connection. Is a stateful communication model and server is aware of all the open connections.



EXCLUSIVE PAIR COMMUNICATION MODEL

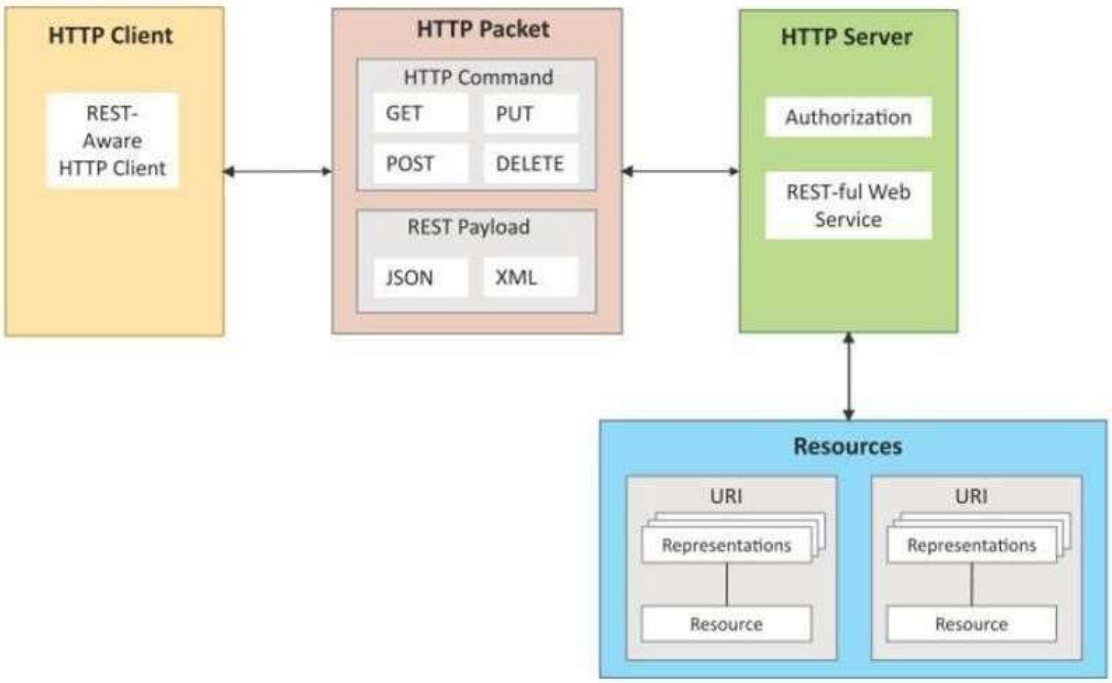
a) REST based communication APIs(Request-Response Based Model)

b) WebSocket based Communication APIs(Exclusive PairBased Model)

a) REST based communication APIs:

Representational State Transfer(REST) is a set of architectural principles by which we can design web services and web APIs that focus on a systems resources and have resource states are addressed and transferred.

The REST architectural constraints: Fig. shows communication between client server with REST APIs.



Client-Server: The principle behind client-server constraint is the separation of concerns. Separation allows client and server to be independently developed and updated.

Stateless: Each request from client to server must contain all the info. Necessary to understand the request, and cannot take advantage of any stored context on the server.

Cache-able: Cache constraint requires that the data within a response to a request be implicitly or explicitly labeled as cache-able or non-cacheable. If a

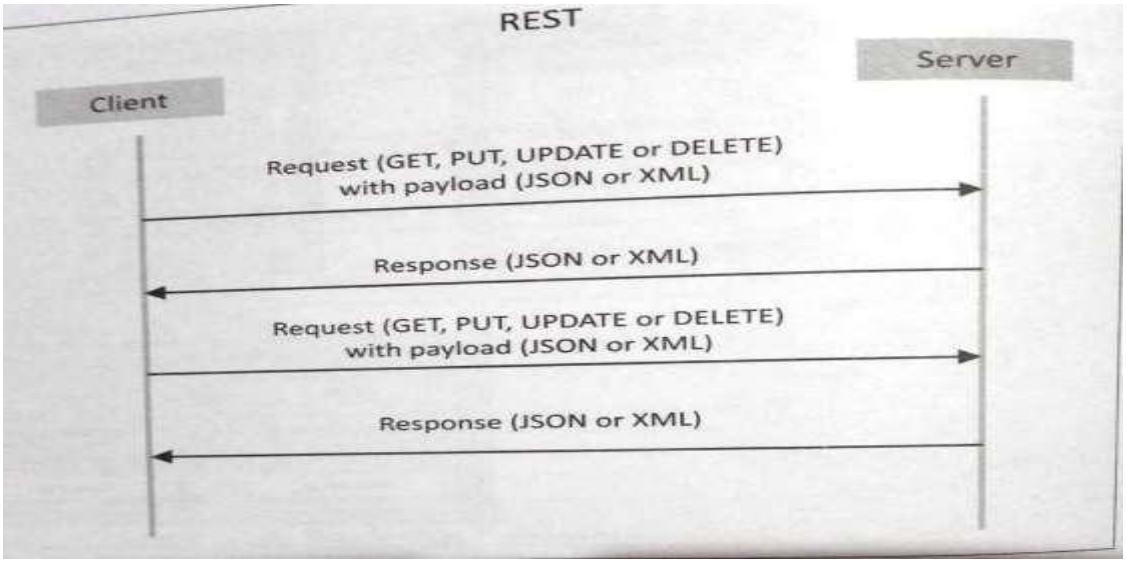
response is cache-able, then a client cache is given the right to reuse that response data for later, equivalent requests.

Layered System: constraints the behavior of components such that each component cannot see beyond the immediate layer with which they are interacting. User

Interface: constraint requires that the method of communication between a client and a server must be uniform.

Code on Demand: Servers can provide executable code or scripts for clients to execute in their context. This constraint is the only one that is optional.

Request-Response model used by REST:



Tful web service is a collection of resources which are represented by URIs. RESTful web API has a base URI(e.g: http://example.com/api/tasks/). The clients and requests to these URIs using the methods defined by the HTTP protocol(e.g: GET, PUT, POST or DELETE). A RESTful web service can support various internet media types.

b) WebSocket Based Communication APIs:

WebSocket APIs allow bi-directional, full duplex communication between clients and servers. WebSocket APIs follow the exclusive pair communication model.

IOT ENABLING TECHNOLOGIES

IoT is enabled by several technologies including Wireless Sensor Networks, Cloud Computing, Big Data Analytics, Embedded Systems, Security Protocols and architectures, Communication Protocols, Web Services, Mobile internet and semantic search engines.

1) **Wireless Sensor Network(WSN):** Comprises of distributed devices with sensors which are used to monitor the environmental and physical conditions. Zig Bee is one of the most popular wireless technologies used byWSNs.

   WSNs used in IoT systems are described as follows:

   - Weather Monitoring System: in which nodes collect temp, humidity and other data, which is aggregated and analyzed.
   - Indoor air quality monitoring systems: to collect data on the indoor air quality and concentration of various gases.
   - Soil Moisture Monitoring Systems: to monitor soil moisture at variouslocations.
   - Surveillance Systems: use WSNs for collecting surveillance data(motiondata detection).
   - Smart Grids : use WSNs for monitoring grids at variouspoints.
   - Structural Health Monitoring Systems: Use WSNs to monitor the health of structures(building, bridges) by collecting vibrations from

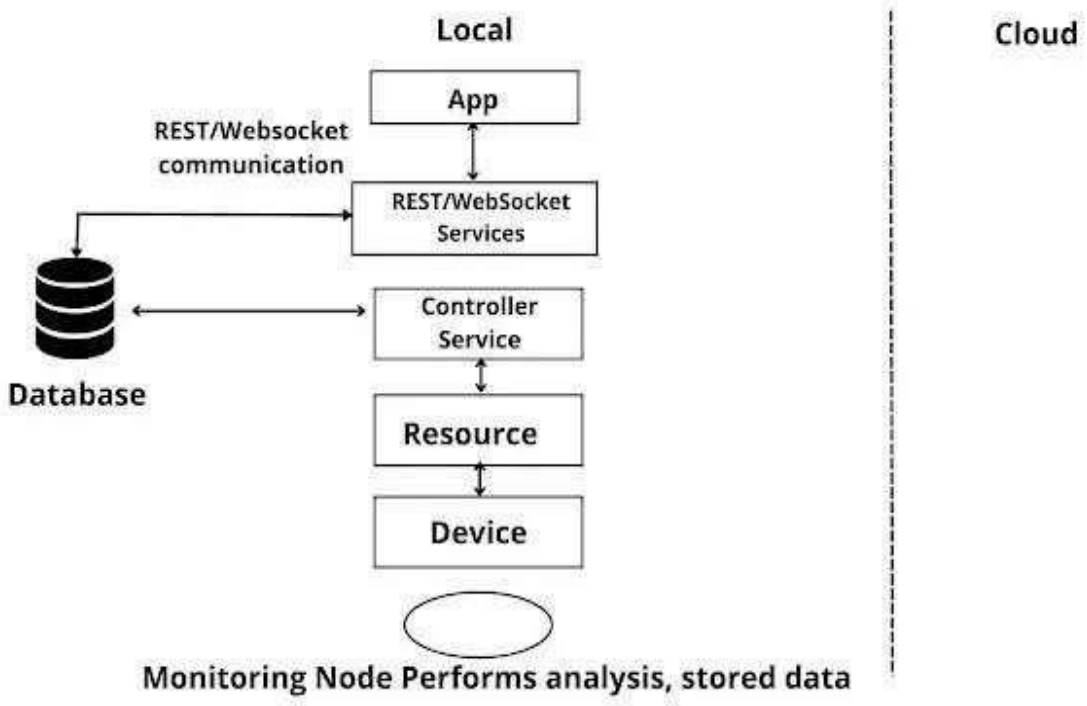
sensor nodes deployed at various points in the structure.

2) **Cloud Computing:** Services are offered to users in different forms.
    - Infrastructure-as-a-service(IaaS):provides users the ability to provision computing and storage resources. These resources are provided to the users as a virtual machine instances and virtual storage.
    - Platform-as-a-Service(PaaS): provides users the ability to develop and deploy application in cloud using the development tools, APIs, software libraries and services provided by the cloud service provider.
    - Software-as-a-Service(SaaS): provides the user a complete software application or the user interface to the application itself.
3) **Big Data Analytics:** Some examples of big data generated by IoT are
    - Sensor data generated by IoT systems.
    - Machine sensor data collected from sensors established in industrial and energy systems.
    - Health and fitness data generated IoT devices.
    - Data generated by IoT systems for location and tracking vehicles.
    - Data generated by retail inventory monitoring systems.
4) **Communication Protocols**: form the back-bone of IoT systems and enable network connectivity and coupling to applications.
    - Allow devices to exchange data over network.
    - Define the exchange formats, data encoding addressing schemes for device and routing of packets from source to destination.
    - It includes sequence control, flow control and retransmission of lost packets.
5) **Embedded Systems:** is a computer system that has computer hardware and software embedded to perform specific tasks. Embedded System range from low cost miniaturized devices such as digital watches to devices such as digital cameras, POS terminals, vending machines, appliances etc.,

P.HARI KRISHNA

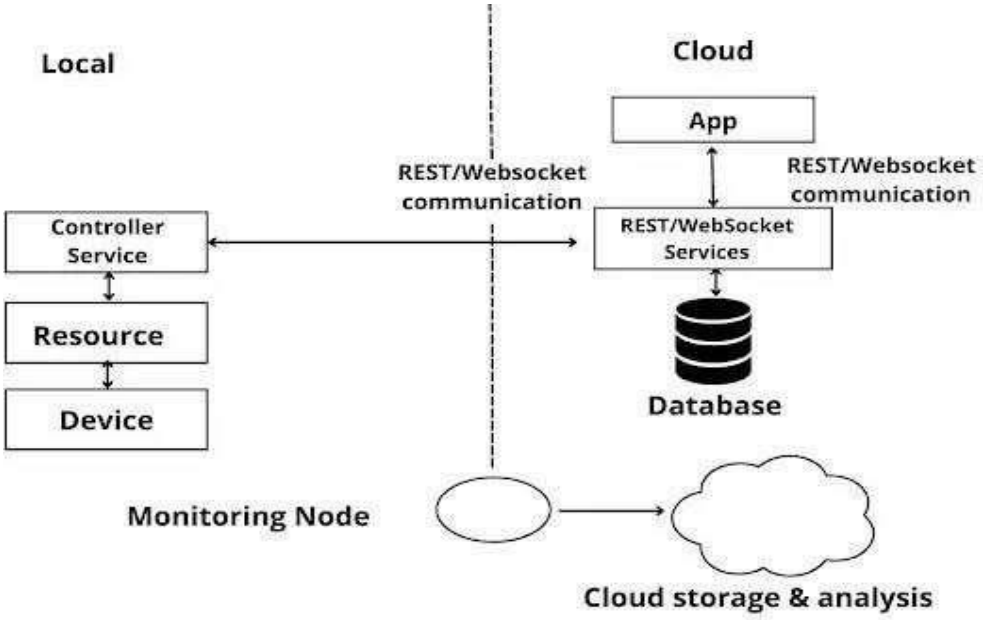
IoT Levels and Deployment Templates

1) **IoT Level1:** System has a single node that performs sensing and/or actuation, stores data, performs analysis and host the application as shown in fig. Suitable for modeling low cost and low complexity solutions where the data involved is not big and analysis requirement are not computationally intensive. An e.g., of IoT Level1 is Home automation.



2) **IoT Level2:** has a single node that performs sensing and/or actuating and local analysis as shown in fig. Data is stored in cloud and application is usually cloud based. Level2 IoT systems are suitable for solutions where data are involved is big, however, the primary analysis requirement is not computationally intensive and can be done locally itself. An e,g., of Level2 IoT system for Smart Irrigation.

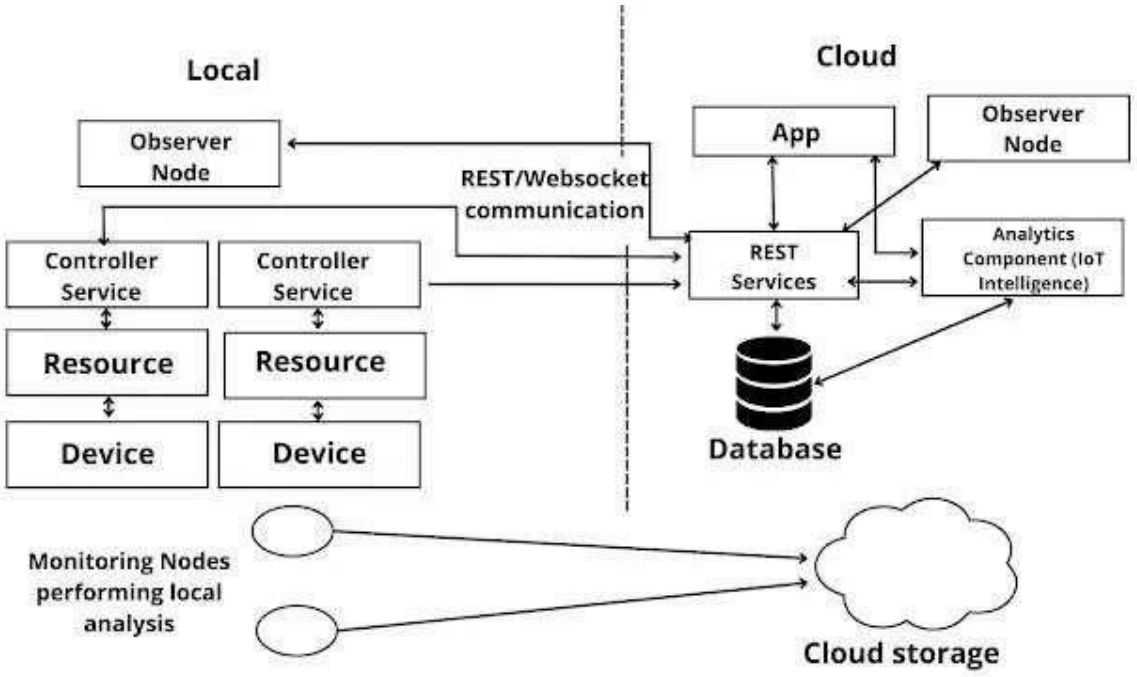
3) **IoT Level3:** system has a single node. Data is stored and analyzed in the cloud application is cloud based as shown in fig. Level3 IoT systems are suitable for solutions where the data involved is big and analysis requirements are computationally intensive. An example of IoT level3 system for tracking package



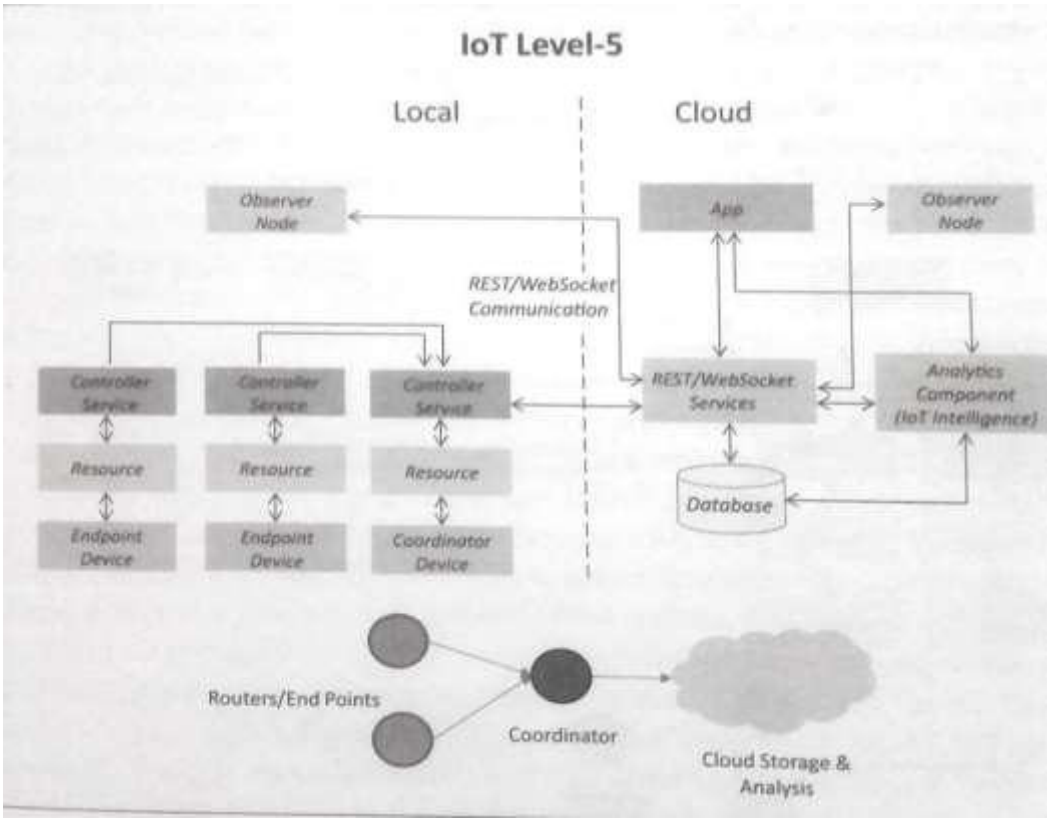
4) **IoT Level4:** System has multiple nodes that perform local analysis. Data is

stored in the cloud and application is cloud based as shown in fig. Level4 contains local and cloud based observer nodes which can subscribe to and

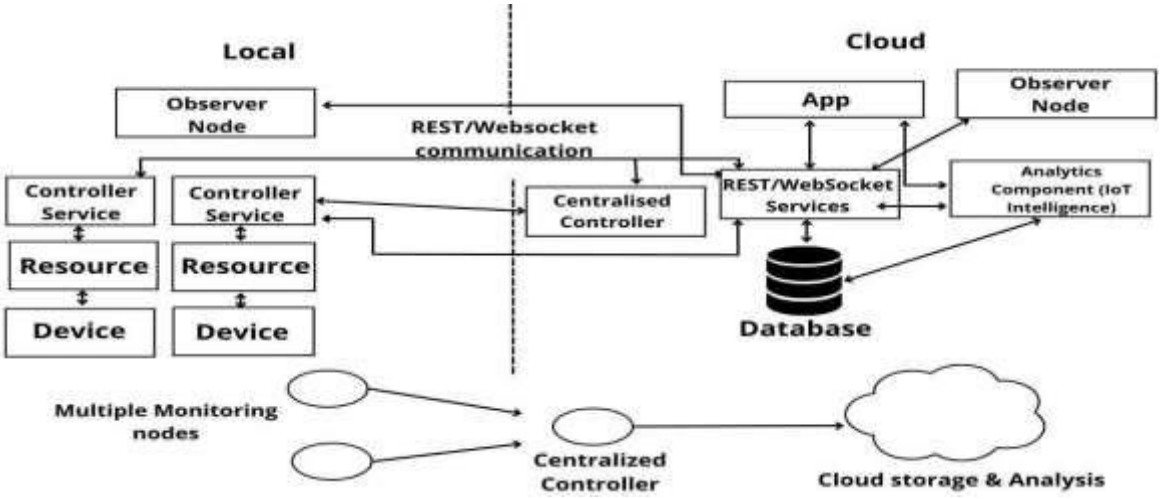
receive information collected in the cloud from IoT devices. An example of a Level4 IoT system for Noise Monitoring.



5) **IoT Level5:** System has multiple end nodes and one coordinator node as shown in fig. The end nodes that perform sensing and/or actuation. Coordinator node collects data from theendnodesandsendstothecloud.Dataisstoredand analysed inthecloudand application is cloud based. Level5 IoT systems are suitable for solution based on wireless sensor network, in which data involved is big and analysis requirements are computationally intensive. An example of Level5 system for Forest Fire Detection.



6) IoT Level6: System has multiple independent end nodes that perform sensing and/or actuation and sensed data to the cloud. Data is stored in the cloud and application is cloud based as shown in fig. The analytics component analyses the data and stores the result in the cloud data base. The results are visualized with cloud based application. The centralized controller is aware of the status of all the end nodes and sends control commands to nodes. An example of a Level6 IoT system for Weather Monitoring System.



P.HARI KRISHNA

DOMAIN SPECIFIC IoTs

1) Home Automation:

a) Smart Lighting: helps in saving energy by adapting the lighting to the ambient conditions and switching on/off or diming the light when needed.

b) Smart Appliances: make the management easier and also provide status information to the users remotely.

c) Intrusion Detection: use security cameras and sensors(PIR sensors and door sensors) to detect intrusion and raise alerts. Alerts can be in the form of SMS or email sent to the user.

d) Smoke/Gas Detectors: Smoke detectors are installed in homes and buildings to detect smoke that is typically an early sign of fire. Alerts raised by smoke detectors can be in the form of signals to a fire alarm system. Gas detectors can detect the presence of harmful gases such as CO, LPGetc.,

2) Cities:

a) Smart Parking: make the search for parking space easier and convenient for drivers. Smart parking are powered by IoT systems that detect the no. of empty parking slots and send information over internet to smart application backends.

b) Smart Lighting: for roads, parks and buildings can help in saving energy.

c) Smart Roads: Equipped with sensors can provide information on driving condition, travel time estimating and alert in case of poor driving conditions, traffic condition and accidents.

d) Structural Health Monitoring: uses a network of sensors to monitor the vibration levels in the structures such as bridges and buildings.

e) Surveillance: The video feeds from surveillance cameras can be aggregated in cloud based scalable storage solution.

f) Emergency Response: IoT systems for fire detection, gas and water leakage detection can help in generating alerts and minimizing their effects on the critical infrastructures.

3) Environment:

a) Weather Monitoring: Systems collect data from a no. of sensors attached and send the data to cloud based applications and storage back ends. The data collected in cloud can then be analyzed and visualized by cloud based applications.

b) Air Pollution Monitoring: System can monitor emission of harmful gases(CO2, CO, NO, NO2 etc.,) by factories and automobiles using gaseous and meteorological sensors. The collected data can be analyzed to make informed decisions on pollutions control approaches.

c) Noise Pollution Monitoring: Due to growing urban development, noise levels in cities have increased and even become alarmingly high in some cities. IoT based noise pollution monitoring systems use a no. of noise monitoring systems that are deployed at different places in a city. The data on noise levels from the station is collected on servers or in the cloud. The collected data is then aggregated to generate noise maps.

d) Forest Fire Detection: Forest fire can cause damage to natural resources, property and human life. Early detection of forest fire can help in minimizing damage.

e) River Flood Detection: River floods can cause damage to natural and human resources and human life. Early warnings of floods can be given by monitoring the water level and flow rate. IoT based river flood monitoring system uses a no. of sensor nodes that monitor the water level and flow rate sensors.

4) Energy:

a) Smart Grids: is a data communication network integrated with the electrical grids that collects and analyze data captured in near-real-time about power transmission, distribution and consumption. Smart grid technology provides predictive information and recommendations to utilities, their suppliers, and their customers on how best to manage power. By using IoT based sensing and measurement technologies, the health of equipment and integrity of the grid can be evaluated.

b) Renewable Energy Systems: IoT based systems integrated with the transformers at the point of interconnection measure the electrical variables and how much power is fed into the grid. For wind energy systems, closed-loop controls can be used to regulate the voltage at point of interconnection which coordinate wind turbine outputs and provides power support.

c) Prognostics: In systems such as power grids, real-time information is collected using specialized electrical sensors called Phasor MeasurmentUnits(PMUs) at the substations. The information received from PMUs must be monitored in real-time for estimating the state of the system and for predicting failures.

5) Retail:

a) Inventory Management: IoT systems enable remote monitoring of inventory using data collected by RFIDreaders.

b) Smart Payments: Solutions such as contact-less payments powered by technologies such as Near Field Communication(NFC) and Bluetooth.

c) Smart Vending Machines: Sensors in a smart vending machines monitors its operations and send the data to cloud which can be used for predictive maintenance.

6) Logistics:

a) Route generation & scheduling: IoT based system backed by cloud can provide first response to the route generation queries and can be scaled upto serve a large transportation network.

b) Fleet Tracking: Use GPS to track locations of vehicles inreal-time. c) Shipment Monitoring: IoT based shipment monitoring systems use sensors such as temp, humidity, to monitor the conditions and send data to cloud, where it can be analyzed to detect foods poilage.

d) Remote Vehicle Diagnostics: Systems use on-board IoT devices for collecting data on Vehicle operations(speed, RPMetc.,) and status of various

vehicle subsystems.

7) Agriculture:

a) Smart Irrigation: to determine moisture amount in soil. b) Green House Control: to improve productivity.

8) Industry:

a) Machine diagnosis and prognosis

b) Indoor Air Quality Monitoring

9) Health and LifeStyle:

a) Health & Fitness Monitoring

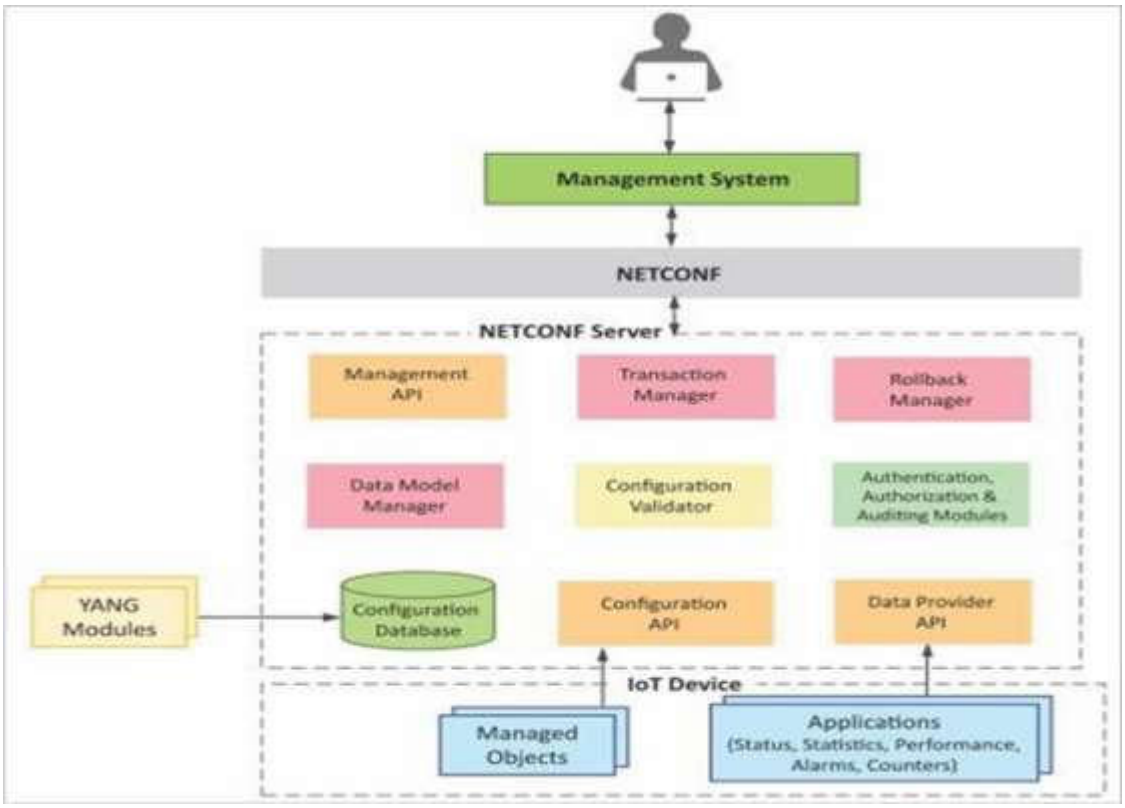
b) Wearable Electronics

IoT Systems Management with NETCONF-YANG

YANG is a data modeling language used to model configuration and state data manipulated by the NETCONF protocol.

The generic approach of Io T device management with NETCONF-YANG. Roles of various components are:

1) Management System
2) Management API
3) Transaction Manager
4) Rollback Manager
5) Data Model Manager
6) Configuration Validator
7) Configuration Database
8) Configuration API
9) Data Provider API



1) **Management System :** The operator uses a management system to send NETCONF messages to configure the IoT device and receives state information and notifications from the device as NETCONF messages.
2) **Management API:** allows management application to start NETCONF sessions.
3) **Transaction Manager:** executes all the NETCONF transactions and ensures that ACID properties hold true for the transactions.
4) **Rollback Manager:** is responsible for generating all the transactions

Necessary to rollback a current configuration to it so riginal state.

5) **Data Model Manager :** Keeps track of all the YANG data models and the corresponding managed objects. Also keeps track of the applications which provide data for each part of a data model.
6) **Configuration Validator:** checks if the resulting configuration after applying a transaction would be a valid configuration.
7) **Configuration Database:** contains both configuration and operational data.
8) **Configuration API :** Using the configuration API the application on the IoT device can be read configuration data from the configuration datas tore and write opeartional data to the opearational data store.
9) **Data Provider API:** Applications onthe IoT device can register for callbacks for various events using the Data Provider API. Through the Data Provider API, the applications can report statistics and opeartionaldata.

Steps for IoT device Management with NETCONF-YANG

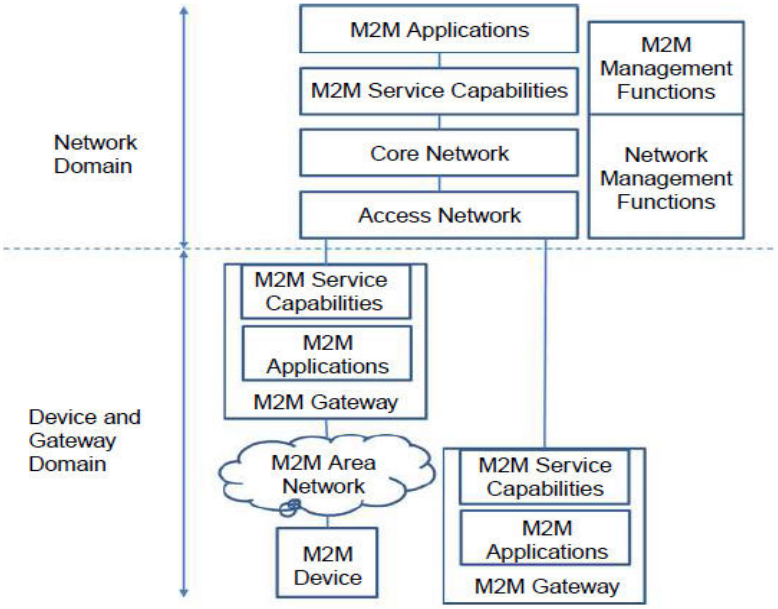
1) Create a YANG model of the system that defines the configuration and state data of the system.
2) Complete the YANG model with the_Inctool' which comes with Lib net conf.
3) Fill in the IoT device management code in the Trans API module.
4) Build the call backs Cfile to generate the library file.
5) Load the YANG module and the Trans API module into the Net opeer server using Net opeer manager tool.
6) The operator can now connect from the management system to the Netopeer server using the Netopeer CLI.
7) Operator can issue NETCONF commands from the Net opeer CLI. Command can be issued to changew the configurationdsta, get operational dat or execute an RPC on the IoT device.

UNIT II: IoT ARCHITECTURE

M2M high- IETF architecture for IoT - OGC architecture - IoT reference model - Domain model - information model - functional model - communication model - IoT reference architecture.

ETSI M2M high-level 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).
- 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).



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**: 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 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.
- **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.
- 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.

• **M2M Authentication Server (MAS):** This is the safe execution environment where permanent security credentials such as the M2M Root Key are stored.

➢ 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.

➢ An M2M Application is the main application logic that uses the Service Capabilities to achieve the M2M system requirements.

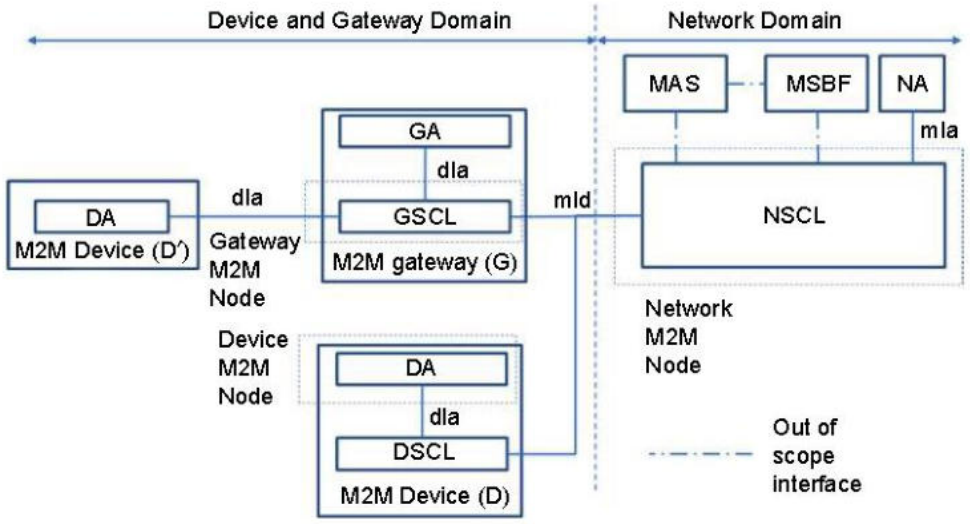


FIGURE 6.2

M2M Service Capabilities, M2M Nodes and Open Interfaces.

➢ The application logic can be deployed on a Device (Device Application, DA), Gateway (Gateway Application, GA) or Network (Network Application, NA).

➢ The SCL (Service Capability Layer) 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.

IETF architecture for IoT

- Internet Engineering Task Force architecture

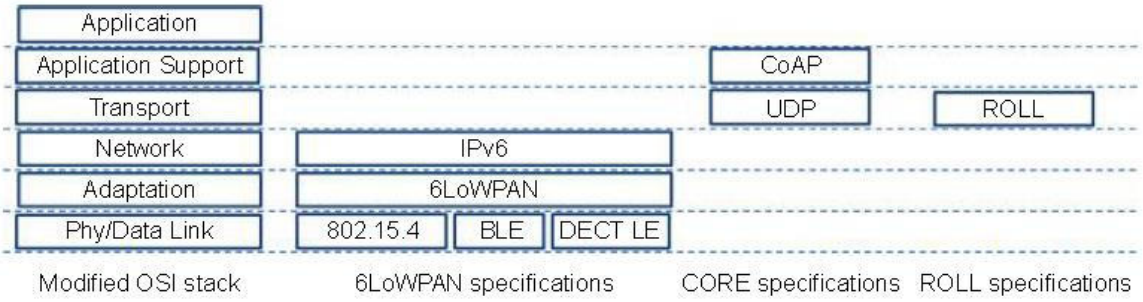


FIGURE 6.7

IETF Working Groups and Specification Scope.

- 6LoWPAN (IPv6 over Low-power WPAN), CoRE (Constrained RESTful Environments), and ROLL (Routing Over Low power and Lossy networks).
- Each set of specifications makes an attempt to address a different part of the communication stack of a constrained device.
- One layer called Application Support which includes the Presentation and Session Layers combined. one intermediate layer is introduced: the Adaptation Layer
- It 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; however, it does not describe the specific names of resources a node should host.
- The IETF CoAP draft specification describes the Transport and Application Support Layers, which essentially defines the transport packet formats, reliability support on top of UDP, and a RESTful application protocol with GET/PUT/POST/DELETE methods similar to HTTP with CoAP clients operating on CoAP server resources.
- A CoAP server is just a logical protocol entity, and the name “server” does not necessarily

imply that its functionality is deployed on a very powerful machine; a CoAP server can be hosted on a constrained device.

- The CoRE Link Format specification describes a discovery method for the CoAP resources of a CoAP server.
- For example, a CoAP client sending a request with the GET method to a specific well defined server resource (./well-known/core) should receive a response with a list of CoAP resources and some of their capabilities (e.g. resource type, interface type).
- The CoRE interface specification describes interface types and corresponding expected behavior of the RESTful methods.
- The IETF stack for IoT does not currently include any specifications that are similar to the profile specifications of other IoT technologies such as ZigBee
- Profile specification means a document that describes a list of profile names and their mappings to specific protocol stack behavior, specific information model, and specific serialization of this information model over the relevant communication medium.
- An example of a profile specification excerpt would mandate that an exemplary "Temperature" profile:
  (a) should support a resource called /temp,
  (b) the resource /temp must respond to a GET method request from a client, and
  (c) the response to a GET method request shall be a temperature value in degrees Celsius formatted as a text string with the format ",temperature value encoded in a decimal number ._C"
- 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.

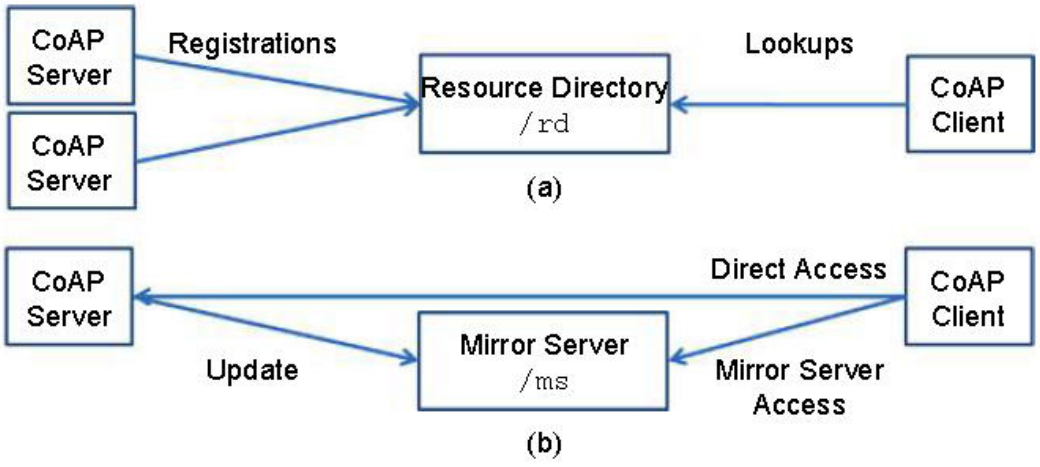


FIGURE 6.8

IETF CoRE Functional Components: (a) Resource Directory, (b) Mirror Server.

- 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. (e.g. temperature sensor resource type).
- Resource Directory is a rendezvous mechanism for CoAP Server resource descriptions, a Mirror Server 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 6.8b).
- 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.
- An example of such a CoAP Server is one that resides on a real device whose communication capabilities are turned off in order to preserve energy, e.g. battery-powered radio devices whose radio and/or processor goes to sleep mode.
- Typically, a Mirror Server is hosted on a device or machine that is always available.
- The IETF CoRE workgroup has included the fundamentals of a mapping process between HTTP and CoAP in the IETF CoAP specification as well as a set of guidelines for the interworking between HTTP and CoAP.
- 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.
- HTTP Client sends an HTTP request to a CoAP server (Figure 6.9a) 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 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 in order to reach the gateway.
- The Cross proxy translates the CoAP response code to the corresponding HTTP code, transforms the included media, creates the HTTP response, and dispatches it to the HTTP client.

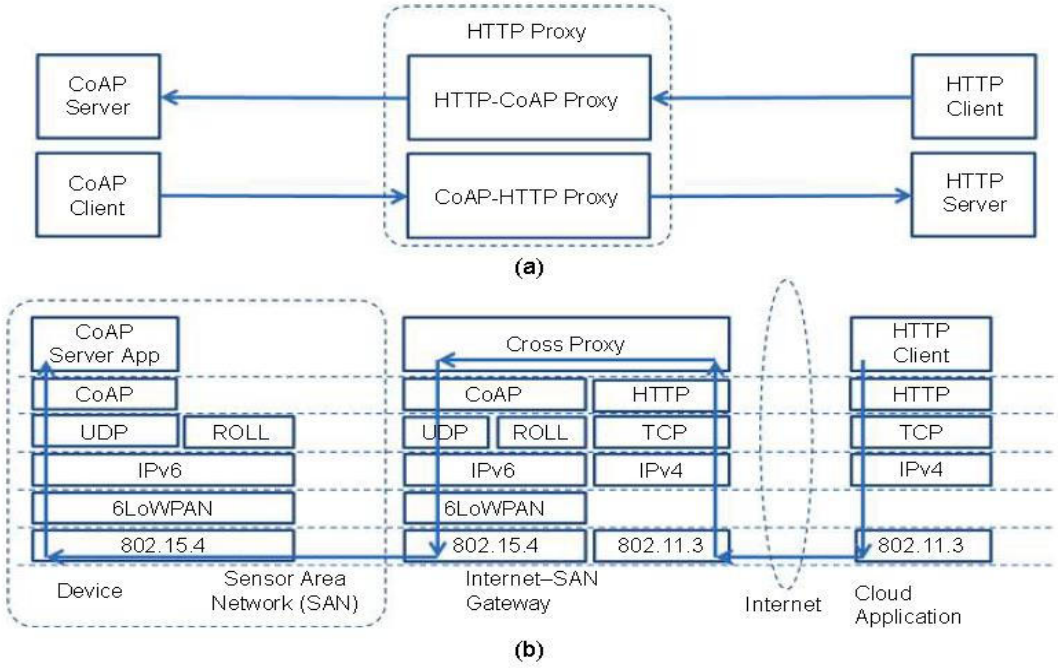


FIGURE 6.9

IETF CoRE HTTP Proxy: (a) possible configurations, (b) example layer interaction upon a request from a HTTP Client to a CoAP Server via a HTTP Proxy.

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.
- OGC includes, among other working groups, the Sensor Web Enablement (SWE) (OGC SWE 2013) domain working group, which develops standards for sensor system models (e.g. Sensor Model Language, or SensorML), sensor information models (Observations &

Measurements, or O&M), and sensor services that follow the Service-Oriented Architecture (SOA) paradigm.

- 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.

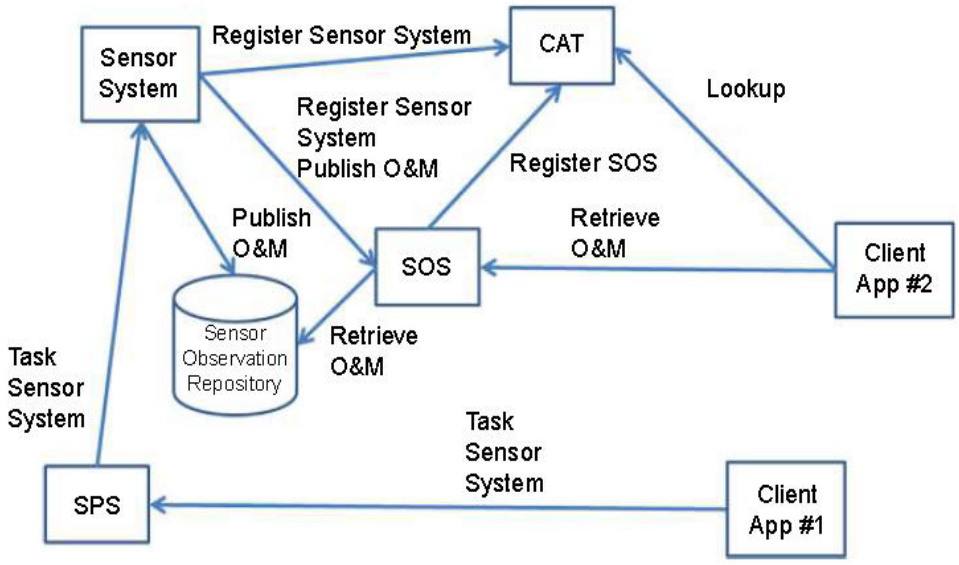


FIGURE 6.10

OGC functional architecture and interactions.

- 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 sensors every 10 seconds and publish the measurements using O&M and the SWE Common Data model to the SOS.
- Another client application #2 looks up the Catalog, aiming at locating an SOS for retrieving the measurements from the Sensor System.
- The application receives the contact information of the SOS and requests from the sensor observations from the specific sensor system from the SOS.
- As a response, the measurements from the sensor system using O&M and the SWE Common Data model are dispatched to the client application #2.
- The main objective of the OGC standards is to enable data, information, and service interoperability.

IOT Reference Model

- The IoT Reference Model aims at establishing a common grounding and a common language for IoT architectures and IoT systems.
- A reference model describes the domain using a number of sub-models (Figure 7.1).
- The domain model of an architecture model captures the main concepts or entities in the domain, the domain model adds descriptions about the relationship between the concepts.
- 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 need 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.

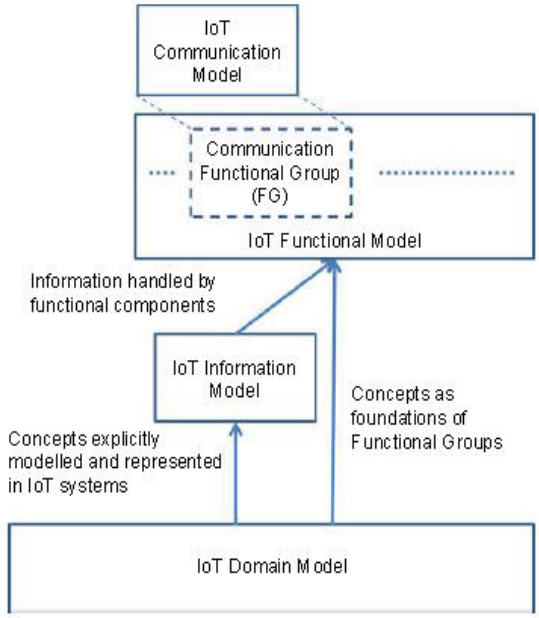


FIGURE 7.1
IoT Reference Model.

- 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.
- 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 modelled, which is explicitly gathered, stored and processed in an IoT system, e.g. information about Devices, IoT Services and Virtual Entities.

- 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.
- A key functionality in any distributed computer system is the communication between the different components.
- 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.

IoT domain model

- Domain model as a description of concepts belonging to a particular area of interest.
- The domain model also defines basic attributes of these 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.
- The main purpose of a domain model is to generate a common understanding of the target domain in question.
- 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.
- The domain model captures the basic attributes of the main concepts and the relationship between these concepts.

Model notation and semantics

- 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 and a set of attributes and operations.
- Notation-wise this is represented as a box with two compartments, one containing the class name and the other containing the attributes.
- 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.

- For example, in Figure 7.4, Class A is a general case of Class B or Class B is special case or specialization of Class A.
- Generalization is also called an "is-a" relationship. For example, in Figure 7.4 Class B "is-a" Class A. A specialized class/subclass/child class inherits the attributes and the operations from the general/super/parent class, respectively, and also contains its own attributes and operations.
- 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.

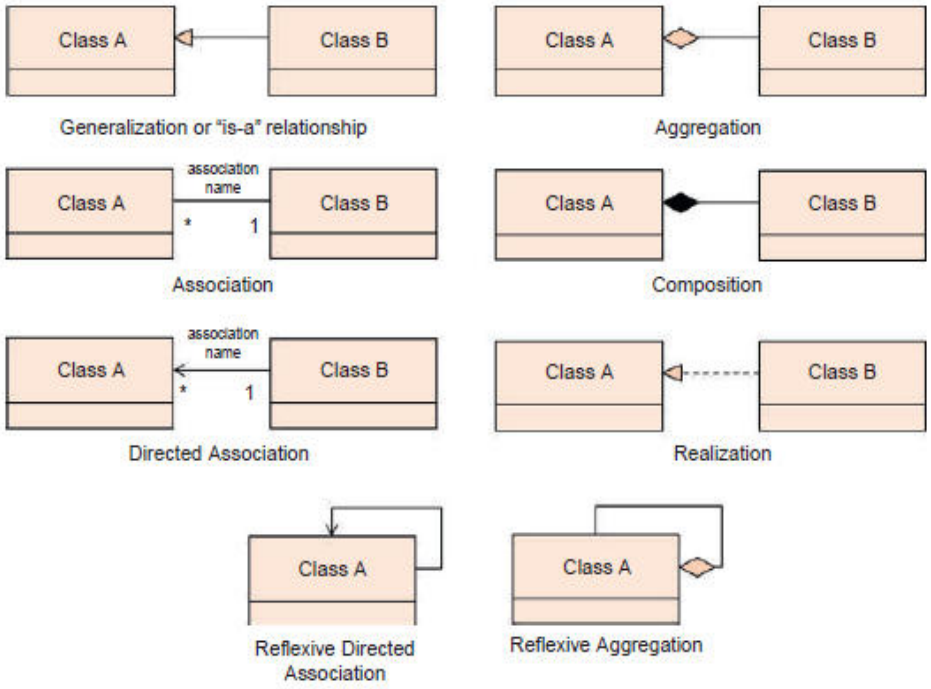


FIGURE 7.4

UML Class diagram main modeling concepts.

- The class that touches the hollow diamond is the whole class while the other class is the part class.
- For example, in Figure 7.4, class B represents a part of the whole Class A, or in other words, an object of Class A "contains" or "has-a" object of Class B.
- When the line with the hollow diamond starts and ends in the same class, then this relationship of one class to itself is called Reflexive Aggregation, and it denotes that objects of a class (e.g. Class A in Figure 7.4) contain objects of the same 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. For example, in Figure 7.4, Class B is part of Class A. Composition and Aggregation are very similar, with the difference being the coincident lifetime to the objects of classes related with composition.

- In other words, if an object of Class B is part of an object of Class A (composition), when the object of Class A disappears, the object of Class B also disappears.
- A plain line without arrowheads or diamonds represents the Association relationship.
- Directed Association that is represented with a line with a normal arrowhead.
- An Association (Directed or not) contains an explicit association name. The Directed Association implies navigability from a Class B to a Class A in Figure 7.4.
- Navigability means that objects of Class B have the necessary attributes to know that they relate to objects of Class A while the reverse is not true: objects of Class A can exist without having references to objects of Class B.
- When the arrow starts and ends at the same class, then the class is associated to itself with a Reflexive Directed Association, which means that an object of this class is associated with objects of the same class with the specific named association.
- An arrow with a hollow triangle head and a dashed line represents the Realization relationship. This relationship represents a association between the class that specifies the functionality and the class that realizes the functionality.
- For example, Class A in Figure 7.4 specifies the functionality while Class B realizes it. Contain multiplicity information such as numbers (e.g "1"), ranges (e.g. "0_1", open ranges "1. . ._"), etc. in one or the other end of the relationship line/arrow.
- These multiplicities denote the potential number of class objects that are related to the other class object.
- For example, in Figure 7.4, a plain association called "association name," relates one (1) object of Class B with zero (0) or more objects from Class A. An asterisk "_" denotes zero (0) or more

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.

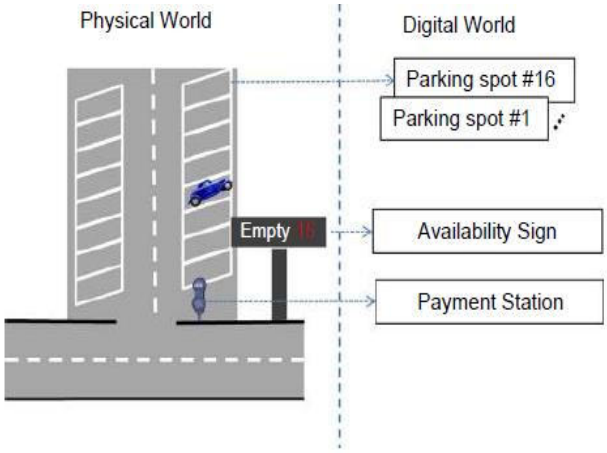


FIGURE 7.5

Physical vs. Virtual World.

- 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.
- 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.
- 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.
- Internet serves a rather virtual world of content and services (although these services are hosted on real physical machines)
- IoT is all about interaction through the Internet with physical Things.
- As interaction with the physical world is the key for the IoT; it needs to be captured in the domain model.
- A User and a Physical Entity are two concepts that belong to the domain model.
- 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).
- The objects, places, and things represented as Physical Entities are the same as Assets.
- 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.
- 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 (e.g. the parking spot availability).
- The Virtual Entity representation and the Physical Entity actual state should be synchronized.
- For example, a remotely controlled light (Physical Entity) represented by a memory location (Virtual Entity) in an application could be switched on/off by the User by changing the Virtual Entity representation, or in other words writing a value in the corresponding memory location.
- A Digital Artifact is an artifact of the digital world, and can be passive (e.g. a database entry) or active (e.g. application software).
- The model captures human-to-machine, application (active digital artifact)-to-machine, and M2M interaction when a digital artifact, and thus a User, interacts with a Device that is a Physical Entity.
- Physical Entities or their surrounding environment needs to be instrumented with certain kinds of Devices, or certain Devices need to be embedded/attached to the environment.
- 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 video camera can be 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.

- Any type of IoT Device needs to
  (a) have energy reserves (e.g. a battery)
  (b) be connected to the power grid
  (c) perform energy scavenging (e.g. converting solar radiation to energy).
- The Device communication, processing and storage, and energy reserve capabilities determine several design decisions such as if the resources should be on-Device or not.
- Resources are software components that provide data for, or are endpoints for, controlling Physical Entities.
- Resources can be of two types, on-Device resources and Network Resources.
- An on-Device Resource is typically hosted on the Device itself and provides information, or is the control point for the Physical Entities that the Device itself is attached to.
- An example is a temperature sensor on a temperature node deployed in a room that hosts a software component that responds to queries about the temperature of the room.
- The Network Resources are software components hosted somewhere in the network or cloud.

- A Virtual Entity is associated with potentially several Resources that provide information or control of the Physical Entity represented by this Virtual Entity.
- Resources can be of several types: sensor resources that provide sensor data, actuator resources that provide actuation capabilities or actuator state (e.g. "on"/"off"), processing resources that get sensor data as input and provide processed data as output, storage resources that store data related to Physical Entities, and tag resources that provide identification data of Physical Entities.
- IoT Services can be classified into three main classes according to their level of abstraction:

  1. Resource-Level Services typically expose the functionality of a Device by exposing the on-Device Resources. In addition, these services typically handle quality aspects such as security, availability, and performance issues. An example of such a Network Resource is a historical database of measurements of a specific resource on a specific Device.

  2. Virtual Entity-Level Services provide information or interaction capabilities about Virtual Entities, and as a result the Service interfaces typically include an identity of the Virtual Entity.

  3. Integrated Services are the compositions of Resource-Level and Virtual Entity-Level services, or any combination of both service classes.

Further considerations

- Identification of Physical Entities is important in an IoT system in order for any User to interact with the physical world though the digital world.
- Two ways to describe :
  (a) primary identification that uses natural features of a Physical Entity, and
  (b) secondary identification when using tags or labels attached to the Physical Entity.
- Both types of identification are modeled in the IoT Domain Model.
- Extracting natural features can be performed by a camera Device (Sensor) and relevant Resources that produce a set of features for specific Physical Entities.
- In physical spaces, a GPS Device or another type of location Device (e.g. an indoor location Device) can also be used to record the GPS coordinates of the space occupied by the Physical Entity.
- With respect to secondary identification, tags or labels attached to Physical Entities are modeled in the IoT Domain model, and there are relevant RFID or barcode technologies to realize such identification mechanisms.
- Apart from identification, location and time information are important for the annotation of the information collected for specific Physical Entities and represented in Virtual Entities.
- Information without one or the other (i.e. location or time) is practically useless apart from the case of Body Area Networks (BAN, networks of sensors attached to a human

body for live capture of vital signals, e.g. heart rate); that location is basically fixed and associated with the identification of the Human User.

- Nevertheless, in such cases, sometimes the location of the whole BAN or Human User is important for correlation purposes (e.g. upon moving outdoors, the Human User heart rate increases in order to compensate for the lower temperature than indoors).
- Therefore, the location, and often the timestamp of location, for the Virtual Entity can be modeled as an attribute of the Virtual Entity that could be obtained by location sensing resources (e.g. GPS or indoor location systems).

Communication model

- The communication model for an IoT Reference Model consists of the identification of the endpoints of interactions, traffic patterns (e.g. unicast vs. multicast), and general properties of the underlying technologies used for enabling such interactions.
- It is used to identification of the endpoints of the communication paths.
- The potential communicating endpoints or entities are the Users, Resources, and Devices from the IoT Domain Model.
- Users include Human Users and Active Digital Artifacts (Services, internal system components, external applications).
- Devices with a Human_Machine Interface mediate the interactions between a Human User and the physical world (e.g. keyboards, mice, pens, touch screens, buttons, microphones, cameras, eye tracking, and brain wave interfaces, etc.), and therefore the Human User is not a communication model endpoint.
- The User (Active Digital Artifact, Service)-to-Service interactions include the User-to-Service and Service-to-Service interactions as well as the Service_Resource_Device interactions.
- The User-to-Service and Service-to-Service communication is typically based on Internet protocols and one or both Services are hosted in Service-to-Service interactions on constrained/low-end Devices such as embedded systems.
- The communication model for these interactions includes several types of gateways (e.g. network, application layer gateways) to bridge between two or more disparate communication technologies.
- The Devices may be so constrained that they cannot host the Services, while the Resources could be hosted or not depending on the Device capabilities.
- This inability of the Device to host Resources or Services results in moving the corresponding Resources and/or Services out of the Device and into more powerful Devices or machines in the cloud.
- Then the Resource-to-Device or the Service-to-Resource communication needs to involve multiple types of communication stacks.

Functional model

- The IoT Functional Model aims at describing mainly the Functional Groups (FG) and their interaction with the ARM.
- Functional View of a Reference Architecture describes the functional components of an FG, interfaces, and interactions between the components.
- The Functional View is typically derived from the Functional Model in conjunction with high level requirements.

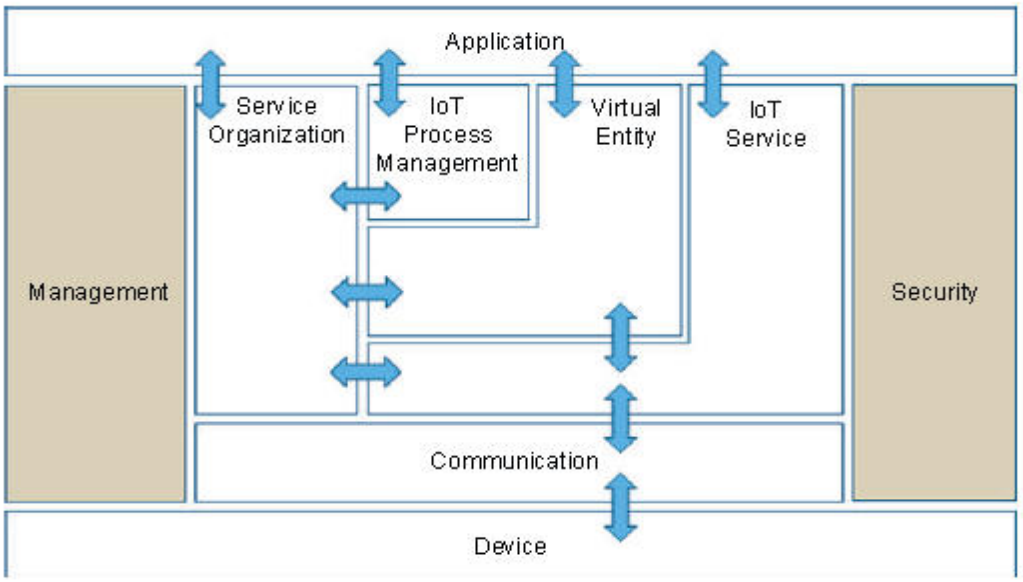


FIGURE 7.11

IoT-A Functional Model.

- The Application, Virtual Entity, IoT Service, and Device FGs are generated by starting from the User, Virtual Entity, Resource, Service, and Device classes from the IoT Domain Model.
- The need to compose simple IoT services in order to create more complex ones, as well as the need to integrate IoT services (simple or complex) with existing Information and Communications Technology (ICT) infrastructure, is the main driver behind the introduction of the Service Organization and IoT Process Management FGs respectively.
- All the above-mentioned FGs need to be supported by management and security functionality captured by the corresponding FGs.

Device functional group

- The Device FG contains all the possible functionality hosted by the physical Devices that are used for instrumenting the Physical Entities.
- This Device functionality includes sensing, actuation, processing, storage, and identification components, the sophistication of which depends on the Device capabilities.

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.
- Examples of such functions include wired bus or wireless mesh technologies through which sensor Devices are connected to Internet Gateway Devices.
- Communication technologies used between Applications and other functions such as functions from the IoT Service FG are out of scope because they are the typical Internet technologies.

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).
- Support functions such as directory services, which allow discovery of Services and resolution to Resources, are also part of this FG.

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.
- An example of a static association between Virtual Entities is the hierarchical inclusion relationship of a building, floor, room/corridor/open space, i.e. a building contains multiple floors that contain rooms, corridors, and open spaces.
- An example of a dynamic association between Virtual Entities is a car moving from one block of a city to another (the car is one Virtual Entity while the city block is another).
- A major difference between IoT Services and Virtual Entity Services is the semantics of the requests and responses to/from these services.
- The parking lot example, the Parking Sensor Service provides as a response only a number “0” or “1” given the identifier of a Loop Sensor (e.g. #11).
- The Virtual Entity Parking Spot #01 responds to a request about its occupancy status as “free.” The IoT Service provides data or information associated to specific Devices or Resources, including limited semantic information (e.g. Parking sensor #11, value5“0”, units 5 none); the Virtual IoT Service provides information with richer semantics (“Parking spot #01 is free”), and is closer to being human-readable and understandable.

IoT Service Organization functional group

- The purpose of the IoT Service Organization FG is to host all functional components that support the composition and orchestration of IoT and Virtual Entity services.
- This FG acts as a service hub between several other functional groups such as the IoT Process Management FG when, for example, service requests from Applications or the IoT Process Management are directed to the Resources implementing the necessary Services.

- Therefore, the Service Organization FG supports the association of Virtual Entities with the related IoT Services, and contains functions for discovery, composition, and choreography of services.
- Simple IoT or Virtual Entity Services can be composed to create more complex services, e.g. a control loop with one Sensor Service and one Actuator service with the objective to control the temperature in a building.
- Choreography is the brokerage of Services so that Services can subscribe to other services in a system.

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), Authorization of access to Services by Users, secure communication (ensuring integrity and confidentiality of messages) between entities of the system such as Devices, Services, Applications, assurance of privacy of sensitive information relating to Human Users.
- These include privacy mechanisms such as anonymization of collected data, anonymization of resource and Service accesses (Services cannot deduce which Human User accessed the data), and un-linkability (an outside observer cannot deduce the Human User of a service by observing multiple service requests by the same User).

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.
- An application can also be a part of a bigger ICT system that employs IoT services such as a supply chain system that uses RFID readers to track the movement of goods within a factory in order to update the Enterprise Resource Planning (ERP) system.

Modular IoT functions

- It is important to note that not all the FGs are needed for a complete actual IoT system.

- The Functional Model, as well as the Functional View of the Reference Architecture, contains a complete map of the potential functionalities for a system realization.
- The functionalities that will eventually be used in an actual system are dependent on the actual system requirements.
- FGs are organized in such a way that more complex functionalities can be built based on simpler ones, thus making the model modular.

Information model

- Information is defined as the enrichment of data (raw values without relevant or usable context) with the right context, so that queries about who, what, where, and when can be answered.
- IoT information model captures the details of a Virtual Entity centric model.

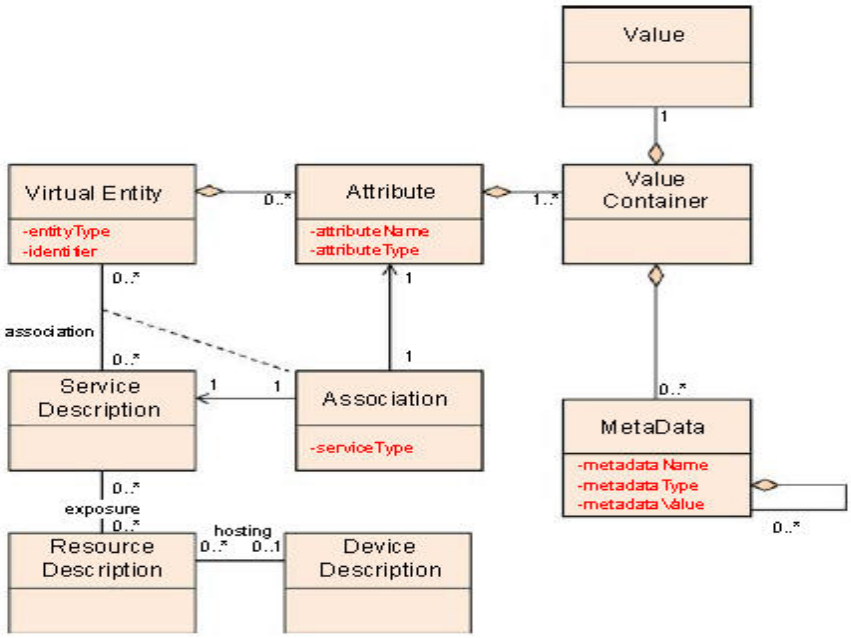


FIGURE 7.8

High-level IoT Information Model.

- Association class in Figure 7.8 contains information about the specific association between a Virtual Entity and a related Service.
- On a high-level, the IoT Information Model maintains the necessary information about Virtual Entities and their properties or attributes.
- These properties/attributes can be static or dynamic and enter into the system in various forms, e.g. by manual data entry or reading a sensor attached to the Virtual Entity.
- Virtual Entity attributes can also be digital synchronized copies of the state of an actuator.
- In the presentation of the high-level IoT information model, we omit the attributes that are not updated by an IoT Device (sensor, tag) or the attributes that do not affect any IoT

Device (actuator, tag), with the exception of essential attributes such as names and identifiers.

- Examples of omitted attributes that could exist in a real implementation are room names and floor numbers, in general, context information that is not directly related to IoT Devices, but that is nevertheless important for an actual system.
- The IoT Information Model describes Virtual Entities and their attributes that have one or more values annotated with meta-information or metadata.
- The attribute values are updated as a result of the associated services to a Virtual Entity.
- The associated services are related to Resources and Devices as seen from the IoT Domain Model.
- A Virtual Entity object contains simple attributes/properties:
  (a) entityType to denote the type of entity, such as a human, car, or room (the entity type can be a reference to concepts of a domain ontology, e.g. a car ontology);
  (b) a unique identifier; and
  (c) zero or more complex attributes of the class Attributes.
- The class Attributes should not be confused with the simple attributes of each class.
- This class Attributes is used as a grouping mechanism for complex attributes of the Virtual Entity.
- Objects of the class Attributes, in turn, contain the simple attributes with the self descriptive names attributeName and attributeType.
- The attribute type is the semantic type of the value (e.g. that the value is a temperature value), and can refer to an ontology such as the NASA quantities and units SWEET ontology (NASA JPL 2011).
- The Attribute class also contains a complex attribute ValueContainer that is a container of the multiple values that an attribute can take.
- The container includes complex attributes of the class Value and the class MetaData.
- The container contains exactly one value and meta-information (modeled as the class MetaData), such as a timestamp, describing this single value.
- Objects of the MetaData class can contain MetaData objects as complex attributes, as well as the simple attributes with the self-descriptive namesmetadataName, metadataType, and metadataValue.
- a Virtual Entity is associated with Resources that expose Services about the specific Virtual Entity.
- This association between a Virtual Entity and its Services is captured in the Information Model with the explicit class called Association.
- Objects of this class capture the relationship between objects of the complex Attribute class (associated with a Virtual Entity) and objects of the Service Description class.
- The class Association describes the relationship between a Virtual Entity and Service Description through the Attribute class, there is a dashed line between Association class and the line between the Virtual Entity and Service Description classes.

- The attribute serviceType can take two values:
  (a) "INFORMATION," if the associated service is a sensor service (i.e. allows reading of the sensor), or
  (b) "ACTUATION," if the associated serviceis an actuation service (i.e. allows an action executed on an actuator).
- In both cases, the eventual value of the attribute will be a result of either reading a sensor or controlling an actuator.

Example

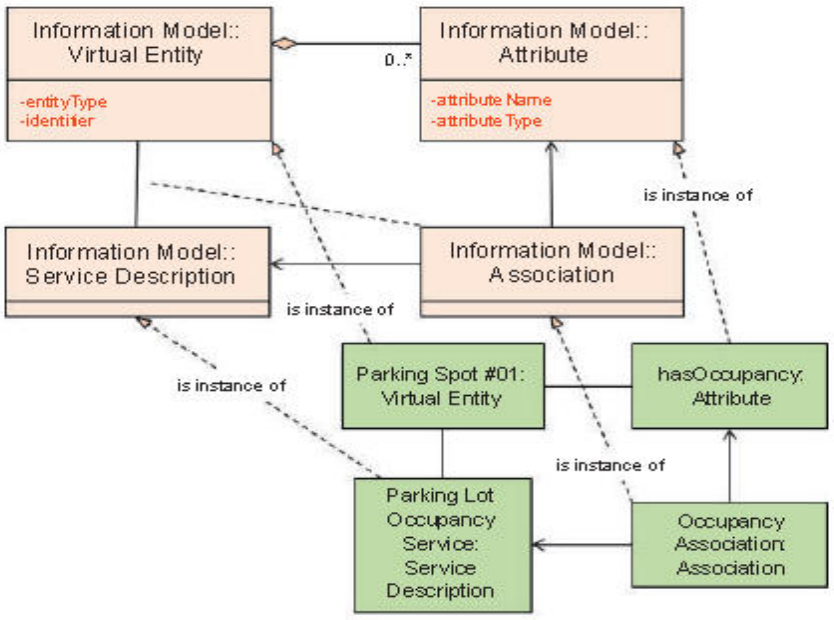


FIGURE 7.9
IoT Information Model example.

- Not show all the possible Virtual Entities, but only one corresponding to one parking spot.
- This Virtual Entity is described with one Attribute (among others) called hasOccupancy.
- This Attribute is associated with the Parking Lot Occupancy Service Description through the Occupancy Association.
- The Occupancy Association is the explicit expression of the association (line) between the Parking Spot #1 Virtual Entity and the Parking Lot Occupancy Service.
- The dashed arrows with hollow arrowheads represent the relationship "is instance of" for the information model, as opposed to the Realization relationship for the IoT Domain Model.

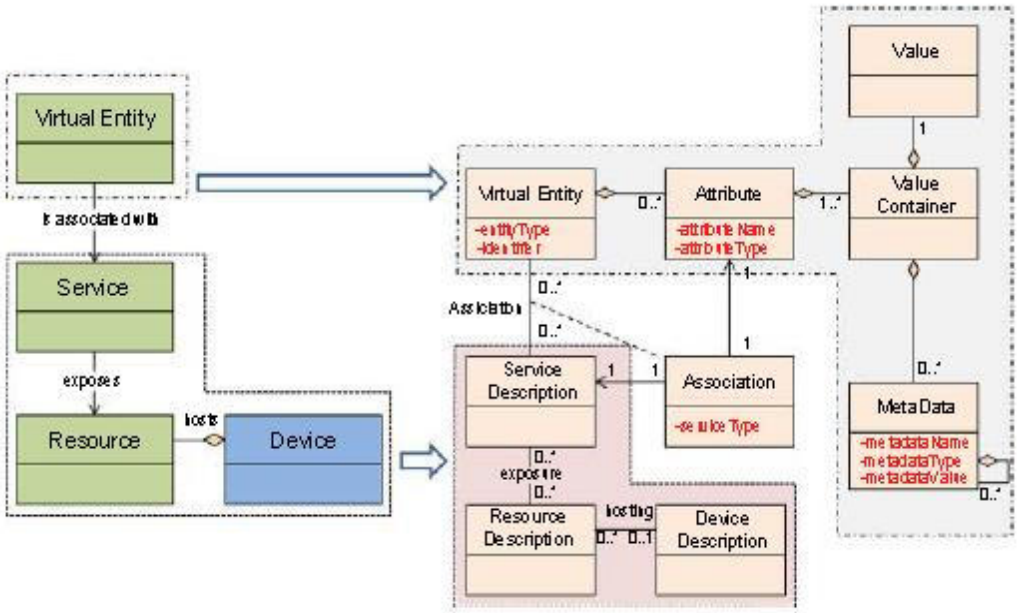


FIGURE 7.10

Relationship between core concepts of IoT Domain Model and IoT Information Model.

- Figure 7.10 presents the relationship between the core concepts of the IoT Domain Model and the IoT Information Model.
- The Information Model captures the Virtual Entity in the Domain Model being the "Thing" in the Internet of Things as several associated classes (Virtual Entity, Attribute, Value, MetaData, Value Container) that basically capture the description of a Virtual Entity and its context.
- The Device, Resource, and Service in the IoT Domain Model are also captured in the IoT Information Model because they are used as representations of the instruments and the digital interfaces for interaction with the Physical Entity associated with the Virtual Entity.
- The Information Model is a very high-level model, and omits certain details that could potentially be required in a concrete architecture and an actual system.
- These details could be derived by specific requirements from specific use cases describing the target actual system.
- Several other attributes or properties that could exist in a Virtual Entity description:

  1. Location and its temporal information are important because Physical Entities represented by Virtual Entities exist in space and time. These properties are extremely important when the interested Physical Entities are mobile (e.g. a moving car).A mobile Physical Entity affects the associations between Attributes and related Services, e.g. a person moving close to a camera (sensor) is associated with the Device, Resource, and Services offered by the camera for as long as she stays within the field of view of the camera. In such cases, the temporal availability of the associations between Attributes

and Services need to be captured, as availability denotes also temporal observability of the Virtual Entity.

2. **Even non-moving Virtual Entities contain properties that are dynamic with time**, and therefore their temporal variations need to be modeled and captured by an information model.

3. Information such as ownership is also important in commercial settings because it may determine access control rules or liability issues. It is important to note that the Attribute class is general enough to capture all the interested properties of a Physical Entity, and thus provides an extensible model whose details can only be specified by the specific actual system in mind.

- The Services in the IoT Domain Model are mapped to the Service Description in the IoT Information Model.
- The Service Description contains the following :

1. Service type, which denotes the type of service, such as Big Web Service or RESTful Web Service. The interfaces of a service are described based on the description language for each service type, for example, Web Application Description Language (WADL) for RESTful Web Services, Web Services Description Language (WSDL) for Big Web Services, Universal Service Description Language (USDL). The interface description includes, among other information, the invocation contact information, e.g. a Uniform Resource Locator (URL).

2. Service area and Service schedule are properties of Services used for specifying the geographical area of interest for a Service and the potential temporal availability of a Service, respectively. For sensing services, the area of interest is equivalent to the observation area, whereas for actuation services the area of interest is the area of operation or impact.

3. Associated resources that the Service exposes.

4. Metadata or semantic information used mainly for service composition. This is information such as the indicator of which resource property is exposed as input or output, whether the execution of the service needs any conditions satisfied before invocation.

- The IoT Information Model also contains Resource descriptions because Resources are associated with Services and Devices in the IoT Domain model.
- **A Resource description contains the following information**:

1. Resource name and identifier for facilitating resource discovery.
2. Resource type, which specifies if the resource is

(a) a sensor resource, which provides sensor readings;

(b) an actuator resource, which provides actuation capabilities (to affect the physical world) and actuator state;

(c) a processor resource, which provides processing of sensor data and output of processed data;

(d) a storage resource, which provides storage of data about a Physical Entity;
(e) a tag resource, which provides identification data for Physical Entities.
3. Free text attributes or tags used for capturing typical manual input such as "fire alarm, ceiling."
4. Indicator of whether the resource is an on-Device resour ce or network resource.
5. Location information about the Device that hosts this resource in case of an on-Device resource.
6. Associated Service information.
7. Associated Device description information.

- A Device is a Physical Entity that could have a sensor, actuator, or tag instantiation.

IoT reference architecture

- Architecture Reference Model (ARM) consists of two main parts:
  1. a Reference model
  2. a Reference Architecture.
- The foundation of an IoT Reference Architecture description is an IoT reference model.
- 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 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 7.2.

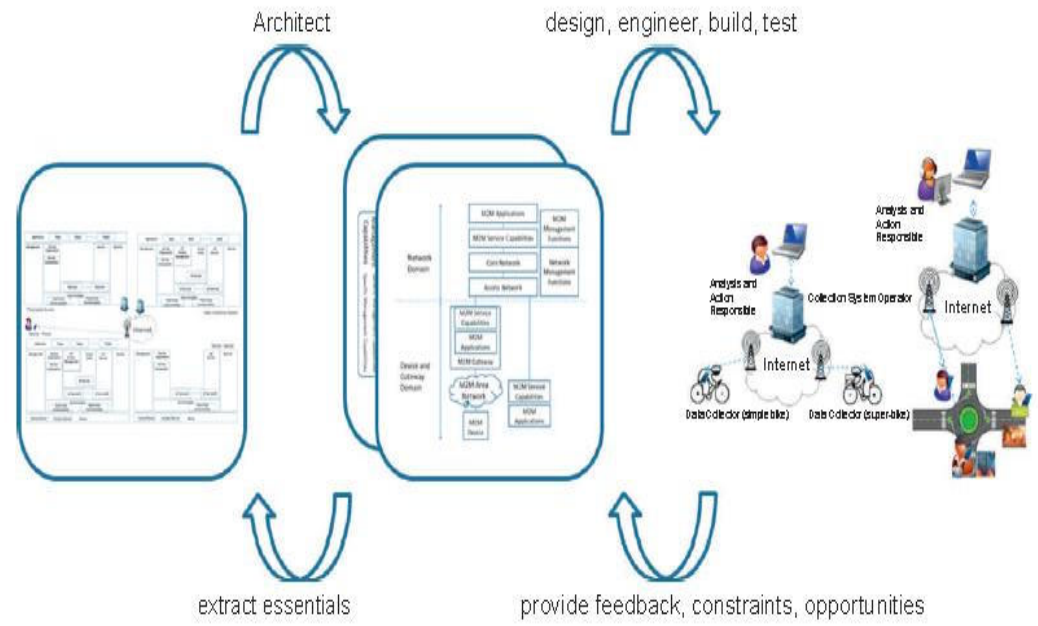


FIGURE 7.2

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.
- The general essentials out of multiple concrete architectures can then are aggregated, and contribute to the evolution of the Reference Architecture.

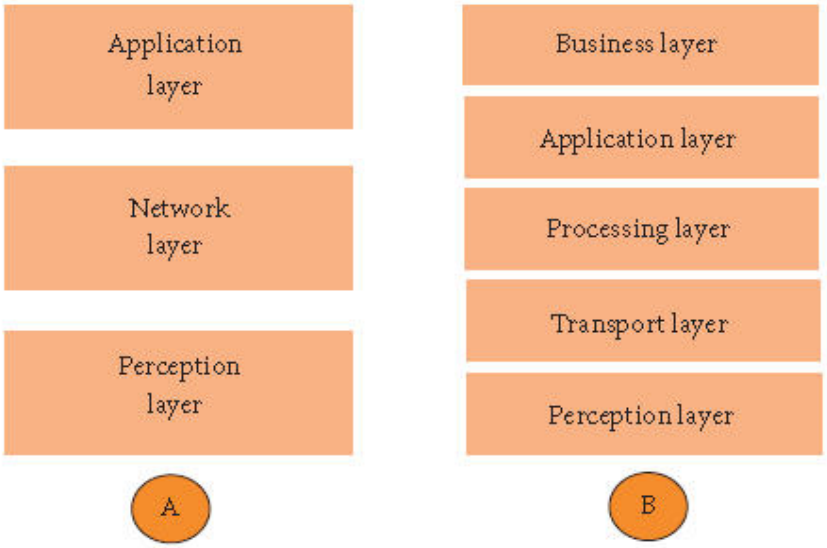


FIGURE 1: Architecture of IoT (A: three layers) (B: five layers).

- It has two types of Architecture:
  Three Layer Architectures
  Five-Layer Architectures

Three Layer Architectures

- It has three layers, namely, the perception, network, and application layers.

  (i) The *perception layer* is the physical layer, which has sensors for sensing and gathering information about the environment. It senses some physical parameters or identifies other smart objects in the environment.
  (ii) The *network layer* is responsible for connecting to other smart things, network devices, and servers. Its features are also used for transmitting and processing sensor data.
  (iii) The *application layer* is responsible for delivering application specific services to the user. It defines various applications in which the Internet of Things can be deployed, for example, smart homes, smart cities, and smart health.

- The three-layer architecture defines the main idea of the Internet of Things, but it is not sufficient for research on IoT because research often focuses on finer aspects of the Internet of Things.

Five Layer Architectures

- The five layers are perception, transport, processing, application, and business layers (see

➢ The role of the perception and application layers is the same as the architecture with three layers. We outline the function of the remaining three layers.

(i) The *transport layer* transfers the sensor data from the perception layer to the processing layer and vice versa through networks such as wireless, 3G, LAN, Bluetooth, RFID, and NFC.

(ii) The *processing layer* is also known as the middleware layer. It stores, analyzes, and processes huge amounts of data that comes from the transport layer. It can manage and provide a diverse set of services to the lower layers. It employs many technologies such as databases, cloud computing, and big data processing modules.

(iii) The *business layer* manages the whole IoT system, including applications, business and profit models, and users' privacy.

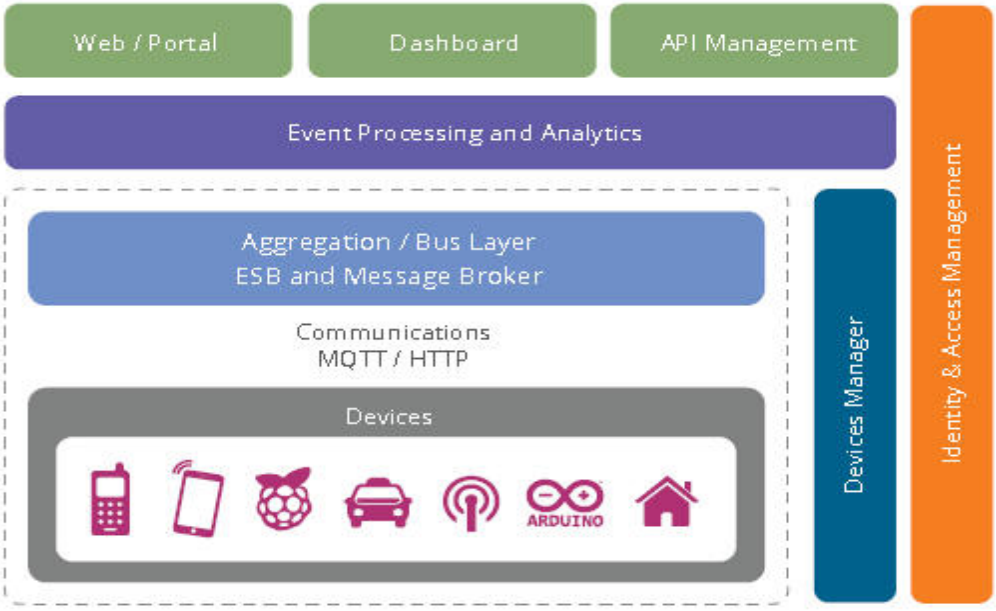


Figure 2. Reference architecture for IoT

➢ The layers are :
- ✓ Client/external communications - Web/Portal, Dashboard, APIs
- ✓ Event processing and analytics (including data storage)
- ✓ Aggregation/bus layer – ESB and message broker
- ✓ Relevant transports - MQTT/HTTP/XMPP/CoAP/AMQP, etc.
- ✓ Devices

➢ The cross-cutting layers are :
- ✓ Device manager
- ✓ Identity and access management

The Device Layer

- The bottom layer of the architecture is the device layer.
- Devices can be of various types, but in order to be considered as IoT devices, they must have some communications that either indirectly or directly attaches to the Internet.
- Examples of direct connections are :
  • Arduino with Arduino Ethernet connection
  • Arduino Yun with a Wi-Fi connection
  • Raspberry Pi connected via Ethernet or Wi-Fi
  • Intel Galileo connected via Ethernet or Wi-Fi Examples of indirectly connected device include
  • ZigBee devices connected via a ZigBee gateway
  • Bluetooth or Bluetooth Low Energy devices connecting via a mobile phone
  • Devices communicating via low power radios to a Raspberry Pi
- Each device typically needs an identity.
- The identity may be one of the following:
  • A unique identifier (UUID) burnt into the device
  • A UUID provided by the radio subsystem (e.g. Bluetooth identifier, Wi-Fi MAC address)
  • An OAuth2 Refresh/Bearer Token
  • An identifier stored in nonvolatile memory such as EEPROM

The Communications Layer

- The communication layer supports the connectivity of the devices.
- There are multiple potential protocols for communication between the devices and the cloud.

- The most well known three potential protocols are :

  • HTTP/HTTPS (and RESTful approaches on those)

  • MQTT 3.1/3.1.1

  • Constrained application protocol (CoAP)

- HTTP supports many libraries. Because it is a simple textbased protocol, many small devices such as 8-bit controllers can only partially support the .
- The larger 32-bit based devices can utilize full HTTP client libraries that properly implement the whole protocol.
- MQTT solve issues in embedded systems and SCADA.
- MQTT is a publish-subscribe messaging system based on a broker model. The protocol has a very small overhead.
- It is designed to support lossy and intermittently connected networks.
- MQTT was designed to flow over TCP.

- In addition there is an associated specification designed for ZigBee-style networks called MQTT-SN (Sensor Networks).
- CoAP is a protocol from the IETF that is designed to provide a RESTful application protocol modeled on HTTP semantics. CoAP is a more traditional client-server approach
- CoAP is designed to be used over UDP.

The Aggregation/Bus Layer

- This layer aggregates and brokers communications.
- This is an important layer for three reasons:

  1. The ability to support an HTTP server and/or an MQTT broker to talk to the devices

  2.The ability to aggregate and combine communications from different devices and to route communications to a specific device (possibly via a gateway)

  3.The ability to bridge and transform between different protocols, e.g. to offer HTTP based APIs that are mediated into an MQTT message going to the device.

- The bus layer may also provide some simple correlation and mapping from different correlation models (e.g. mapping a device ID into an owner's ID or vice-versa).
- It must be able to act as an OAuth2 Resource Server (validating Bearer Tokens and associated resource access scopes).
- It must also be able to act as a policy enforcement point (PEP) for policy-based access.

The Event Processing And Analytics Layer

- This layer takes the events from the bus and provides the ability to process and act upon these events.
- A core capability here is the requirement to store the data into a database.
- It has the following approaches:
  • Highly scalable, column-based data storage for storing events
  • Map-reduce for long-running batch-oriented processing of data
  • Complex event processing for fast in-memory processing and near real-time reaction and autonomic actions based on the data and activity of devices and other systems

Client/External Communications Layer

- The reference architecture needs to provide a way for these devices to communicate outside of the device-oriented system.
- This includes three main approaches.
  - Firstly, we need the ability to create web-based front-ends and portals that interact with devices and with the event-processing layer.

- Secondly, we need the ability to create dashboards that offer views into analytics and event processing.
- Finally, we need to be able to interact with systems outside this network using machine-to-machine communications (APIs).

- The API management layer provides three main functions:
  • The first is that it provides a developer-focused portal where developers can find, explore, and subscribe to APIs from the system. There is also support for publishers to create, version, and manage the available and published APIs;
  • The second is a gateway that manages access to the APIs, performing access control checks (for external requests) as well as throttling usage based on policies. It also performs routing and load-balancing;
  • The final aspect is that the gateway publishes data into the analytics layer where it is stored as well as processed to provide insights into how the APIs are used.

Device Management

- Device management (DM) is handled by two components.
- A server-side system (the device manager) communicates with devices via various protocols and provides both individual and bulk control of devices.
- It also remotely manages software and applications deployed on the device.
- It can lock and/or wipe the device if necessary.
- The device manager works in conjunction with the device management agents.
- There are multiple different agents for different platforms and device types.
- The device manager also needs to maintain the list of device identities and map these into owners.
- It must also work with the identity and access management layer to manage access controls over devices.
- There are three levels of device: non-managed, semi-managed and fully managed (NM, SM, FM).
- A full DM agent supports:
  • Managing the software on the device
  • Enabling/disabling features of the device (e.g. camera, hardware, etc.)
  • Management of security controls and identifiers
  • Monitoring the availability of the device
  • Maintaining a record of the device's location if available

Identity and Access Management

- The final layer is the identity and access management layer.
- This layer needs to provide the following services:

  • OAuth2 token issuing and validation

• Other identity services including SAML2 SSO and OpenID Connect support for identifying inbound requests from the Web layer

• XACML PDP

• Directory of users (e.g. LDAP)

• Policy management for access control (policy control point)

UNIT III IOT PROTOCOLS

Protocol Standardization for IoT – Efforts – M2M and WSN Protocols – SCADA and RFID Protocols – Issues with IoT Standardization – Unified Data Standards – Protocols – IEEE802.15.4–BACNet Protocol– Modbus –Zigbee– Network layer– 6LOWPAN-COAP-Security.

Protocol Standardization for IoT

Web of Things vs. Internet of Things

- TCP/IP
- HTML and HTTP

The difference between the Internet and the World Wide Web

- The Internet is the term used to identify the massive interconnection of computer networks around the world.
- It refers to the physical connection of the paths between two or more computers.

The World Wide Web is the general name for accessing the Internet via HTTP,

- The Internet is the large container, and the web is a part within the container.
- The key to make the Internet of Things (IoT) takes off is the Web of Things (WoT) – the killer applications platform or base of the IoT.
- The WoT is the next logical step in this IoT evolution toward global networks of sensors and actuators, enabling new applications and providing new opportunities.

The Internet Email FTP World Wide Web IM Telnet Gopher Chat P2P

The WoT explores the layer on top of connectivity with things and addresses issues such as fast prototyping, data integration, and interaction with objects.

- The WoT is a version where things become seamlessly integrated into the web .
- There are also many other WoT applications around the world. Some of the WoT applications are listed here.

- Arduino
- Japan Geiger Map
- Nanode
- The National Weather Study Project
- AgSphere

Two Pillars of the Web

• The application server became the foundation that helped build widely spreading web-based applications. An application server is a software framework or middleware that provides an environment in which applications can run, no matter what the applications are or what they do. An application server acts as a set of components accessible to the software developer through an API defined by the middleware itself. For web applications, these components are usually performed in the same machine where the web server is running, and their main job is to support the construction of dynamic web pages. However, present-day application servers target much more than just web page generation: they implement services like clustering, fail-over, and load balancing, so developers can focus on implementing the business logic.

• The application server is based on the three-tiered (Fig.) or multi-tiered software architecture. The multitier architecture is a client–server architecture in which the presentation, the application processing, and the data management are logically separate processes, which is important for distributed web applications. For example, a web application that uses middleware to service data requests between a user and a database employs multitier architecture.

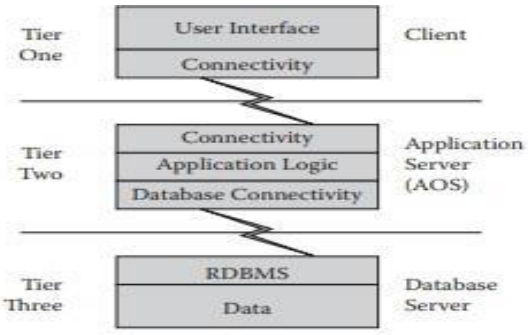
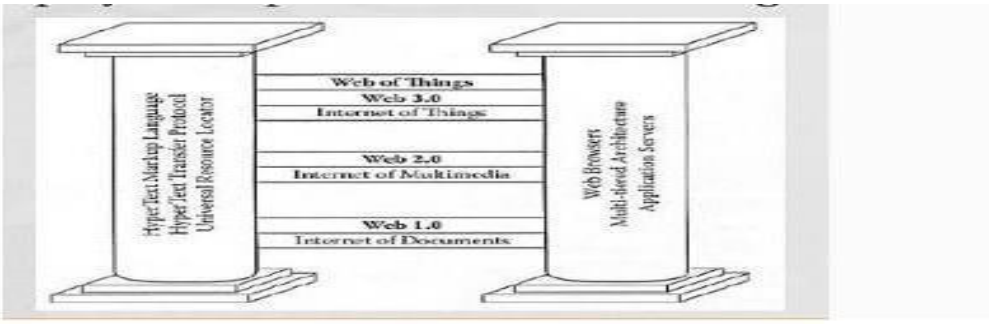


Figure 6.2 Three-tiered architecture.

• As the two pillars for web applications and the Internet revolution, the protocols (HTML)/HTTP/URL and the software will continue to be the two pillars of play an important role in building WoT applications.



Protocol Standardization Efforts:

The current status of IoT standardization

• Fragmented architectures, no coherent unifying concepts, solutions exist only for application silos.

• No holistic approach to implement the IoT has yet been proposed.

• Many island solutions do exist (RFID, sensor nets, etc.)

• Little cross-sector reuse of technology and exchange of knowledge.

The key objectives of the IoT-A consortium are as follows:

• Create the architectural foundations of an interoperable Internet of Things as a key dimension of the larger future Internet.

• Architectural reference model together with an initial set of key building blocks:

• Not reinventing the wheel but federating already existing technologies

• Demonstrating the applicability in a set of use cases

• Removing the barriers of deployment and wide-scale acceptance of the IoT by establishing a strongly involved stakeholder group

- Federating heterogeneous IoT technologies into an interoperable IoT fabric

IPSO (Internet Protocol for Smart Objects, http://www.ipsoalliance.org/) Alliance, formed in 2008, is another effort aiming 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 (Figure 6.7)

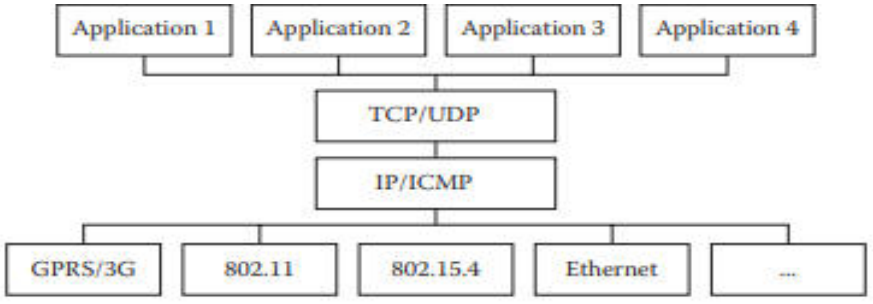


Figure 6.7 All-IP networks.

• The emergency application space for smart objects require scalable and interoperable communication mechanisms that support future innovation as the application space grows. A smart object is defined by IPSO (Internet Protocol Smart Object) as

- An intelligent (RFID) tag
- A sensor: device that measures a physical quantity and converts it to an analog or digital signal, such as power consumption and quality, vibration of an engine, pollution, motion detection, temperature

- An actuator: device that controls a set of equipment, such as controls and/or modulates the flow of a gas or liquid, controls electricity distribution, performs a mechanical operation
- An embedded device: a purpose-built connected device that performs a specific function, such as a factory robotic arm, vending machine, smart grid analyzer
- Any combination of the above features to form a more complex entity

M2M

Machine to machine (M2M) is direct communication between devices using any communications channel, including wired and wireless.-M2M with Internet protocols could be considered a subset of the Internet of Things and understood from a more vertical and closed point of view. On the other hand, the Internet of Things encompasses a more horizontal and meaningful approach where vertical applications arepulled together to address the needs of many people.

Machine-to-Machine (M2M)

•1.1 Billion smart phones

•244 Million smart meters

•487 Million e-readers and tablets

•2.37 Billion networked office devices

•86 Million medical devices

•45 Million connected automobiles

•547 Million connected appliances

•105 Million connected military devices

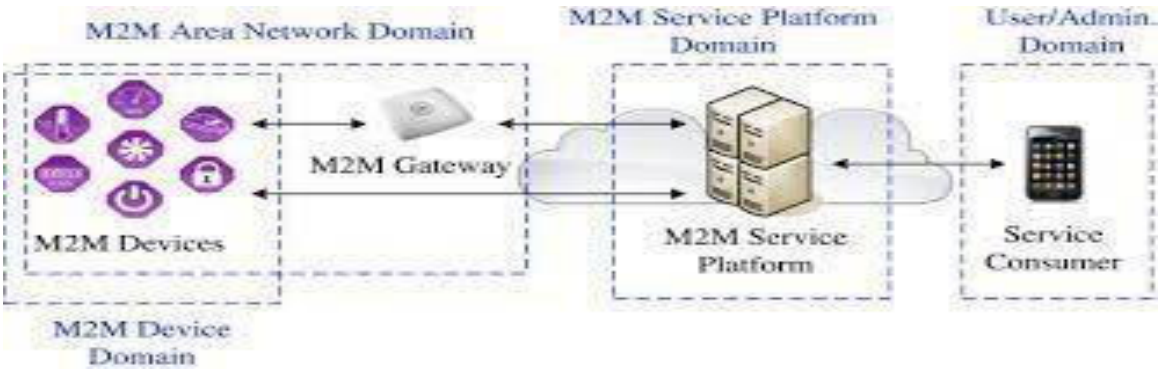
•431 Million information technology devices

•45 Million supervisory control and data acquisition (SCADA)

•5+ Billion other (non-phone/tablet/e-reader) electronic devices

How M2M works

The main purpose of machine-to-machine technology is to tap into sensor data and transmit it to a network. Unlike SCADA or other remote monitoring tools, M2M systems often use public networks and access methods -- for example, cellular or Ethernet -- to make it more cost-effective.



The main components of an M2M system include sensors, RFID, a Wi-Fi or cellular communications link, and autonomic computing software programmed to help a network device interpret data and make decisions. These M2M applications translate the data, which can trigger preprogrammed, automated actions

One of the most well-known types of machine-to-machine communication is telemetry, which has been used since the early part of the last century to transmit operational data. Pioneers in telemetrics first used telephone lines, and later, radio waves, to transmit performance measurements gathered from monitoringinstruments in remote locations.

The Internet and improved standards for wireless technology have expanded the role of telemetry from pure science, engineering and manufacturing to everyday use in products such as heating units, electricmeters and internet-connected devices, such as appliances.

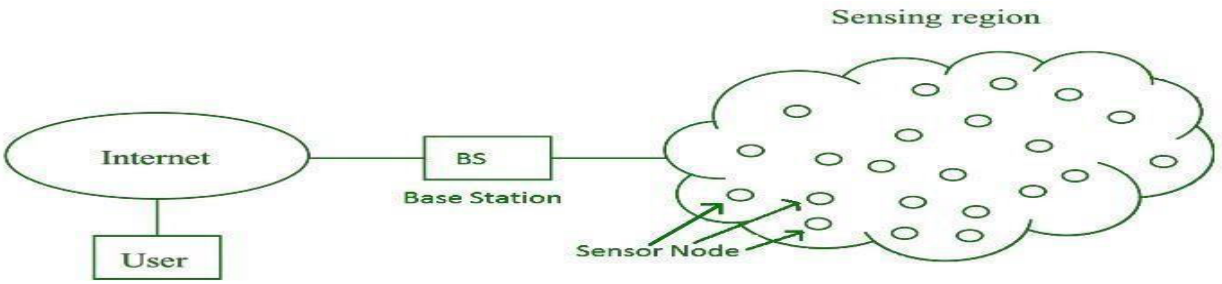
Beyond being able to remotely monitor equipment and systems, the top benefits of M2M include:

- reduced costs by minimizing equipment maintenance and downtime;
- boosted revenue by revealing new business opportunities for servicing products in the field; and
- improved customer service by proactively monitoring and servicing equipment before it fails or onlywhen it is needed

Wireless Sensor Network (WSN) is 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.

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.

Applications of WSN:

1. Internet of Things (IOT)
2. Surveillance and Monitoring for security, threat detection
3. Environmental temperature, humidity, and air pressure
4. Noise Level of the surrounding
5. Medical applications like patient monitoring
6. Agriculture
7. Landslide Detection

Challenges of WSN:

1. Quality of Service
2. Security Issue
3. Energy Efficiency
4. Network Throughput
5. Performance
6. Ability to cope with node failure
7. Cross layer optimisation
8. Scalability to large scale of deployment

Components of WSN:

- **Sensors:**

  Sensors in WSN are used to capture the environmental variables and which is used for data acquisition. Sensor signals are converted into electrical signals.

- **Radio Nodes:**

  It is used to receive the data produced by the Sensors and sends it to the WLAN access point. It consists of a microcontroller, transceiver, external memory, and power source.

- **WLAN Access Point:**

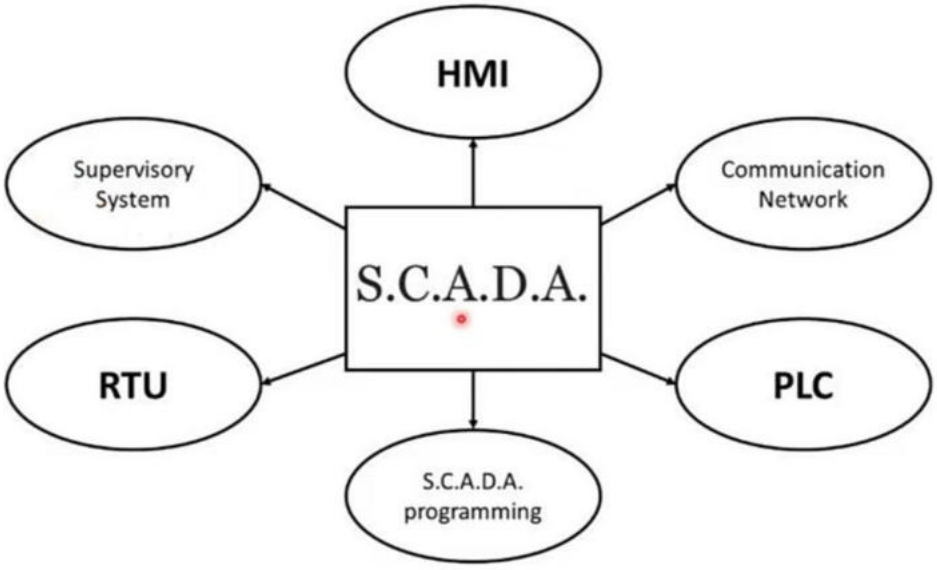
  It receives the data which is sent by the Radio nodes wirelessly, generally through theinternet.

- **Evaluation Software:**

  The data received by the WLAN Access Point is processed by a software called as Evaluation Software for presenting the report to the users for further processing of the data which can be used for processing, analysis, storage, and mining of the data.

What is SCADA?

SCADA – Supervisory Control and Data Acquisition – is an automated software control system that monitors industrial control systems (ICS) and provides data insights to industrial supervisors about the condition of the entire operation. While a SCADA system is automated, the data output is monitored byhuman eyes in a separate control room through a graphical user interface.



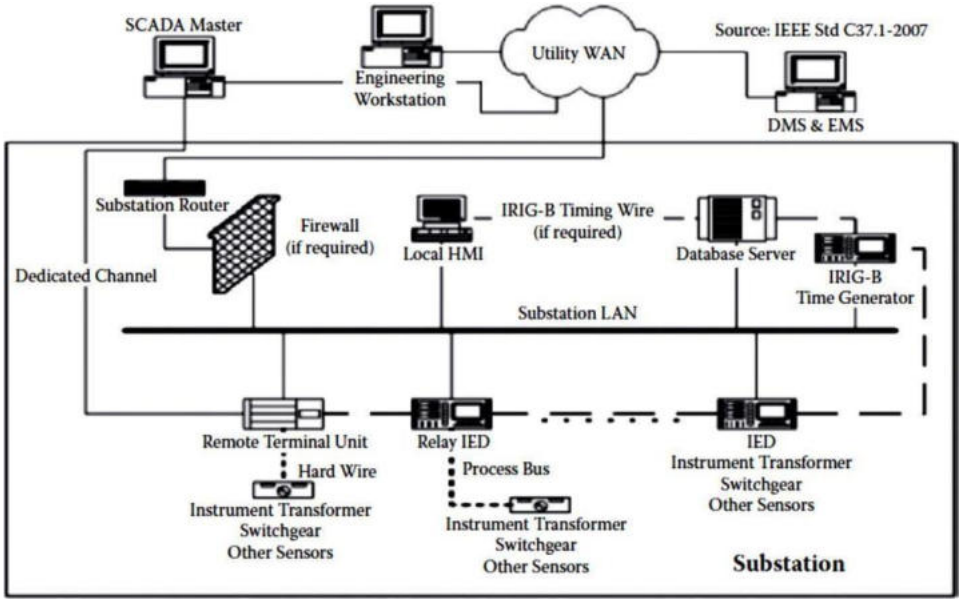
Supervisory control and data acquisition (SCADA)

Supervisory control and data acquisition (**SCADA**) is a control system architecture comprising computers, networked data communications and graphical user interfaces for high-level supervision of machines and processes. It also covers sensors and other devices, such as programmable logic controllers, which interface with process plant or machinery.

How SCADA systems work?

Business decisions are best made when driven by accurate data. In the industrial world, that critical data is collected by sensors, controllers, and other types of IoT-enabled tech that display real-time data overa network. This data – and the network – is monitored by SCADA software.

SCADA systems display operating processes to the human controller, including status messages and alerts. SCADA systems operate devices through switches that turn them on or off. This is an automatic process; however, sometimes human intervention is required.



Human operators can use the interface to make changes to the operating processes in real-time.

These changes might include turning valves on/off and changing the temperature of a thermostat. A control unit connected to the SCADA system collects and passes data over the network where it's collected for making data-driven decisions.

Once a large amount of data has been collected, the system converts that data into a more useful format. Since the data is being collected from multiple sensors, automation (including a Human-Machine Interface)is used to gather and process the data quickly and efficiently.

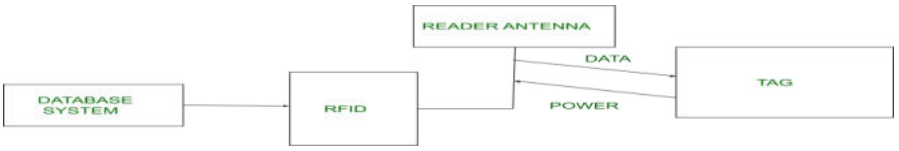
Key tasks performed by IoT industrial controls SCADA

- A SCADA system feeds data to a graphical user interface or dashboard. That data is then monitored byhumans.
- IoT devices connected to the network are monitored.

- Collected data is generally stored in a cloud server.
- SCADA sensors detect small changes in the environment including temperature and sound.

Radio Frequency Identification (RFID)

Radio Frequency Identification (RFID) is a method that is used to track or identify an object by radio transmission uses over the web. Data digitally encoded in an RFID tag which might be read by the reader. This device work as a tag or label during which data read from tags that are stored in the database through the reader as compared to traditional barcodes and QR codes. It is often read outside the road of sight either passive or active RFID.



Kinds of RFID :

There are many kinds of RFID, each with different properties, but perhaps the most fascinating aspect of RFID technology is that most RFID tags have neither an electric plug nor a battery. Instead, all of the energy needed to operate them is supplied in the form of radio waves by RFID readers. This technology is called passive RFID to distinguish it from the(less common) active RFID in which there is a power source on the tag.

UHF RHID (Ultra-High Frequency RFID). It is used on shipping pallets and some driver's licenses. Readers send signals in the 902-928 MHz band. Tags communicate at distances of several meters by changing the way they reflect the reader signals; the reader is able to pick up these reflections. This wayof operating is called backscatter.

HF RFID (High-Frequency RFID). It operates at 13.56 MHz and is likely to be in your passport, credit cards, books, and noncontact payment systems. HF RFID has a short-range, typically a meter or less because the physical mechanism is based on induction rather than backscatter.

There are also other forms of RFID using other frequencies, such as LF RFID(Low-Frequency RFID),which was developed before HF RFID and used for animal tracking

There are two types of RFID :

1. **Passive RFID –**

In this device, RF tags are not attached by a power supply and passive RF tag stored their power. When it is emitted from active antennas and the RF tag are used specific frequency like 125 - 134MHZ as low frequency, 13.56MHZ as a high frequency and 856MHZ to 960MHZ as ultra-high frequency.

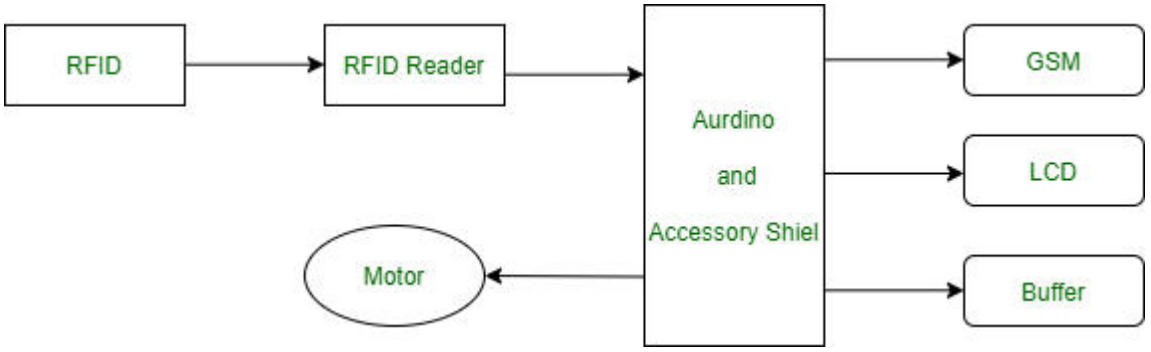
2. Active RFID –

In this device, RF tags are attached by a power supply that emits a signal and there is an antenna which receives the data.

Working Principle of RFID :

Generally, RFID uses radio waves to perform AIDC function. AIDC stands for Automatic Identification and Data Capture technology which performs object identification and collection and mapping of the data.

An antenna is an device which converts power into radio waves which are used for communication between reader and tag. RFID readers retrieve the information from RFID tag which detects the tag and reads or writes the data into the tag. It may include one processor, package, storage and transmitter and receiver unit.



Features of RFID :

- An RFID tag consists of two-part which is an microcircuit and an antenna.
- This tag is covered by protective material which acts as a shield against the outer environment effect.
- This tag may active or passive in which we mainly and widely used passive RFID.

Application of RFID :

- It utilized in tracking shipping containers, trucks and railroad, cars.
- It uses in Asset tracking.
- It utilized in credit-card shaped for access application.
- It uses in Personnel tracking.
- Controlling access to restricted areas.

- It uses ID badging.
- Supply chain management.
- Counterfeit prevention (e.g., in the pharmaceutical industry).

Advantages of RFID :

- It provides data access and real-time information without taking to much time.
- RFID tags follow the instruction and store a large amount of information.
- The RFID system is non-line of sight nature of the technology.
- It improves the Efficiency, traceability of production.
- In RFID hundred of tags read in a short time.

Disadvantages of RFID :

- It takes longer to program RFID Devices.
- RFID intercepted easily even it is Encrypted.
- In an RFID system, there are two or three layers of ordinary household foil to dam the radio wave.
- There is privacy concern about RFID devices anybody can access information about anything.
- Active RFID can costlier due to battery.

Issue with iot standardization

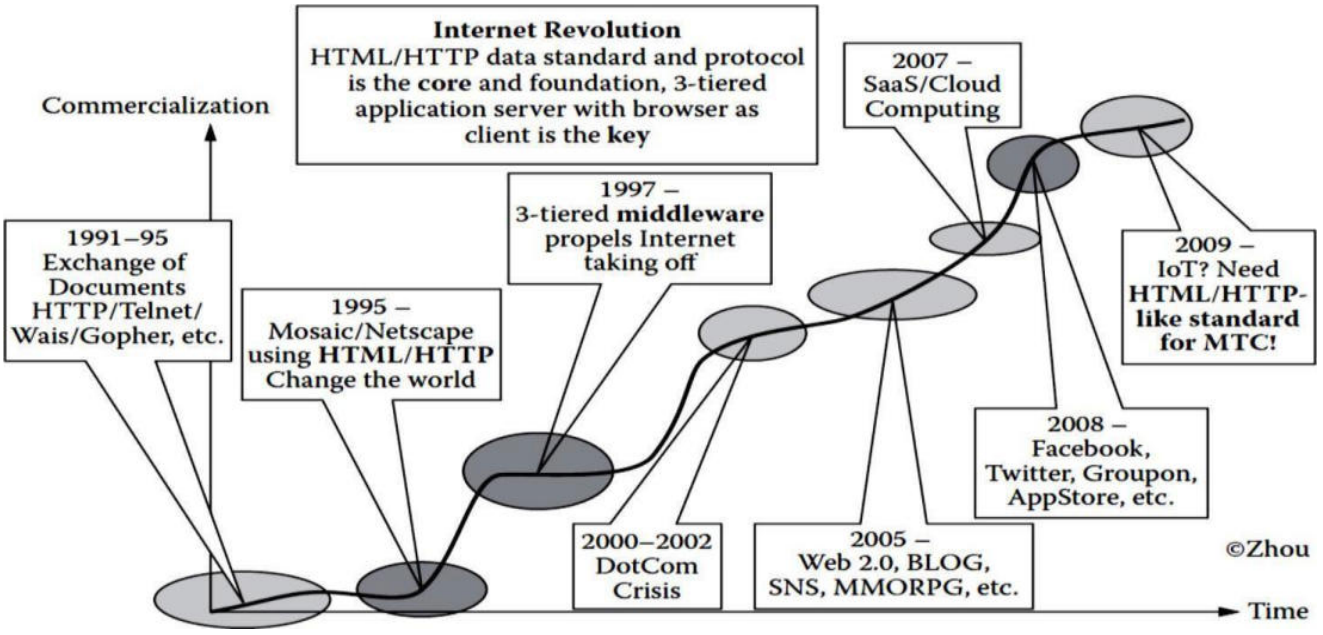
- It should be noted that not everything about standardization is positive
- Standardization is like a double-edged sword:
- Critical to market development
- But it may threaten innovation and inhibit change whenstandards are accepted by the market
- Standardization and innovation are like yin & yang
- They could be contradictory to each other in somecases, even though this observation isdebatable
- 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 middlewarecovers onlyRFID events
- Even within same segment, there are more thanone consortium or forum doing standardization without enough communication with each other
- Some are even competing with each other
- Some people believe that the IoT concept is wellestablished
- 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 maynever have answers:

- ICT standardization is a highly decentralized activity. How can the individual activities of the network of extremelyheterogeneous standards-setting bodies be coordinated?
- 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?

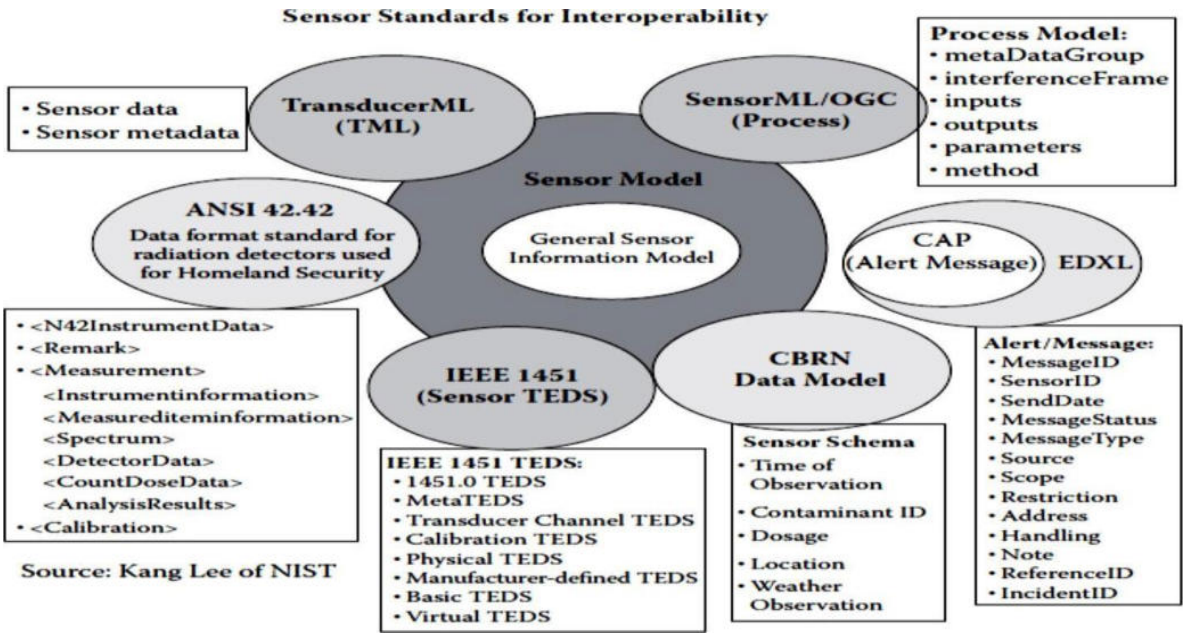
UNIFIED DATA STANDARDS:

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



- Many standardization efforts have been trying to define unified data representation, protocolfor IoT
- Before IoT, Internet was actually an Internet of documents or of multimediadocuments
- 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?

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



Protocol IEEE 802.15.4

IEEE 802.15.4 is a low-cost, low-data-rate wireless access technology for devices that are operated or work on batteries. This describes how low-rate wireless personal area networks (LR-WPANs) function.

Properties:

1. Standardization and alliances: It specifies low-data-rate PHY and MAC layer requirements forwireless personal area networks (WPAN).

IEEE 802.15. Protocol Stacks include:

- **ZigBee:** ZigBee is a Personal Area Network task group with a low rate task group 4. It is a technology of home networking. ZigBee is a technological standard created for controlling and sensing the network. As we know that ZigBee is the Personal Area network of task group 4 so it isbased on IEEE 802.15.4 and is created by Zigbee Alliance.

- **6LoWPAN:** The 6LoWPAN system is used for a variety of applications including wireless sensor networks. This form of wireless sensor network sends data as packets and uses IPv6 – providing thebasis for the name – IPv6 over Low power Wireless Personal Area Networks.
- **ZigBee IP:** Zigbee is a standards-based wireless technology that was developed for low-cost and low-power wireless machine-to-machine (M2M) and internet of things (IoT) networks.
- **ISA100.11a:** It is a mesh network that provides secure wireless communication to process control.
- **Wireless HART:** It is also a wireless sensor network technology, that makes use of time-synchronized and self-organizing architecture.
- **Thread:** Thread is an IPv6-based networking protocol for low-power Internet of Things devices in IEEE 802.15. 4-2006 wireless mesh network. Thread is independent.

2. Physical Layer: This standard enables a wide range of PHY options in ISM bands, ranging from 2.4 GHz to sub-GHz frequencies. IEEE 802.15.4 enables data transmission speeds of 20 kilobits per second,40 kilobits per second, 100 kilobits per second, and 250 kilobits per second.

3. MAC layer: The MAC layer provides links to the PHY channel by determining that devices in the same region will share the assigned frequencies. The scheduling and routing of data packets are also managed at this layer. The 802.15.4 MAC layer is responsible for a number of functions like:

- Beaconing for devices that operate as controllers in a network.
- used to associate and dissociate PANs with the help of devices.
- The safety of the device.
    - Consistent communication between two MAC devices that are in a peer-to-peer relationship.

Several established frame types are used by the MAC layer to accomplish these functions. In 802.15.4,there are four different types of MAC frames:

- frame of data
- Frame for a beacon
- Frame of acknowledgement
- Frame for MAC commands

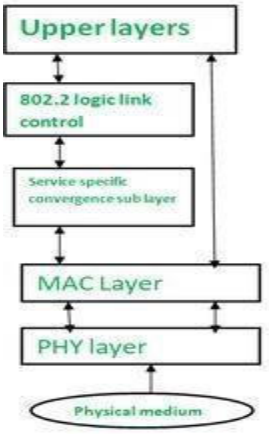
4. Topology: Networks based on IEEE 802.15.4 can be developed in a star, peer-to-peer, or mesh topology. Mesh networks connect a large number of nodes. This enables nodes that would otherwise beout of range to interact with each other to use intermediate nodes to relay data.

5. Security: For data security, the IEEE 802.15.4 standard employs the Advanced Encryption Standard (AES) with a 128-bit key length as the basic encryption technique. Activating such security measures for 802.15.4 significantly alters the frame format and uses a few of the payloads. The very first phase in activating AES encryption is to use the Security Enabled field in the Frame Control part

of the 802.15.4 header. For safety, this field is a single bit which is assigned to 1. When this bit is set, by taking certain bytes from its Payload field, a field known as the Auxiliary Security Header is formed following the Source Address field.

6. Competitive Technologies: The IEEE 802.15.4 PHY and MAC layers serve as a basis for a variety of networking profiles that operate in different IoT access scenarios. DASH7 is a competing radio technology with distinct PHY and MAC layers.

The architecture of LR-WPAN Device:



IEEE 802.15.4

Advantages of IEEE 802.15.4:

IEEE 802.15.4 has the following advantages:

- cheap cost
- long battery life,
- Quick installation
- simple
- extensible protocol stack

Disadvantages of IEEE 802.15.4:

IEEE 802.15.4's drawbacks include:

- IEEE 802.15.4 causes interference and multipath fading.
- doesn't employ a frequency-hopping approach.
- unbounded latency
- interference susceptibility

Applications of IEEE 802.15.4:

IEEE 802.15.4 Applications:

- Wireless sensor networks in the industry
- Building and home automation
- Remote controllers and interacting toys
- Automotive networks

BACnet Protocol: Architecture, Working, Types, Objects & Its Applications

BACnet protocol was developed by a committee named ASHRAE or the American Society of Heating, Refrigerating & Air-Conditioning Engineers in 1987. The main motto of this committee is to make a protocol that would provide systems from various manufacturers to communicate together in a pleasant way. So this protocol is a registered brand of ASHRAE. Since the time protocol was developed it is undergoing continuous changes with an open agreement procedure. So that all interested parties are welcome to participate with no fees.

What is BACnet Protocol?

A data communication protocol that is used to build an automated control network, is known as BACnet or Building Automation Control Network. This data communication protocol is both an ISO & ANSI standard used for interoperability between cooperating building automation devices. Bacnet Protocol includes a set of rules for governing the data exchange on a computer network that simply covers all from what type of cable to utilize, to form a particular command or request in a normal way.

To attain interoperability across a broad spectrum of equipment, the BACnet specification includes three major parts. Primary, Secondary, and tertiary. So the primary part defines a technique to represent any kind of building automation apparatus in a normal way.

The secondary part describes messages that can be transmitted across a network of computers to check and manage such equipment. The final part describes a set of suitable LANs which are used for conveying BACnet communications.

Why is Bacnet Protocol required?

The **BACnet protocol's importance** is to define typical techniques that manufacturers can execute to build components as well as systems that are interoperable through other components & systems of BACnet.

It also specifies how data is signified on the network as well as the services that are utilized to transmit data from one node of BACnet to another node. It also has messages that recognize network & data nodes.

BACnet protocol is used in all types of automated building systems. So, there are interoperable products available within different categories like security, fire, lighting, elevators, HVAC, etc. This protocol simply addresses the interoperability goal through simply defining a general working model of automation devices, a technique used for defining the data that they include, & also a technique used for explaining protocols that a single device can utilize to inquire one more device to execute some preferredaction.

Characteristics of BACnet include:

- Open source standard
- No license fee required for its implementation
- A large number of manufacturer have adopted these standards, making it less dependent on a specific vendor for its implementation

BACNet Communication

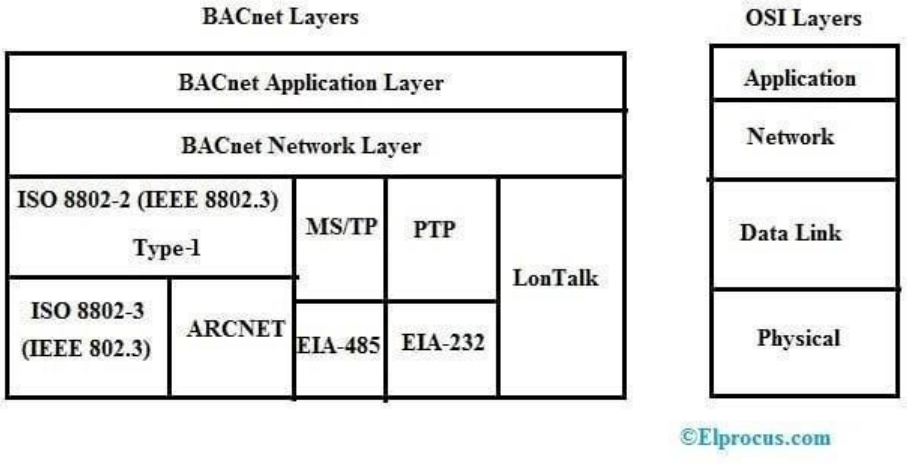
BACnet's communication model is the Client-Server model. It deals with the sensing of messages between the controller and different devices. In this model, when there is a data requirement data, the client sends a request to the server. The server, in turn, responds with the required information, called a server response.

There are 38 types of services a BACNet offers divided into:

- Alarm and Event Services
- File Access Services
- Object Access Services
- Remote Device Management Services
- Virtual Terminal Services

Bacnet Protocol Architecture

The BACnet protocol architecture is predominately restricted to lighting controls, HVAC & gateways. This protocol highlights lightweight and efficient communication which is optimized for short messages, small networks, and inter-networks.



Bacnet Protocol Architecture

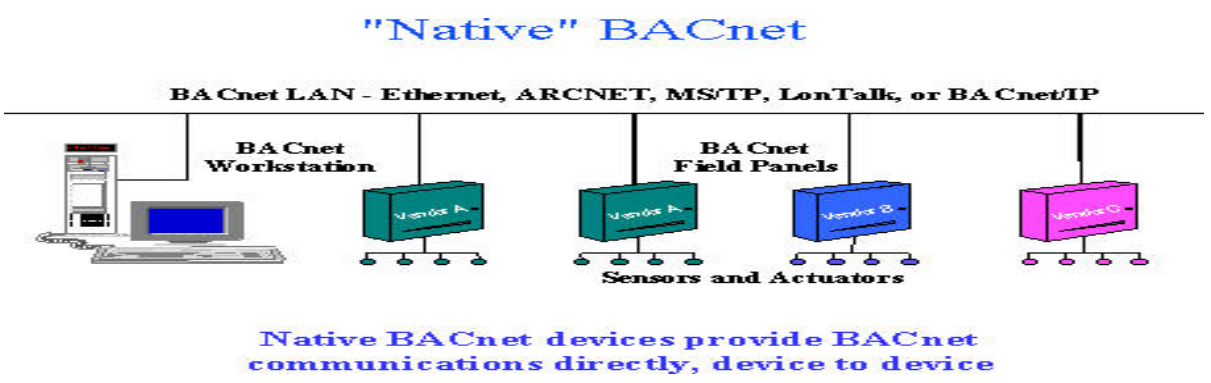
BACnet protocol architecture is a collapsed architecture that matches to 4-layers of the OSI model. The four layers in the BACnet architecture mainly include Application, Network, Data Link & Physical.Even though, just the Network layer & Application layer are simply BACnet.

Network Technologies

The final step in implementing BACnet is the selection of a network technology through which all of the information travels to and from designated destinations. Some of the network technologies used include:

- ARCnet
- MS/TP
-
- Ethernet
- IP
- Lon Talk
- ZigBee

The devices that use BACnet are called **Native BACnet** , which means that devices generate signals that are compatible with the BACnet technology, and can be interconnected with any BACnet network



BACnet IP

IP is one of the networking technologies that BACnet supports. Support of BACnet-compatible devices can be connected and utilized over an IP network, called BACnet IP. BACnet IP represents a network that uses IP as its networking technique and devices that support IP as a means of communication can be used in the BACnet.

Many companies have their networking technology in the IP structure, even when not connected to the internet, making it easy and cost-effective to implement a BACNet structure.

The devices on the BACNet IP have a unique IP address. The IP address is understandable and recognizable on the BACnet.

Some advantages of BACNet IP are:

- IP based on the Ethernet structure offers a fast response speed. Normally, Ethernet supports up to 100 megabits per second.
- A large number of devices can be connected to each other on the IP.
- Easy troubleshooting.

What Is the Modbus Protocol?

Modbus is a request-response protocol implemented using a master-slave relationship. In a master- slave relationship, communication always occurs in pairs—one device must initiate a request and then wait for a response—and the initiating device (the master) is responsible for initiating every interaction. Typically, the master is a human machine interface (HMI) or Supervisory Control and Data Acquisition (SCADA) system and the slave is a sensor, programmable logic controller (PLC), or programmable automation controller (PAC). The content of these requests and responses, and the network layers across which these messages are sent, are defined by the different layers of the protocol.

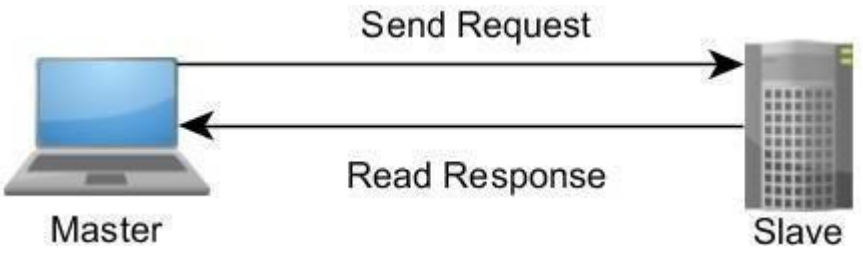


Figure 1. *A Master-Slave Networking Relationship*

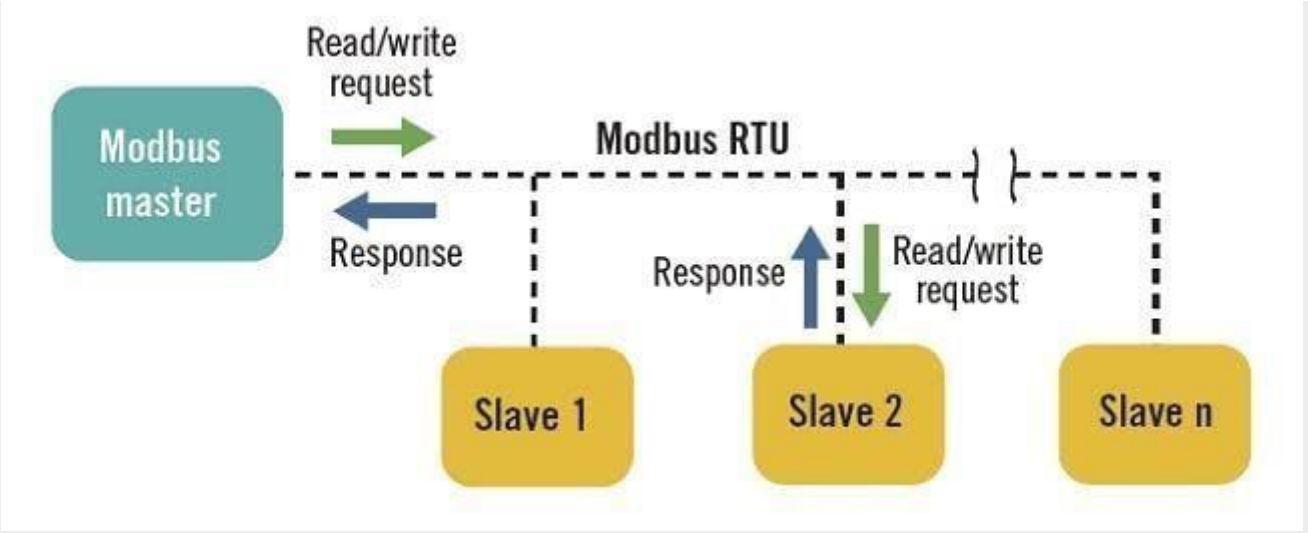
Layers of the Modbus Protocol

In the initial implementation, Modbus was a single protocol built on top of serial, so it could not be divided into multiple layers. Over time, different application data units were introduced to either change the packet format used over serial or to allow the use of TCP/IP and user datagram protocol (UDP) networks. This led to a separation of the core protocol, which defines the protocol data unit (PDU), and the network layer, which defines the application data unit (ADU).

Modbus is a communication protocol for transmitting information between electronic devices over serial lines (original version) or via the Ethernet, and is commonly used in process and factory automation. While it's an open protocol and anybody can use it, "Modbus" is a registered

trademark of Schneider Electric USA, Inc. (current owner of the Modicon brand). The Modbus.org organization was created to further the use of Modbus and Schneider Electric has been a partner in it. This article is an introduction to Modbus and its basic functions—Modbus.org has extensive coverage on Modbus, the specifications for the various types of Modbus, software, testing, interface code and more. The Internet also has available tutorials and specific information on individual device Modbus implementations.

Modbus serial protocol (the original version) is a master/slave protocol, e.g. one master that controls the Modbus data transactions with multiple slaves that respond to the master's requests to read from or write data to the slaves. Modbus TCP, also known as Modbus TCP/IP, uses a client/server architecture. These network architectures are shown Figures 1 and 2.



Modbus serial architecture

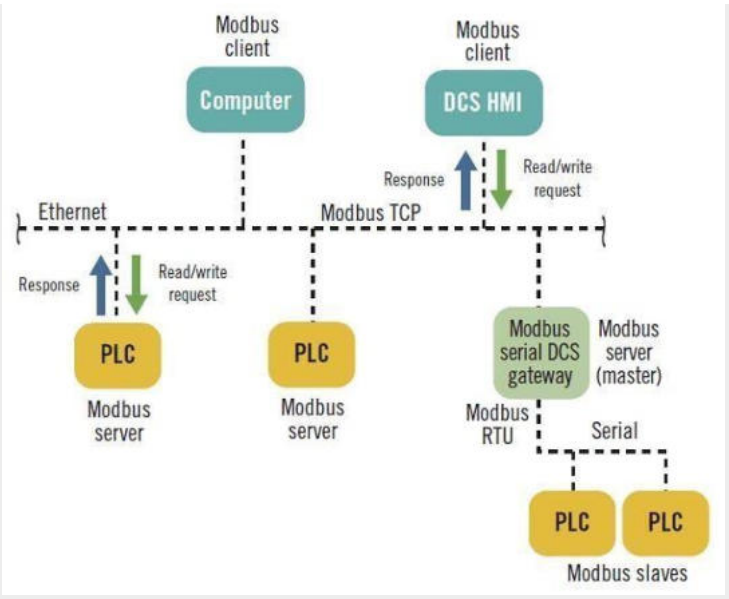
Figure 1: In a standard Modbus serial network, there is one master and up to 247 slaves, each with a unique slave address.

In a standard Modbus serial network, there is one master and as many as 247 slaves, each with a unique
slave address. Modbus TCP is typically implemented on an Ethernet network, and data transactions from aModbus client are directed toward a Modbus server via an IP address.

The original fieldbus

The Modbus protocol is the grandfather of modern fieldbuses. Modbus's popularity is due to its simplicity, its openness and ubiquitous nature—it's used everywhere. It has withstood the test of time and is still kicking after almost four decades. Modbus was originally published by Modicon in

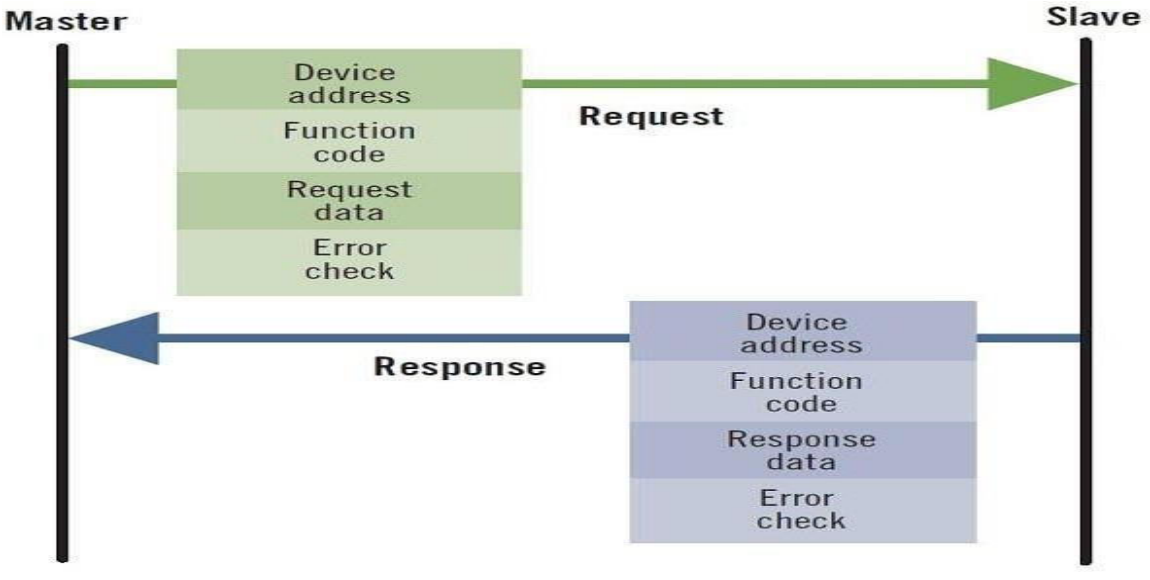
1979, primarily for use with its own PLCs. When industrial Ethernet appeared, Modbus TCP was developed, retaining much of Modbus' simplicity in a TCP/IP wrapper.



Modbus TCP architecture

Figure 2: Modbus TCP is typically implemented on an Ethernet network, and data transactions from a client are directed toward a server via an IP address.

Modbus is an application-layer protocol, independent of the data transmission medium. Data transactions are based on the master/client requesting data from or writing data to the slave/server. The data transactions are controlled by the master/client and there is no data-by-exception transmitted in standard Modbus. Data is based on 16-bit registers that can contain discrete on/off or 16-bit integer values. Some implementations use two or more integer registers to represent floating data or long integer values. Diagnostic data can be requested by a Modbus serial master from the slave, and the slave/server can send error codes to the master/client if they perceive there is something wrong with the request they received. Modbus data transactions only contain a function code, register addresses and data, and it is up to the master/client and the slave/server to make sense of the data.



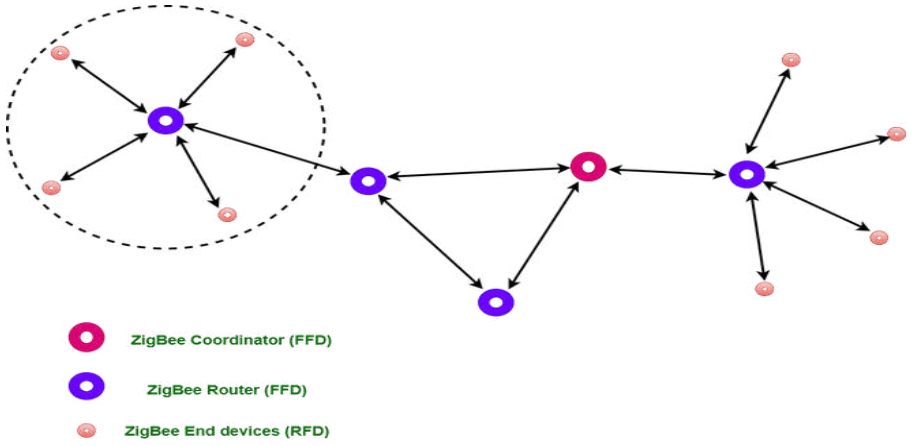
ZigBee

ZigBee is a Personal Area Network task group with low rate task group 4. It is a technology of home networking. ZigBee is a technological standard created for controlling and sensing the network. As we know that ZigBee is the Personal Area network of task group 4 so it is based on IEEE 802.15.4 and is created by Zigbee Alliance.

ZigBee is a standard that addresses the need for very low-cost implementation of Low power deviceswith Low data rates for short-range wireless communications.

Types of ZigBee Devices:

- **Zigbee Coordinator Device:** It communicates with routers. This device is used for connecting the devices.
- **Zigbee Router:** It is used for passing the data between devices.
- **Zigbee End Device:** It is the device that is going to be controlled.



General Characteristics of Zigbee Standard:

- Low Power Consumption
- Low Data Rate (20- 250 kbps)
- Short-Range (75-100 meters)
- Network Join Time (~ 30 msec)
- Support Small and Large Networks (up to 65000 devices (Theory); 240 devices (Practically))
- Low Cost of Products and Cheap Implementation (Open Source Protocol)
- Extremely low duty cycle.
- 3 frequency bands with 27 channels.

Operating Frequency Bands (Only one channel will be selected for use in a network):

1. **Channel 0**: 868 MHz (Europe)

2. **Channel 1-10**: 915 MHz (the US and Australia)
3. **Channel 11-26**: 2.4 GHz (Across the World)

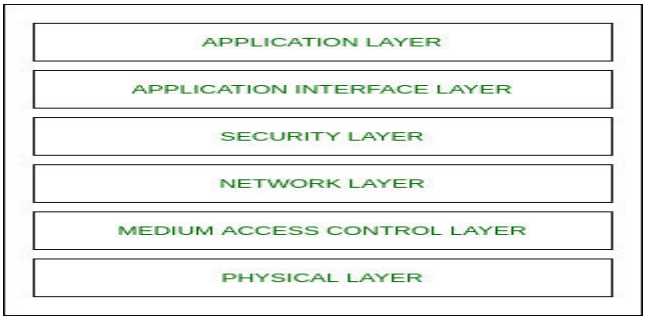
Zigbee Network Topologies:

- **Star Topology** (ZigBee Smart Energy): Consists of a coordinator and several end devices, end devices communicate only with the coordinator.
- **Mesh Topology** (Self Healing Process): Mesh topology consists of one coordinator, several routers,and end devices.
- **Tree Topology**: In this topology, the network consists of a central node which is a coordinator, several routers, and end devices. the function of the router is to extend the network coverage.

Architecture of Zigbee:

Zigbee architecture is a combination of 6 layers.

1. Application Layer
2. Application Interface Layer
3. Security Layer
4. Network Layer
5. Medium Access Control Layer
6. Physical Layer



- **Physical layer:** The lowest two layers i.e the physical and the MAC (Medium Access Control) Layer are defined by the IEEE 802.15.4 specifications. The Physical layer is closest to the hardware and directly controls and communicates with the Zigbee radio. The physical layer translates the data packets in the over-the-air bits for transmission and vice-versa during the reception.
- **Medium Access Control layer (MAC layer):** The layer is responsible for the interface between the physical and network layer. The MAC layer is also responsible for providing PAN ID and also network discovery through beacon requests.
- **Network layer:** This layer acts as an interface between the MAC layer and the application layer. It isresponsible for mesh networking.

- **Application layer:** The application layer in the Zigbee stack is the highest protocol layer and it consists of the application support sub-layer and Zigbee device object. It contains manufacturer-defined applications.

Channel Access:

1. **Contention Based Method** (Carrier-Sense Multiple Access With Collision Avoidance Mechanism)
2. **Contention Free Method** (Coordinator dedicates a specific time slot to each device (Guaranteed Time Slot (GTS)))

Zigbee Applications:

1. Home Automation
2. Medical Data Collection
3. Industrial Control Systems
4. meter reading system
5. light control system

Network Layer in Zigbee Architecture

Network Layer provides interface between MAC layer and the application layer. It is responsiblefor routing and establishing different Zigbee network topologies namely Star, Mesh and Tree topologies.

When a coordinator attempts to establish a Zigbee network, an energy scan is initiated to find the best RF channel for its new network. When a channel has been chosen, the coordinator assigns a PAN-ID which will be applied to all the devices that join the network.

PAN-ID is a 16 bit number that is used as a network identifier. A node is allowed to communicate on a network only when it undergoes the association process. The association function is used to join a node to a parent.

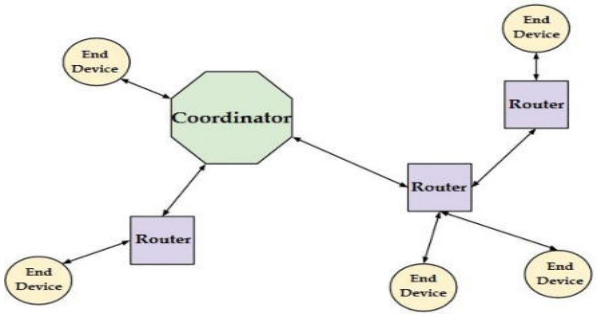


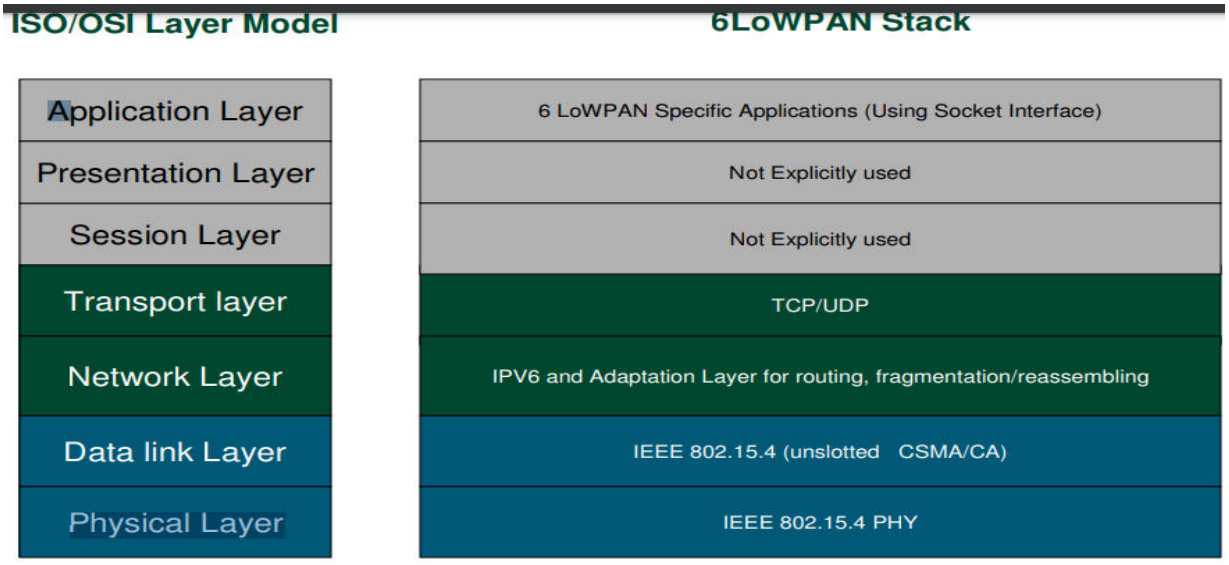
Fig. Types of Network Nodes in Zigbee Architecture (Zigbee Stack)

6LOWPAN:

6LoWPAN – Introduction:

- With more and more successful real environment WSN deployments on one hand, and the success and ubiquitous of IP Networks on the other, experts having been working towards an approach to bring the two together. 6LowPAN is an approach in this direction.
- 6LowPAN is an acronym for IPV6 over Low Power Personal Area Networks (such as the IEEE 802.15.4 radio).
- It is the working group name in the IETF (Internet Engineering Task Force).
- The IETF 6LoWPAN working group was formed in 2004 to address the challenge of enabling wireless IPv6 communication over the newly standardized IEEE 802.15.4 low-power radio for devices with limited space, power and memory, such as sensor nodes.
- 6LoWPAN provides a WSN node with IP communication capabilities. Simply speaking, it puts an adaptation layer above the 802.15.4 link layer and provides the ability of TCP/IP communication above the adaptation layer. The adaptation layer is necessary because stacking IP and above layers "as is" may not fit within one 802.15.4 frame.

6LoWPAN Stack Architecture:



6LoWPAN: Physical Layer

- The PHY layer provides the basic communication capabilities of the physical radio.
- It is based on IEEE 802.15.4 with a data rate of 250 Kbps and operating frequency of 2400 – 2483.5 MHz.
- The Physical layer PDU is IEEE 802.15.4 compliant with a maximum payload of 127 bytes.

6LoWPAN: Data Link Layer

- The Data Link layer provides services to enable reliable, single-hop communication links between devices.
- The MAC PDU is IEEE 802.15.4 compliant.
- IEEE 802.15.4 networks does not require to run in beaconenabled mode.
- In nonbeacon - enabled networks, data frames (including those carrying IPv6 packets) are sent via the

contention-based channel access method of unslotted CSMA/CA.

6LoWPAN: Adaptation Layer

- The adaptation layer is the main component of 6LoWPAN.
- The first major function of this layer is the TCP/IP header compression. TCP/IP headers are too large for 802.15.4, which has a maximum packet size of 128 bytes; instead IPv6 header size is 40 bytes, UDP and ICMP header sizes are both 4 bytes, TCP header size is 20. Without compression, 802.15.4 is not possible to transmit any payload effectively.
- A second major function of the adaptation layer is to handle packet fragmentation and reassembling. IEEE 802.15.4 has a maximum frame size of 128 bytes, while IPv6 requires a maximum transmission unit (MTU) of 1280 bytes. This mismatch has to be handled in the adaptation layer.
- The third major function of the adaptation layer is routing. The border nodes of the WSN should be able to route IPv6 packets into the WSN nodes from outside and route inside packets to outside IP network.

6LoWPAN: Network Layer

- It provides the Internetworking capability to sensor nodes.
- Addresses IPv6 node requirements.
- Appropriate security services.
- Routing considerations.
- Network management with SNMP (Simple Network Management Protocol).

6LoWPAN : Transport Layer

- The Transport Layer is responsible for delivering data to the appropriate application process on the host computers.
- Some Transport layer protocols are UDP and TCP.

6LoWPAN : Application Layer

6LoWPAN specific applications using socket interface

- Applications
- Equipment health monitoring
- Environment monitoring
- Security
- Home
- Building automation

Fragment Header:

The fragment header for the first fragment specifies the full (reassembled) packet size, and uses a datagram tag common for all fragments of this IP packet, which will be used by the receiver, together with the sender and destination MAC addresses, to identify fragments belonging to the same packets. Subsequent fragments also specify the offset of the fragment in the full IP packet, in multiples of 8 bytes (see Figure 12.2).

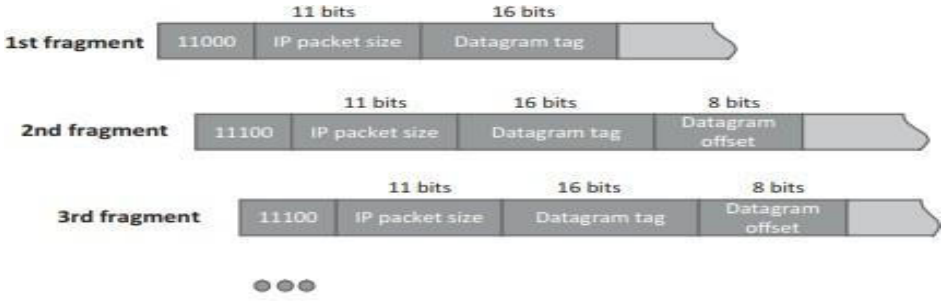


Figure 12.2 6LoWPAN fragment header.

COAP:

What Is CoAP?

Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things. CoAP is designed to enable simple, constrained devices to join the IoT even through constrained networks with low bandwidth and low availability. It is generally used for machine-to-machine (M2M) applications such as smart energy and building automation. The protocol was designed by the Internet Engineering Task Force (IETF), CoAP is specified in IETF RFC 7252.

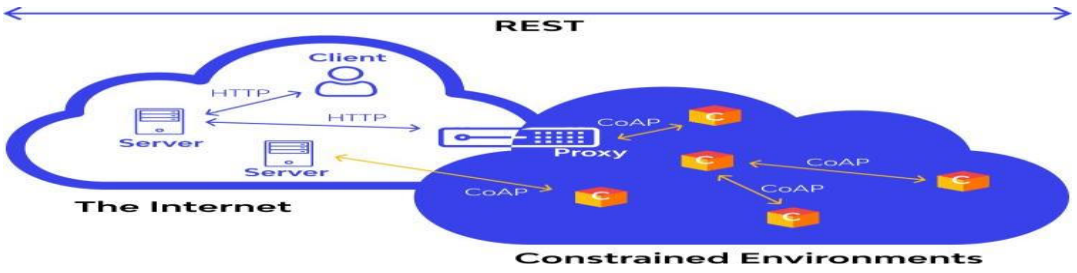
CoAP Architecture

The WWW and the constraints environment are two of the CoAP protocol's core aspects. The server monitors and assists communication through CoAP and HTTP, while proxy devices bridge the distance between these ecosystems, making communication more fluid.

CoAP allows HTTP clients (also known as CoAP clients) to speak or share data/information within resource limits.

Understanding the architecture requires familiarity with the following terms:

- ✓ Endpoints are the nodes that the host knows of;
- ✓ The client sends requests and replies to incoming requests;
- ✓ The server gets and forwards requests. It also gets and forwards the messages received in response to the requests it has processed.
- ✓ The sender creates and sends the original message.
- ✓ The recipient gets the information sent by the client or forwarded by the server.



CoAP Features

- Web Protocol Used in M2M With Constrained Requirements
- Asynchronous Message Exchange
- Low Overhead
- Very Simple To Perform Syntactic Analysis
- (URI) Uniform Resource Identifier
- Proxy and Caching Capabilities

How Does CoAP Function?

CoAP functions as a sort of HTTP for restricted devices, enabling equipment such as sensors or actuators to communicate on the IoT. These sensors and actuators are controlled and contribute by passing along their data as part of a system. The protocol is designed for reliability in low bandwidth and high congestion through its low power consumption and low network overhead. In a network with a lot of congestion or limited connectivity, CoAP can continue to work where TCP-based protocols such as MQTT fail to exchange information and communicate effectively.

Additionally, the effective and conventional CoAP features enable devices operating in poor signal quality to send their data reliably or enable an orbiting satellite to maintain its distant communication successfully. CoAP's also supports networks with billions of nodes. For security, the DTLS parameters chosen for default are an equivalent to 128 bit RSA keys.

COAP uses UDP as the underlying network protocol. COAP is basically a client-server IoT protocol where the client makes a request and the server sends back a response as it happens in HTTP. The methods used by COAP are the same used by HTTP.

CoAP Security

One must take security into account when dealing with IoT protocols. For example, CoAP uses UDP to transport information. CoAP relies on UDP security features to protect information. As HTTP uses TLS over TCP, CoAP uses Datagram TLS over UDP. DTLS supports RSA, AES, and so on.

The smallest CoAP message is 4 bytes in length, if omitting Token, Options and Payload. CoAP makes use of two message types, requests and responses, using a simple, binary, base header format. The base header may be followed by options in an optimized Type-Length-Value format. CoAP is by default bound to UDP and optionally to DTLS, providing a high level of communications security.

Any bytes after the headers in the packet are considered the message body. The length of the message body is implied by the datagram length. The entire message must fit within a single datagram when bound to UDP. When used with 6LoWPAN, as defined in RFC 4944, messages SHOULD also fit into a single IEEE 802.15.4 frame to minimize fragmentation.

V: BUILDING IoT WITH ARDUINO &RASPBERRY PI

Building IOT with Arduino- Building IOT with RASPERRY PI- IoT Systems - Logical Design using Python – IoT Physical Devices & Endpoints - IoT Device -Building blocks - Pi - Raspberry Pi Interfaces - **Case study:**Smart Home & Smart Industry.

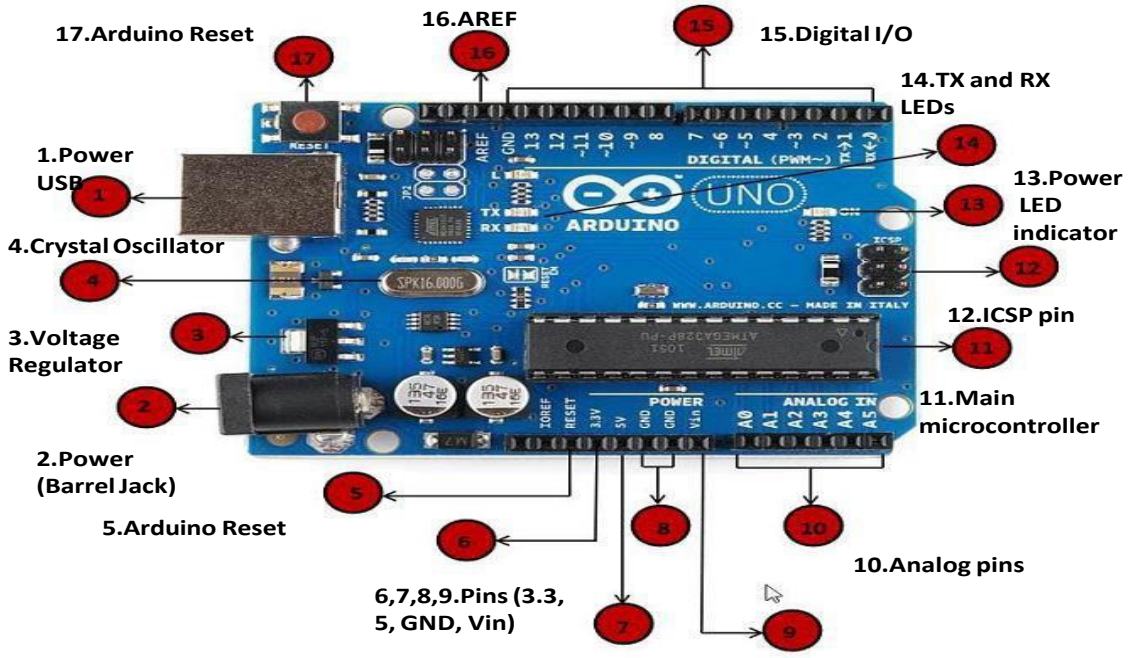
Building IOT with Arduino

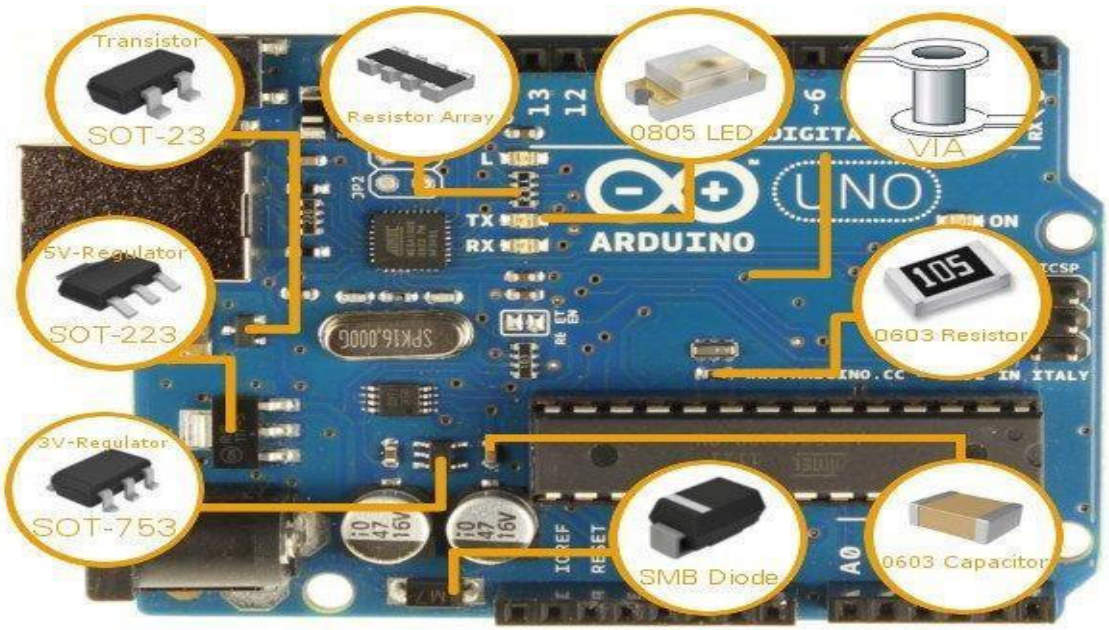
Internet of Things

The Internet of Things (IoT) is a scenario in which objects, animals or people are provided with single identifiers and the capability to automatically transfer and the capability to automatically transfer data more to a network without requiring human-to-human or human-to-computer communication.

Arduino Board:

- An Arduino is actually a microcontroller based kit.
- It is basically used in communications and in controlling or operating many devices.
- Arduino UNO board is the most popular board in the Arduino board family.
- In addition, it is the best board to get started with electronics and coding.
- Some boards look a bit different from the one given below, but most Arduino's have majority of these components in common.
- It consists of two memories- Program memory and the data memory.
- The code is stored in the flash program memory, whereas the data is stored in the data memory.
- Arduino Uno consists of 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button





1.Power USB

Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).

2.Power (Barrel Jack)

Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).

3.Voltage Regulator

The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

4.Crystal Oscillator

The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.

5,17.Arduino Reset

You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

6,7,8,9.Pins (3.3, 5, GND, Vin)

- 3.3V (6) − Supply 3.3 output volt
- 5V (7) − Supply 5 output volt
- Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.
- GND (8)(Ground) − There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- Vin (9) − This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.

10. Analog pins

The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.

11. Main microcontroller

Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

12. ICSP pin

Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.

13. Power LED indicator

This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

14. TX and RX LEDs

On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

15. Digital I/O

- The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled "~" can be used to generate PWM.

16. AREF

- AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Program an Arduino

- The most important advantage with Arduino is the programs can be directly loaded to the device without requiring any hardware programmer to burn the program.
- This is done because of the presence of the 0.5KB of Bootloader which allows the program to be burned into the circuit.
- All we have to do is to download the Arduino software and writing the code.
- The Arduino tool window consists of the toolbar with the buttons like verify, upload, new, open, save, serial monitor.
- It also consists of a text editor to write the code, a message area which displays the feedback like showing the errors, the text console which displays the output and a series of menus like the File, Edit, Tools menu.

Steps to program an Arduino

- Programs written in Arduino are known as sketches. A basic sketch consists of 3 parts

- Declaration of Variables
- Initialization: It is written in the setup() function.
- Control code: It is written in the loop () function.

- The sketch is saved with .ino extension. Any operations like verifying, opening a sketch, saving a sketch can be done using the buttons on the toolbar or using the tool menu.
- The sketch should be stored in the sketchbook directory.
- Chose the proper board from the tools menu and the serial port numbers.
- Click on the upload button or chose upload from the tools menu. Thus the code is uploaded by the bootloader onto the microcontroller.

Basic Adruino functions are:

- **digitalRead**(pin): Reads the digital value at the given pin.
- **digitalWrite**(pin, value): Writes the digital value to the given pin.
- **pinMode**(pin, mode): Sets the pin to input or output mode.
- **analogRead**(pin): Reads and returns the value.
- **analogWrite**(pin, value): Writes the value to that pin.
- **serial.begin**(baud rate): Sets the beginning of serial communication by setting the bit rate.

Design your own Arduino

- The following components are needed to design Arduino Board- A breadboard, a led, a power jack, a IC socket, a microcontroller, few resistors, 2 regulators, 2 capacitors.
    - The IC socket and the power jack are mounted on the board.
    - Add the 5v and 3.3v regulator circuits using the combinations of regulators and capacitors.
    - Add proper power connections to the microcontroller pins.
    - Connect the reset pin of the IC socket to a 10K resistor.
    - Connect the crystal oscillators to pins 9 and 10
    - Connect the led to the appropriate pin.
    - Mount the female headers onto the board and connect them to the respective pins on the chip.
    - Mount the row of 6 male headers, which can be used as an alternative to upload programs.
    - Upload the program on the Microcontroller of the readymade Adruino and then pry it off and place back on the user kit.

Advantages of Arduino Board

1. It is inexpensive
2. It comes with an open source hardware feature which enables users to develop their own kit using already available one as a reference source.
3. The Arduino software is compatible with all types of operating systems like Windows, Linux, and Macintosh etc.
4. It also comes with open source software feature which enables experienced software developers to use the Arduino code to merge with the existing programming language libraries and can be extended and modified.
5. It is easy to use for beginners.
6. We can develop an Arduino based project which can be completely stand alone or projects which involve direct communication with the software loaded in the computer.
7. It comes with an easy provision of connecting with the CPU of the computer using serial communication over USB as it contains built in power and reset circuitry.

Interfaces

UART Peripheral:

- A UART (Universal Asynchronous Receiver/Transmitter) is a serial interface.
- It has only one UART module.
- The pins (RX, TX) of the UART are connected to a USB-to-UART converter circuit and also connected to pin0 and pin1 in the digital header.

SPI Peripheral:

- The SPI (Serial Peripheral Interface) is another serial interface. It has only one SPI module.

TWI:

- The I2C or Two Wire Interface is an interface consisting of only two wires, serial data, and a serial clock: SDA, SCL.
- You can reach these pins from the last two pins in the digital header or pin4 and pin5 in the analog header.

Building IOT with RASPERRY PI

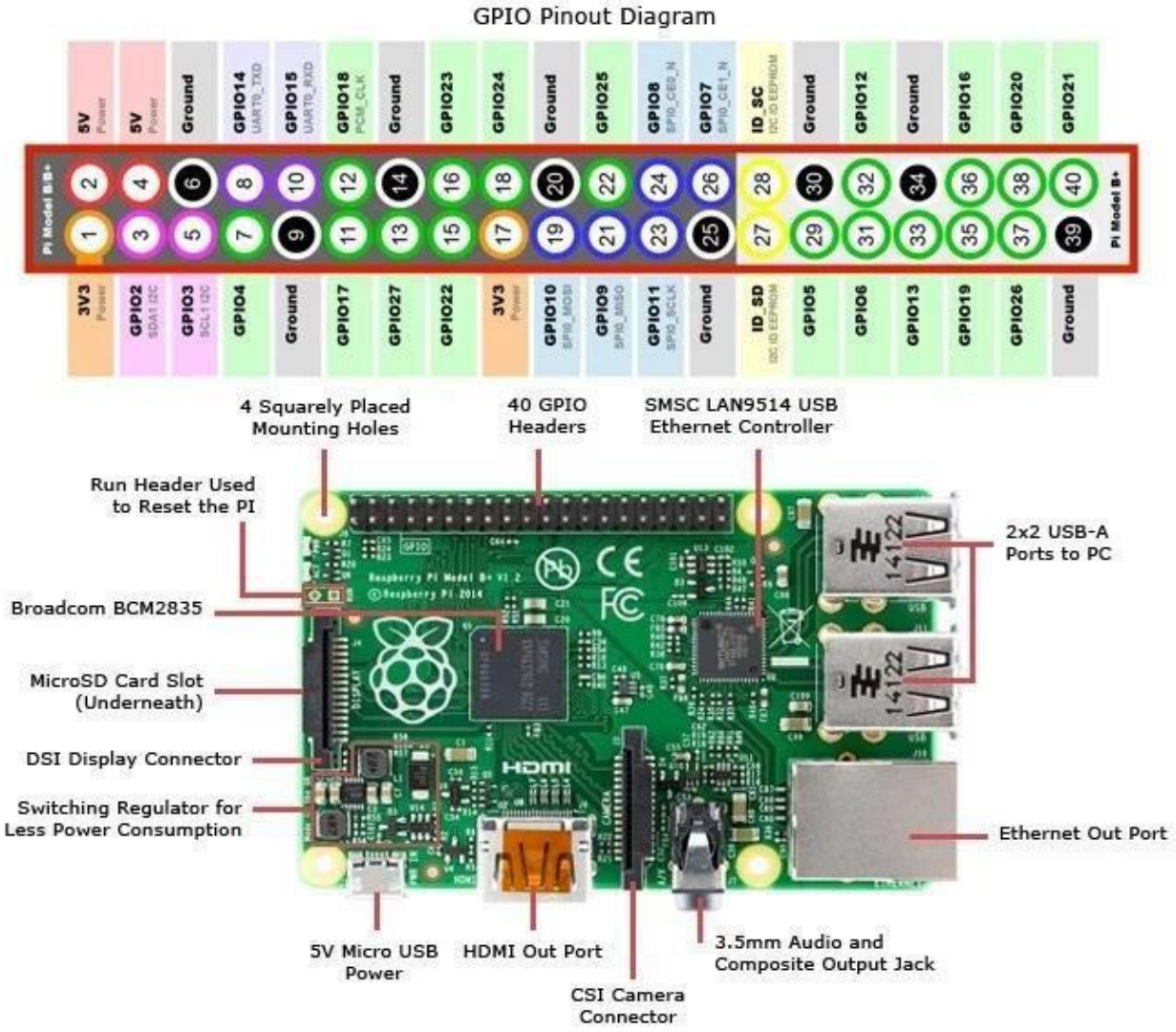
Internet of Things

The Internet of Things (IoT) is a scenario in which objects, animals or people are provided with single identifiers and the capability to automatically transfer and the capability to automatically transfer data more to a network without requiring human-to-human or human-to-computer communication.

Raspberry Pi

- The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins that allow you to control electronic components for physical computing and explore the Internet of Things (IoT).
- Raspberry Pi was basically introduced in 2006.
- It is particularly designed for educational use and intended for Python.
- A Raspberry Pi is of small size i.e., of a credit card sized single board computer, which is developed in the United Kingdom(U.K) by a foundation called Raspberry Pi.
- There have been three generations of Raspberry Pis: Pi 1, Pi 2, and Pi 3
- The first generation of Raspberry (Pi 1) was released in the year 2012, that has two types of models namely model A and model B.

- Raspberry Pi can be plugged into a TV, computer monitor, and it uses a standard keyboard and mouse.
- It is user friendly as can be handled by all the age groups.
- It does everything you would expect a desktop computer to do like word-processing, browsing the internet spreadsheets, playing games to playing high definition videos.
- All models feature on a broadcom system on a chip (SOC), which includes chip graphics processing unit GPU(a Video Core IV), an ARM compatible and CPU.
- The CPU speed ranges from 700 MHz to 1.2 GHz for the Pi 3 and on board memory range from 256 MB to 1 GB RAM.
- An operating system is stored in the secured digital SD cards and program memory in either the MicroSDHC or SDHC sizes.
- Most boards have one to four USB slots, composite video output, HDMI and a 3.5 mm phone jack for audio. Some models have WiFi and Bluetooth.
- Several generations of Raspberry Pis have been released.
- All models feature a Broadcom system on a chip (SoC) with an integrated ARM-compatible central processing unit (CPU) and on-chip graphics processing unit (GPU).
- Processor speed ranges from 700 MHz to 1.4 GHz for the Pi 3 Model B+ or 1.5 GHz for the Pi 4; on-board memory ranges from 256 MB to 1 GB with up to 4 GB available on the Pi 4 random-access memory (RAM).
- Secure Digital (SD) cards in MicroSDHC form factor (SDHC on early models) are used to store the operating system and program memory.
- The boards have one to five USB ports. For video output, HDMI and composite video are supported, with a standard 3.5 mm tip-ring-sleeve jack for audio output.
- Lower-level output is provided by a number of GPIO pins, which support common protocols like I²C. The B-models have an 8P8C Ethernet port and the Pi 3 and Pi Zero W have on-board Wi-Fi and Bluetooth.



Components and Peripherals

- **Voltages:** Two 5V pins and two 3V3 pins are present on the board, as well as a number of ground pins (0V). The remaining pins are all general purpose 3V3 pins
- A GPIO pin designated as an output pin can be set to high (3V3) or low (0V). A GPIO pin designated as an input pin can be read as high (3V3) or low (0V).
- **Processor & RAM:** Raspberry based on ARM11 processor. Latest version supports 700MHz processor and 512MB SDRAM. The Central processing unit is the brain of the raspberry pi board and that is responsible for carrying out the instructions of the computer through logical and mathematical operations.
- **Ethernet:** The Ethernet port of the raspberry pi is the main gateway for communicating with additional devices. The raspberry pi Ethernet port is used to plug your home router to access the internet.

- **USB Ports:** It has 2 USB ports. USB port provide current upto 100mA. For connecting devices that draw current more than 100mA, an external USB powered hub is required.
- **Ethernet Port:** It has standard RJ45 Ethernet port. Connect Ethernet cable or USB wifi adapter to provide internet connectivity.
- **HDMI Output:** It supports both audio and video output. Connect raspberry Pi to monitor using HDMI cable.
- **Composite video Output:** Raspberry comes with a composite video output with an RCA jack that supports both PAL and NTSC video output.
- **Audio Output:** It has 3.5mm audio output jack. This audio jack is used for providing audio output to old television along with RCA jack for video.
- **GPIO Pins:** It has a number of general purpose input/output pins. These pins are used to connect other electronic components. For example, you can connect it to the temperature sensor to transmit digital data.
- **Display Serial Interface (DSI):** DSI interface are used to connect an LCD panel to Raspberry PI.
- **Cameral Serial Interface(CSI):** CSI interface are used to connect a camera module to Raspberry PI.
- **SD Card slot:** Raspberry does not have built in OS and storage. Plug in an SD card loaded with Linux to SD card slot.
- **Power Input:** Raspberry has a micro USP connector for power input.
- **Memory:** The raspberry pi model A board is designed with 256MB of SDRAM and model B is designed with 51MB.Raspberry pi is a small size PC compare with other PCs. The normal PCs RAM memory is available in gigabytes. But in raspberry pi board, the RAM memory is available more than 256MB or 512MB
- **Status LEDs:** Raspberry has 5 status LEDs.

Status LED	Function
ACT	SD card Access
PWR	3.3V power is present
FDX	Full duplex LAN Connected
LNK	Link/Network Activity
100	100 Mbit LAN connected

Raspberry PI Interfaces:

- It supports SPI, serial and I2C interfaces for data transfer.
- **Serial :** Serial Interface on Raspberry has receive(Rx) and Transmit(Tx) pins for communication with serial peripherals.
- **SPI:** Serial Peripheral Interface (SPI) is a synchronous serial data protocol used for communicating with one or more peripheral devices. In an SPI connection, there is one master device and one or more peripheral devices. There are 5 pins Raspberry for SPI interface.

- **MISO(Master In Slave Out):** Master line for sending data to the peripherals.
- **MOSI(Master Out Slave In):** Slave Line for sending data to the master.
- **SCK(Serial Clock):** Clock generated by master to synchronize data transmission.
- **CE0(Chip Enable 0):** To enable or disable devices.
- **CE1(Chip Enable 1):** To enable or disable devices.

- **I2C:** I2C Interface pins are used to connect hardware modules. I2C interface allows synchronous data transfer with two pins: SDA(data line) and SCL (Clock Line)

Features of Raspberry PI

1. Where the system processing is huge. They can process high end programs for applications like Weather Station, Cloud server, gaming console etc. With **1.2GHz clock speed** and **1 GB RAM RASPBERRY PI** can perform all those advanced functions.

2. RASPBERRY PI 3 has wireless LAN and Bluetooth facility by which you can setup WIFI HOTSPOT for internet connectivity.

3. RASPBERRY PI had dedicated port for connecting touch LCD display which is a feature that completely omits the need of monitor.

4. RASPBERRY PI also has dedicated camera port so one can connect camera without any hassle to the PI board.

5. RASPBERRY PI also has PWM outputs for application use.

6. It supports HD steaming

Applications

- ✓ Hobby projects.
- ✓ Low cost PC/tablet/laptop
- ✓ IoT applications
- ✓ Media center
- ✓ Robotics
- ✓ Industrial/Home automation
- ✓ Server/cloud server
- ✓ Print server
- ✓ Security monitoring
- ✓ Web camera
- ✓ Gaming
- ✓ Wireless access point

IoT Systems - Logical Design using Python

Introduction

Characteristics of Python:

- **Multi-paradigm programming language**: Python supports more than one programming paradigms including object oriented programming and structured programming.
- **Interpreted Language:** It is an interpreted language and does not require an explicit compilation step. Python interpreter executes the program source code directly and statement by statement interpreted.
- **Interactive Language:** user can submit commands at the python prompt and interact with the interpreter directly.
- **Easy to Learn and Use:** Python is easy to learn and use. It is developer-friendly and high level programming language.
- **Object and procedure oriented:** python supports both procedure and object oriented programming. It allows programs to be written around procedures or functions that allow reuse of code.
- **Extendable:** It is an extendable language and allows integration of low level modules written in languages such as C/C++.
- **Scalable:** It provides a manageable structure for large programs.
- **Portable:** Python programs are executed directly from source code and copy from one machine to other without worry about portability. Python interpreter converts source code to an intermediate form called byte code and then translate this into the native language of your specific system and then runs it.
- **Broad Library support:** It supports and works on different platforms such as windows, Linux, Mac etc. Large number of packages are available for various applications such as machine learning, image processing, network programming, cryptography etc.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Uses / Applications of Python

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Installing Python

It is highly portable language that works on different platforms such as windows, Linux, Mac etc.

Windows

Python 2.7 can be downloaded from http://www.python.org/ftp/python/2.7.5/python-2.7.5.msi

Linux

Install python in Ubuntu Linux like:

#download python

wget http://python.org/ftp/python/2.7.5/python-2.7.5.tgz

Python Packages for IoT

JSON

- Javascript Object Notation(JSON) is an easy to read and write data-interchange format.
- It is an alternate to XML.
- It is built on 2 structure: collection of name-value pairs(dictionary) and ordered list of values(list)
- JSON format is used for serializing and transmitting structured data over a network connection.
- If you have a JSON string, you can parse it by using the json.loads() method.
- If you have a Python object, you can convert it into a JSON string by using the json.dumps() method
- Example:

```
import json

# some JSON:
x =  '{ "name":"John", "age":30, "city":"New York"}'

# parse x:
y = json.loads(x)

# the result is a Python dictionary:
print(y["age"])

# a Python object (dict)
x = {
  "name": "John",
  "age": 30,
  "city": "New York"
}

# convert into JSON:
y = json.dumps(x)

# the result is a JSON string:
print(y)
```

- Output:

```
30
{"name": "John", "age": 30, "city": "New York"}
```

XML

- Extensible Markup Language(XML) is a data format for structured document interchange.
- It was designed to store and transport small to medium amounts of data and is widely used for sharing structured information.
- In order to parse an XML document using minidom, we must first import it from the xml.dom module. This module uses the parse function to create a DOM object from our XML file.
- getElementByTagName() to find a specific tag.
- Example:

```
from xml.dom import minidom

# parse an xml file by name
mydoc = minidom.parse('items.xml')

items = mydoc.getElementsByTagName('item')


# all item attributes
print('\nAll attributes:')
for elem in items:
   print(elem.attributes['name'].value)

# one specific item's data
print('\nItem #2 data:')
print(items[1].firstChild.data)


# all items data
print('\nAll item data:')
for elem in items:
   print(elem.firstChild.data)
```

- Output:

```
All attributes:
item1
item2

Item #2 data:
Mango

All item data:
Apple
Mango
```

HTTPlib and URLlib

- HTTPLib2 and URLLib2 are python libraries used in network/internet programming.
- HTTPLib2 is an HTTP client library and URLLib2 is a library for fetching URLs.
- import urllib.requests. From there, we assign the opening of the url to a variable, where we can finally use a .read() command to read the data.
- Example1:

```
import http.client
conn = http.client.HTTPSConnection("www.python.org")
conn.request("GET", "/")
r1 = conn.getresponse()
print(r1.status, r1.reason)
data1 = r1.read()  # This will return entire content.
# The following example demonstrates reading data in chunks.
conn.request("GET", "/")
r1 = conn.getresponse()
while not r1.closed:
    print(r1.read(200))
```

- Example2:

```
import urllib.request
x = urllib.request.urlopen('https://www.gmail.com/')
print(x.read())
```

SMTPLib

- Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending an e-mail and routing e-mail between mail servers.
- Python provides **smtplib** module, which defines an SMTP client session object that can be used to send mails to any Internet machine with an SMTP or ESMTP.
- Syntax:

```
import smtplib
smtpObj = smtplib.SMTP( [host [, port [, local_hostname]]] )
```

- Here is the detail of the parameters −
- **host** − This is the host running your SMTP server. You can specifiy IP address of the host or a domain name like tutorialspoint.com. This is an optional argument.
- **port** − If you are providing *host* argument, then you need to specify a port, where SMTP server is listening. Usually this port would be 25.
- **local_hostname** − If your SMTP server is running on your local machine, then you can specify just *localhost* the option.
- An SMTP object has an instance method called **sendmail**, which is typically used to do the work of mailing a message. It takes three parameters −
- The *sender* − A string with the address of the sender.
- The *receivers* − A list of strings, one for each recipient.
- The *message* − A message as a string formatted as specified in the various RFCs.

- Example:

```
import smtplib

sender = 'from@fromdomain.com'
receivers = ['to@todomain.com']

message = """From: From Person <from@fromdomain.com>
To: To Person <to@todomain.com>
Subject: SMTP e-mail test

This is a test e-mail message.
"""

try:
   smtpObj = smtplib.SMTP('localhost')
   smtpObj.sendmail(sender, receivers, message)
   print "Successfully sent email"
except SMTPException:
   print "Error: unable to send email"
```

IoT physical Devices & Endpoints

IoT Device

- Thing in Internet of Things(IoT) can be any object that has a unique identifier and which can send/receive data over a network.
- IoT devices are connected to the internet and send information about themselves or about their surroundings over a network or allow actuation upon the physical entities or environment around them remotely.
- Some example of IoT devices are:
  - A Home Automation device that allows remotely control and monitor the appliances.
  - An Industrial Machine which sends information about its operation and health monitoring data to a server.
  - A car sends information about its location to a cloud based service.
  - A wireless enabled wearable device that measures data about a person such as number of steps walked and send the data to a cloud based service.

Basic Building block of an IoT device

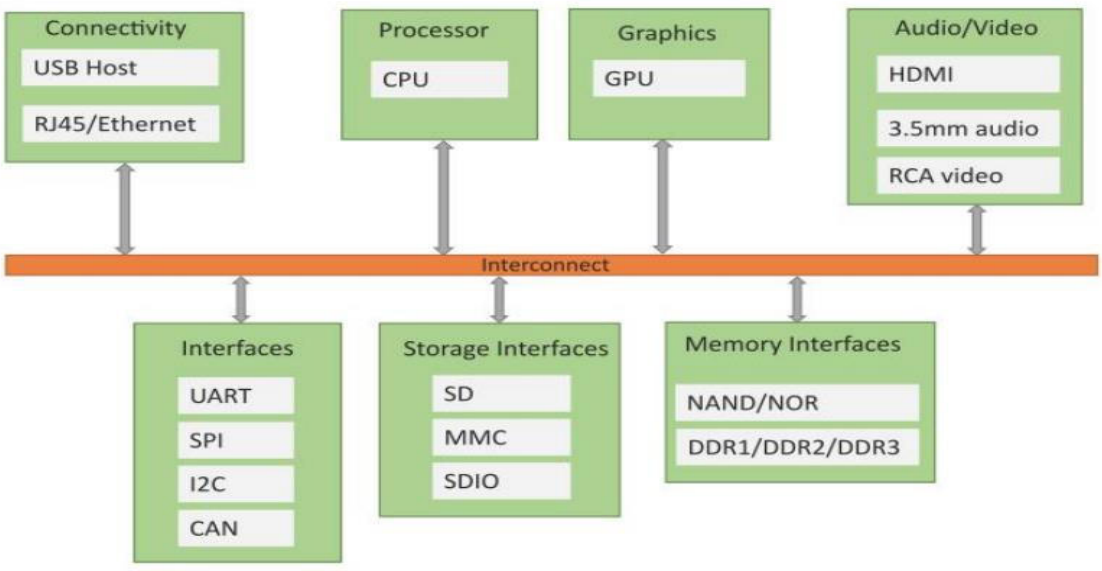
- IoT stands for "Internet of Things," which means using the Internet to connect different things. IoT is an intersection between the physical and virtual worlds. It essentially maps virtual operations onto real interactions.

IoT device consists of a number of modules based on functional attributes:

- **Sensors:** It can be either on board the IoT device or attached to the device. IoT device can collect various types of information from the onboard or attached sensors such as temperature, humidity, light intensity etc. The sensed information can be communicated

either to other devices or cloud based server/storage. These form the front end of the IoT devices. These are the so called "Things" of the system. Their main purpose is to collect data from its surrounding (sensors) or give out data to its surrounding (actuators). These have to be uniquely identifiable devices with a unique IP address so that they can be easily identifiable over a large network. It should be able to collect real time data. Examples of sensors are: gas sensor, water quality sensor, moisture sensor etc.

- **Actuation:** IoT devices can have various types of actuators attached that allow taking actions upon the physical entities in the vicinity of the device. Example relay switch connected to an IoT device can turn appliances on/off based on the commands send to the device.
- **Processors:** Processors are the brain of the IoT system. Their main function is to process the data captured by the sensors and process them so as to extract the valuable data from the enormous amount of raw data collected. It gives intelligence to the data. Processors mostly work on real-time basis and can be easily controlled by applications. These are also responsible for securing the data – that is performing encryption and decryption of data.
- **Communication:** It is responsible for sending collected data to other devices or cloud based servers/storage and receiving data from other devices and commands from remote applications. Gateways are responsible for routing the processed data and send it to proper locations for its (data) proper utilization. Gateway helps in to and fro communication of the data. It provides network connectivity to the data. Network connectivity is essential for any IoT system to communicate. LAN, WAN, PAN etc are examples of network gateways.
- **Analysis & Processing:** These are responsible for taking decision based upon the collected data.
- Given diagram shows the Single Board Computer(SBC) based IoT device that includes CPU, GPU,RAM, storage and various types of interfaces and perpherals.



Programming with Raspberry pi

Controlling LED with Raspberry Pi Python program for blinking LED

```
Import Rpi.GPIO as GPIO
Import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(18,GPIO.OUT)
While True:
        GPIO.output(18,True)
        time.sleep(1)
        GPIO.output(18,False)
        time.sleep(1)
```

Program uses the Rpi.GPIO module to control the GPIO on Raspberry Pi. In this program we set pin 18 direction to output and then write True/False alternatively after a delay of one second.

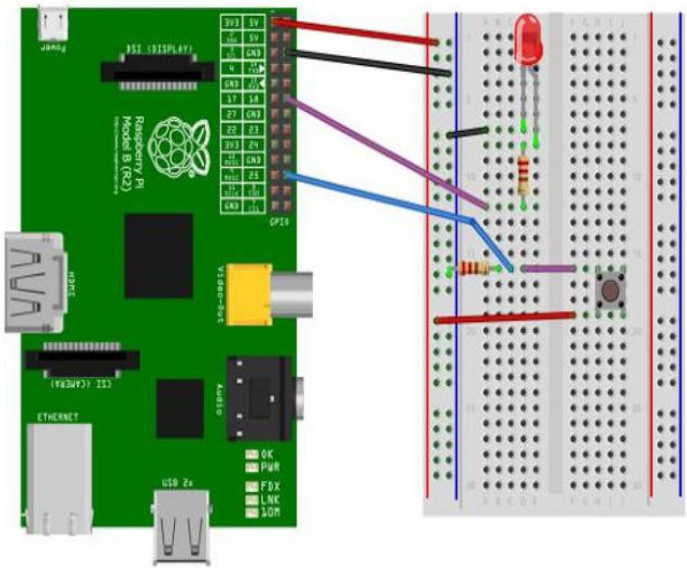
Interfacing an LED and switch with Raspberry Pi:

```
from time import sleep
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

#Switch Pin
GPIO.setup(25, GPIO.IN)
#LED Pin
GPIO.setup(18, GPIO.OUT)
state=false

def toggleLED(pin):
      state = not state
      GPIO.output(pin, state)

while True:
      try:
            if (GPIO.input(25) == True):
                  toggleLED(pin)
            sleep(.01)
            except KeyboardInterrupt:
                  exit()
```



Unit 5 Case Studies and Real-World Applications

Real world design constraints - Applications - Asset management, Industrial automation, Smart grid, Commercial building automation, Smart cities - Participatory sensing - Data Analytics for IoT – Software & Management Tools for IoT Cloud Storage Models & Communication APIs - Cloud for IoT - Amazon Web Services for IoT.

REAL-WORLD DESIGN CONSTRAINTS

The IoT allows for the development of novel applications in all imaginable scenarios. The technical design of any M2M or IoT solution requires a fundamental understanding of the specificity of the intended application and business proposition, in addition to heterogeneity of existing solutions. They are,

1. **Devices and Networks**:
   - The devices that form networks in the M2M Area Network domain must be selected, or designed, with certain functionality suitable to IoT applications.
   - The devices must have an energy source (e.g., batteries), computational capability (e.g., Micro Controller Unit (MCU)), appropriate communications interface (e.g., a Radio Frequency Integrated Circuit (RFIC) and front-end RF circuitry), memory (program and data), and sensing (and/or actuation) capability.
   - These must be integrated in such a way that the functional requirements of the desired application can be satisfied with additional non-functional requirements.

2. **Functional Requirements:**
   - Specific sensing and actuating capabilities.
   - Sensing principle and data requirements: Sometimes continuous sampling of sensing data is required. For some applications, sampling after specific intervals is required.
   - The parameters like higher network throughput, data loss, energy use, etc are decided based on sensing principle.

3. **Sensing and communications field:**
   - The sensing field is to be considered for sensing in local area or distributed sensing. The distance between sensing points is also important factor to be considered.
   - The physical environment has an implication on the communicationstechnologies selected and the reliability of the system in operation thereafter.
   - Devices must be placed in close enough proximity to communicate. Where the distance is too great, routing devices may be necessary.

4. **Programming and embedded intelligence:**
   - Devices in the IoT are heterogeneous such as various computational architectures, including MCUs (8, 16, 32-bit, ARM, 8051, RISC, Intel, etc.), signal conditioning (e.g., ADC), and memory (ROM, S/F/D) RAM, etc.), communications media, peripheral components (sensors, actuators, buttons, screens, LEDs), etc.
   - In every case, an application programmer must consider the hardware selected or designed, and its capabilities.
   - Application-level logic decides the sampling rate of the sensor, the local processing performed on sensor readings, the transmission schedule (or reporting rate), and the management of the communications protocol stack, among other things.
   - The programmers have to reconfigure and reprogram devices in caseof change in devices in IoT application.

5. **Power:**
   - Power is essential for any embedded or IoT device.
   - Depending on the application, power may be provided by the mains, batteries, or hybrid power sources.
   - Power requirements of the application are modeled prior to deployment. This allows the designer to estimate the cost of maintenance over time.

6. **Gateway:**

Gateway devices or proxies are selected according to need of datatransitions.

7. **Non-functional requirements:**

The non-functional requirements are technical and non-technical.

- Regulations:
  - For applications that require placing nodes in public places, priorpermissions are important.
  - Radio Frequency (RF) regulations limit the power with which transmitters can broadcast.
- Ease of use, installation, maintenance, accessibility:

This relates to positioning, placement, site surveying, programming, and physical accessibility of devices for maintenance purposes.

- Physical constraints:
  - Integration of additional electronics into existing system
  - Suitable packaging
  - Kind and size of antenna
  - Type of power supply

8. **Financial cost:**

Financial cost considerations are as follows:

- Component Selection: Typically, the use of these devices in the M2M Area Network domain is to reduce the overall cost burden. However, there are research and development costs likely to be incurred for each individual application in the IoT that requires device development or integration. Developing devices in small quantities is expensive.
- Integrated Device Design: Once the energy, sensors, actuators, computation, memory, power, connectivity, physical, and other functional and non-functional requirements are considered, it is likely that an integrated device must be produced.

9. **Data representation and visualization:**

Each IoT application has an optimal visual representation of the data and the system. Data that is generated from heterogeneous systems has heterogeneous visualization requirements. There are currently no satisfactory standard data representation and storage methods that satisfy all of the potential IoT applications.

IoT APPLICATIONS:

IoT applications promise to bring immense value into our lives. With newer wireless networks, superior sensors and revolutionary computing capabilities, the Internet of Things could be the next frontier in the race for its share of the wallet. IoT applications are expected to equip billions of everyday objects with connectivity and intelligence. It is already being deployed extensively, few applications of IoT:

- Wearables
- Smart Home Applications
- Smart Buildings
- Smart Infrastructure
- Securities
- Health Care
- Smart Cities
- Agriculture
- Industrial Automation

ASSETMANAGEMENT:

Asset management (or asset tracking) is the process of keeping track of the company's physical assets and their information. Depending on the business, physical assets can mean different kinds of equipment, IT devices, tools or vehicles, for example. The emergence of the IoT with its billions of envisioned devices poses a clear challenge to the management of these. Existing asset management practices consider the operations applicable to various physical assets, and in their majority refer to monitoring of their operations and to some degree to the adjustment (control) of their behavior. However, such operations have been up to now strongly coupled with what the devices and underlying systems are capable of, bound with (mostly) proprietary protocols in order to cover the largest possible spectrum of functionalities and guarantee results, and are mostly static.

A typical manageability nightmare scenario includes the configuration of assets in modern enterprises. For example, today employees use several computing devices (e.g. laptops, desktops, smart phones, tablets, access cards, security tokens, etc.); all of these need to be accounted for in backend systems, to be integrated with enterprise-wide monitoring solutions, and to comply with the organization’s policies and requirements. How to achieve that is challenging;for example, how does one protect a company’s assets and the data they contain from unauthorized access or usage?

Expected benefits:

The management in the IoT era, although challenging, may yield significant benefits and enable the mastering of the vast device-based infrastructure. Several benefits are expected with M2M in asset management. These may include:

- Reduced costs, e.g. because of remote telemetry without the need of field personnel to be engaged.
- Increased quality, e.g. because of the fine-grained monitoring data that could be done even in near real-time.
- Increased resilience, e.g. because of analysis of device’s status, can lead to predictive maintenance, which minimizes apart from costs, and also downtime and unexpected failures.
- Increased performance and security: remote updates may enhance the operational capabilities of the assets.
- Increased security: updates of the asset’s software can help correct its behavior and security holes.
- Asset location tracking, e.g. easier recovery and theft prevention of assets.
- Operation optimization, e.g. fleet management optimizing for journeys onthe fly.
- New services, e.g. energy awareness via smart metering, location-based services, e-ticketing, etc.

E-Maintenance in the M2M Era:

The recent focus is on e-maintenance, i.e. “maintenance support which includes the resources, services, and management necessary to enable proactive decision process execution”, especially due to the expected benefits of M2M thatcan be harnessed.

Key strategies of e-maintenance are the following:

- Remote Maintenance: The capability empowered by Information and Communication Technologies (ICT) to provide maintenance practices from anywhere without being physically present (e.g. third party entities outside the enterprise borders). This approach enables far more effective reaction to maintenance, and dramatically affects business models having maintenance services as part of their functionalities.
- Predictive Maintenance: Here the adoption of models and methodologies is implied that analyzes the operational performance of assets and attempts to predict malfunctions and failures, or determine on-demand when maintenance checks are needed (and not as usually done at fixed intervals). Here, apart from the immediate benefit of asset reliability, one can also witness optimization of maintenance schedules carried out by field personnel,enhanced planning for replacement of assets, increased customer satisfaction,etc.
- Real-Time Maintenance: Current failures on the shop floor, although evident, might require significant time until they are assessed, repaired, and re- planned with respect to the operations the enterprise systems had scheduled. With real-time notifications up to enterprise systems, immediate assessment on the operational factory or enterprise-wide processes can be achieved and addressed.
- Collaborative Maintenance: This capability enables traditional maintenance concepts to integrate collaboration among different areas of the enterprise that may lead to leaner processes, simpler landscapes, and more effective management.

Cross-company communication is already a reality, but constrained at enterprise-level operations without in-depth information that could triggersophisticated re-planning scenarios. However, with M2M real-time connection to the devices, malfunctions can be quickly analyzed and resolved with the help of multiple experts such as those with knowledge of the specific process, those responsible for the hardware/software (HW/SW)of the device, and potentially the field personnel on-site (as depicted in Figure below). All these can now collaborate using multiple Internet-based technologies with seamless data, voice,and video integration over a future e-maintenance platform.

Communication done directly (e.g. via common trusted third-party service providers) may simply couple the two companies for the specific business case, and remove the overhead of costly home-grown solutions by propagating all the necessary information to the affected stakeholders. Synergies canbe identified, and information that was too costly to be obtained in a timely manner can now flow into cross-company applications and services. This approach is very well suited for dynamic and short-lived interactions that can be set up, exploited, and removed as easy as a simple composite service.

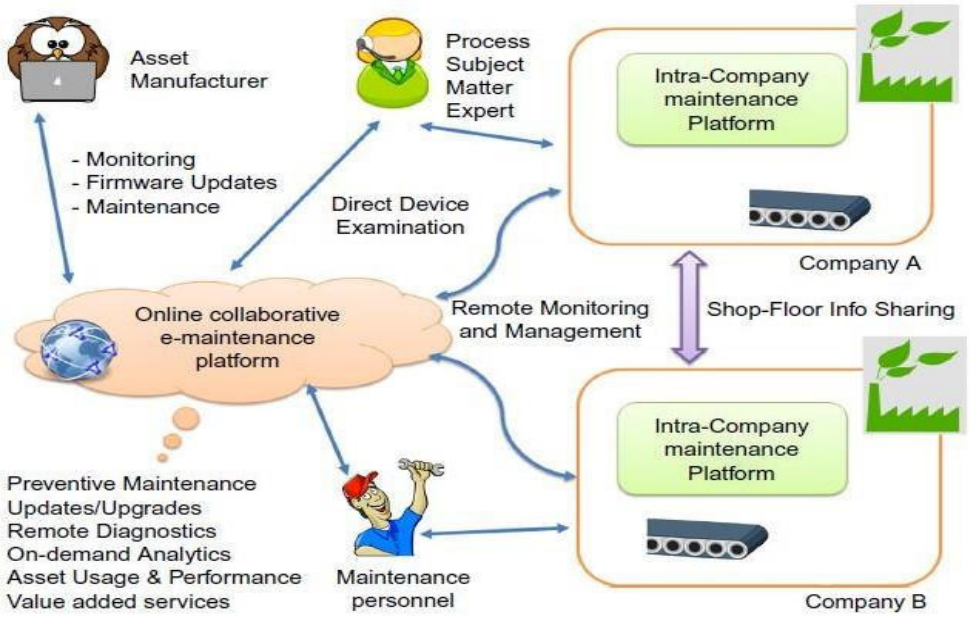


Figure: Outsourced remote continuous cross-company e-maintenance.

Hazardous goods management in the M2M Era:

Traditionally, the monitoring was done with passive radio frequency identification (RFID) tags, however, Collaborative Business Items(CoBIs) deployed Wireless Sensor Networks (WSNs) that could execute business logic locally and communicate among themselves as well as with the enterprise systems. WSNs are seen as one of the most promising technologies that will bridge the physical and virtual worlds, enabling them to measure, assess, and actuate real-world environments. A typical demonstration of how M2M helps with the hazardous goods management scenario is shown in Figure below. All drums are equipped with WSNs that have internally stored information about the chemical within the drum, an "incompatible goods" list indicating which chemicals are dangerous to be in near proximity with it as corrosion might lead to explosions, etc., as well as the max limit according to current regulations of this chemical that can be stored within a limited space.

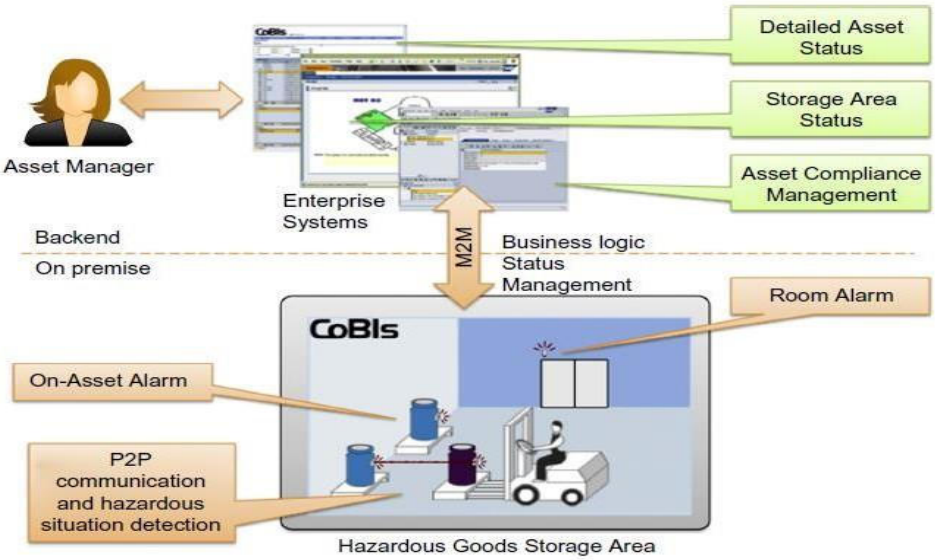


Figure: M2M-enhanced Hazardous Goods Management in CoBIs.

INDUSTRIAL AUTOMATION:

Industrial automation is the use of control systems, such as computers or robots,and information technologies for handling different processes and machineriesin an industry to replace a human being.

Service-oriented architecture-based device integration:

The emerging approach in industrial environments is to create system intelligence by a large population of intelligent, small, networked, embedded devices at a high level of granularity, as opposed to the traditional approach of focusing intelligence on a few large and monolithic applications. This increased granularity of intelligence distributed among loosely coupled intelligent physical objects facilitates the adaptability and re-configurability of the system, allowing it to meet business demands not foreseen at the time of design, and providing real business benefits.

The Service-Oriented Architecture (SOA) paradigm can act as a unifying technology that spans several layers, from sensors and actuators used for monitoring and control at shop-floor level, up to enterprise systems and their processes as envisioned in Figure 1(below). This common "backbone" means that M2M is not limited to direct (e.g. proximity) device interaction, but includes a wide range of interactions in a cross-layer way with a variety of heterogeneous devices, as well as systems and their services. This yields multiple benefits for all stakeholders involved.

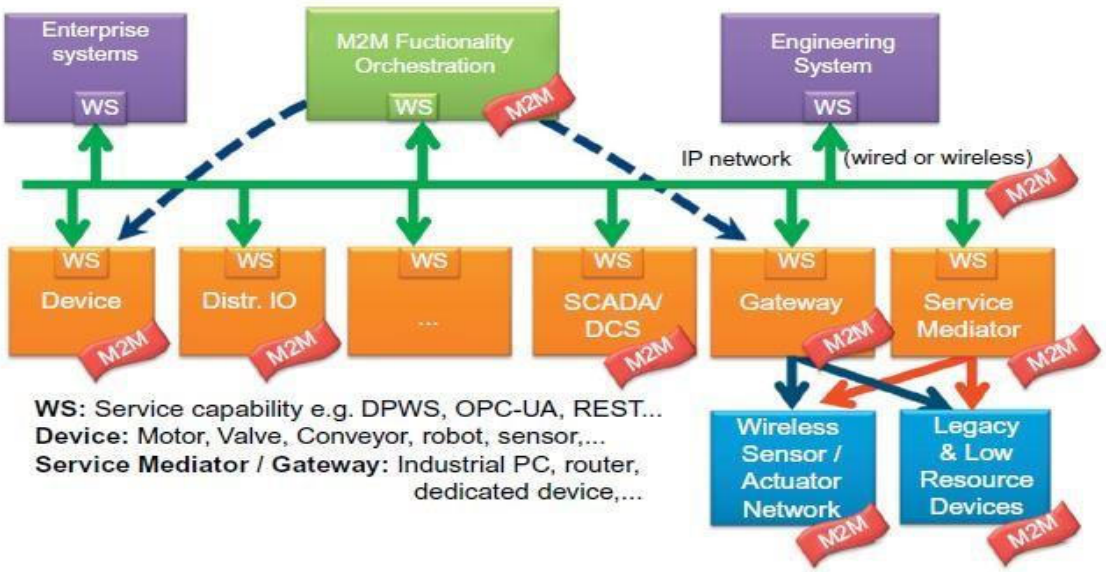


Figure 1: M2M SOA-based integration.

The SOA-based vision is not expected to be realized overnight, but may take a considerable time depending on the lifecycle processes of the specific industry, and may be impacted by micro- and macro-economic aspects. Hence, it is important that migration capabilities are provided so that we can harvest someof the benefits today and provide a step-wise process towards achieving the vision. The concepts of gateway and service mediator as depicted in Figure 2(below) can help towards this direction.

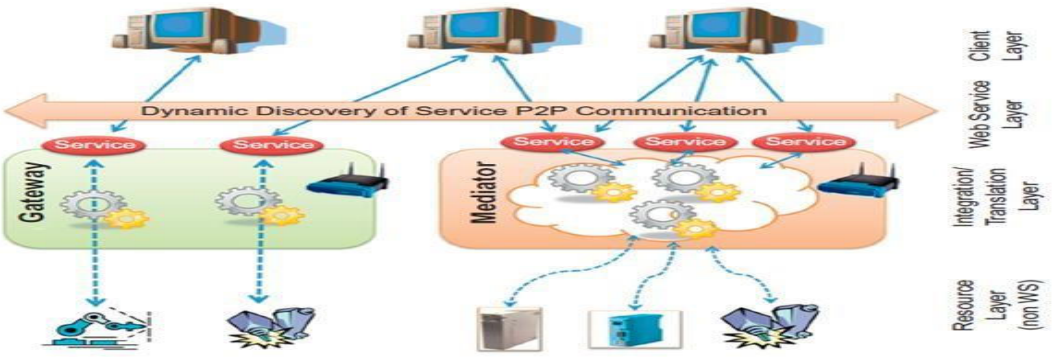
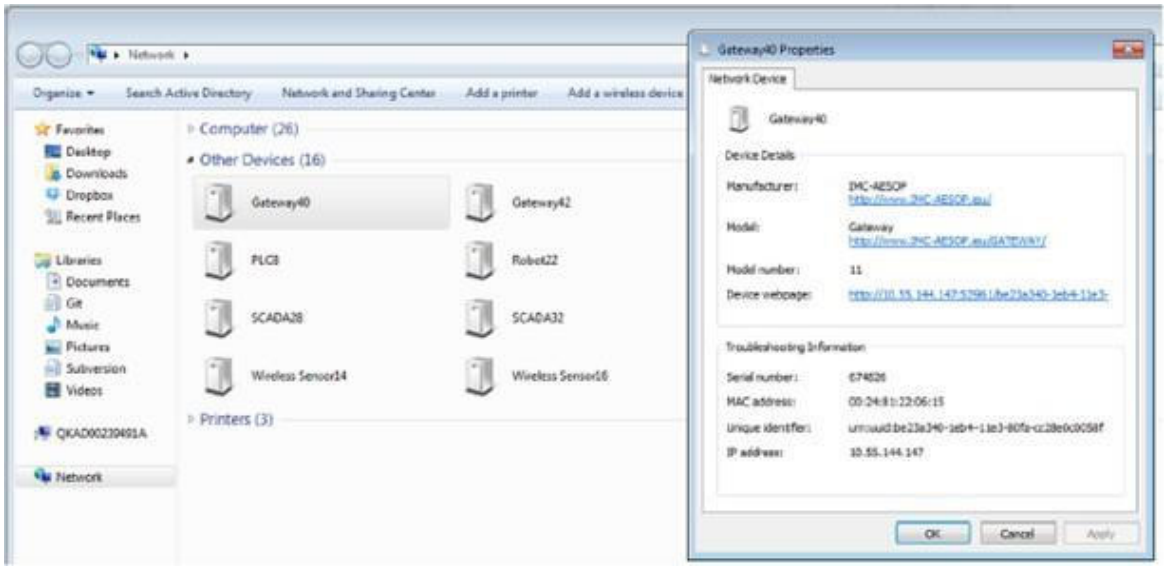


Figure 2: Non-service-enabled device integration: Gateway vs. Service Mediator

Dynamic device discovery is a key functionality in the future M2M. As an example, Figure 3(below) depicts how Windows 7 can discover dynamically heterogeneous devices that are SOA-ready(i.e. equipped with web services; Devices Profile for Web Services, DPWS).

Figure 3: Dynamic device discovery via DPWS in Windows 7.

A Gateway is a device that controls a set of lower-level non-service-enabled devices, each of which is exposed



by the Gateway as a service-enabled device.

This approach allows gradually replacing limited-resource devices or legacy devices by natively WS-enabled devices without impacting the applications using these devices.

SOCRADES: realizing the enterprise integrated Web of Things:

Agility and flexibility are required from modern factories. This, in conjunction with the rapid advances in Information Technology (IT), both in hardware and software, as well as the increasing level of dependency on cross- factory functionalities, sets new challenging goals for future factories. The latter are expected to rely on a large ecosystem of systems where collaboration at largescale will take place. Mashing up services has proven to be a key advantage in the Internet application area; and if now the devices can either host web services natively or be represented as such in higher systems, then existing tools and approaches can be used to create mash-up apps that depend on these devices. A visionary project that followed this line of thinking was the Industry-driven European Commission funded project SOCRADES.

Driven by the key need for cross-layer M2M collaboration (i.e. at shop-floor level among various heterogeneous devices as well as among systems and services up to the Enterprise (ERP) level), an architecture had been proposed, prototyped, andassessed. SOCRADES proposed and realized SOA-based integration, as shown in Figure2 (above), including migration of existing infrastructure via gateways and service mediators, as shown in Figure3 (above).

The SOCRADES Integration Architecture (SIA), as analyzed by Karnous kos etal., enables enterprise-level applications to interact with and consume data froma wide range of networked devices using a high-level, abstract interface that features WS standards (Figure.4 below).

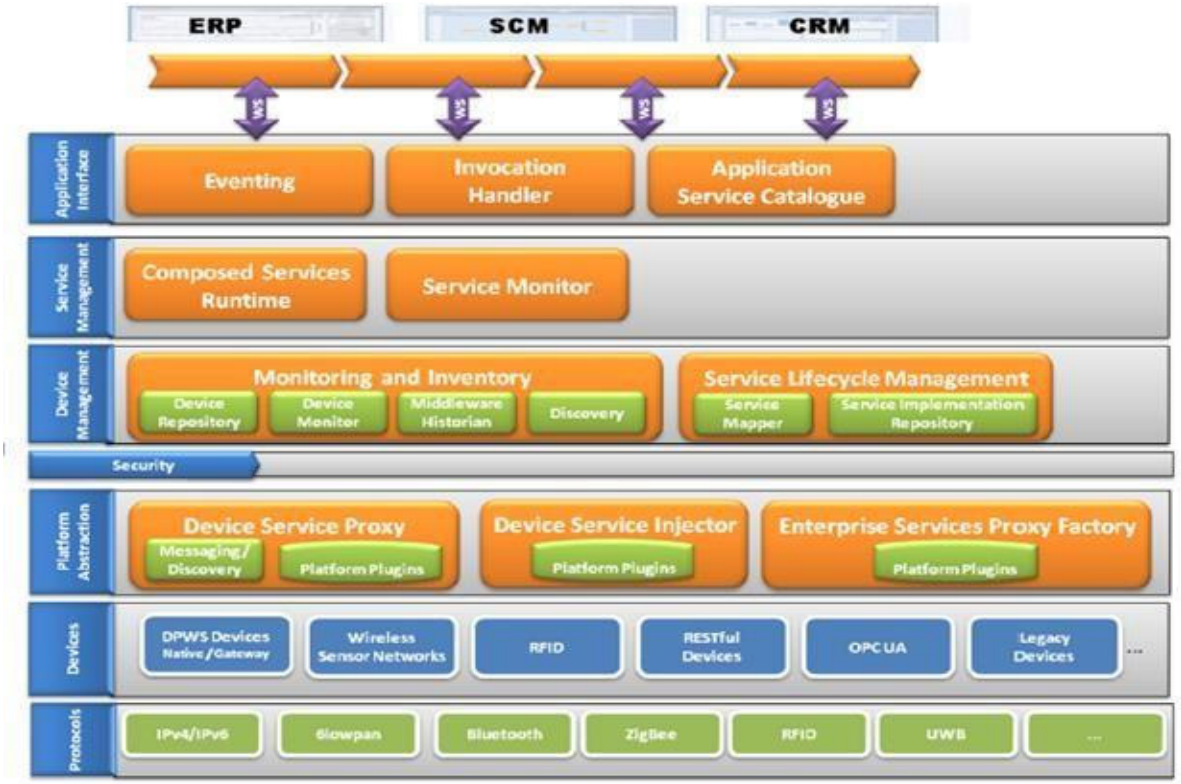


Figure 4: The SOCRADES Integration Architecture (SIA) enabling the coupling of(industrial) machines at shop-floor and enterprise systems

The SOCRADES Integration Architecture distinguished various levels such as:

- Application Interface: This part enables the interaction with traditional enterprise systems and other applications. It acts as the glue for integrating the industrial devices, and their data and functionalities with enterprise repos and traditional information stores.
- Service Management: Functionalities offered by the devices are depicted as services here to ease the integration in traditional enterprise landscapes. Tools for their monitoring are provided.
- Device Management: Includes monitoring and inventory of devices, including service lifecycle management.
- Platform Abstraction: This layer enables the abstraction of all devices independent of whether they natively support WS or not, to be wrapped and represented as services on the higher systems. In addition to service- enabling the communication with devices, this layer also provides a unified view on remotely installing or updating the software that runs on devices.

- Devices & Protocols: These layers include the actual devices that connectover multiple protocols to the infrastructure. The respective plug-ins of course needs to be in place so that they can be seamlessly integrated to SIA.

To realize discovery and interaction in a P2P way, a local gateway/service mediator is implemented. This prototype was named a Local Discovery Unit (LDU), and would enable the dynamic discovery of devices on premise andtheir coupling with the SIA.

SMART GRID:

The "smart grid" is the next generation of those energy systems, which have been updated with communications technology and connectivity to drive smarter resource use. The technologies that make today's IoT-enabled energy grid "smart" include wireless devices such as sensors, radio modules, gateways and routers

An intelligent networked device is emerging whose M2M interactions create new capabilities in the monitoring and management of the electricity grid and the interaction between its stakeholders. IT-empowered innovations integrated with the electricity network and the stakeholders' interactions have paved the way towards a "Smart Grid" that takes advantage of sophisticated bidirectional interactions.

The National Institute of Standards and Technology (NIST) Smart Grid Conceptual Model defines a framework that outlines seven domains: Bulk Generation, Transmission, Distribution, Customers, Operations, Markets and Service Providers. Complementary to that, the IEEE views the Smart Grid as a large 'System of Systems' where each NIST Smart Grid domain is expanded into three Smart Grid foundational layers: (i) the Power and Energy Layer, (ii) the Communication Layer, and (iii) the IT/Computer Layer (Figure 12.1 below).Layers (i) and (iii) are enabling infrastructure platforms of the Power and Energy Layer that makes the grid 'smarter'.

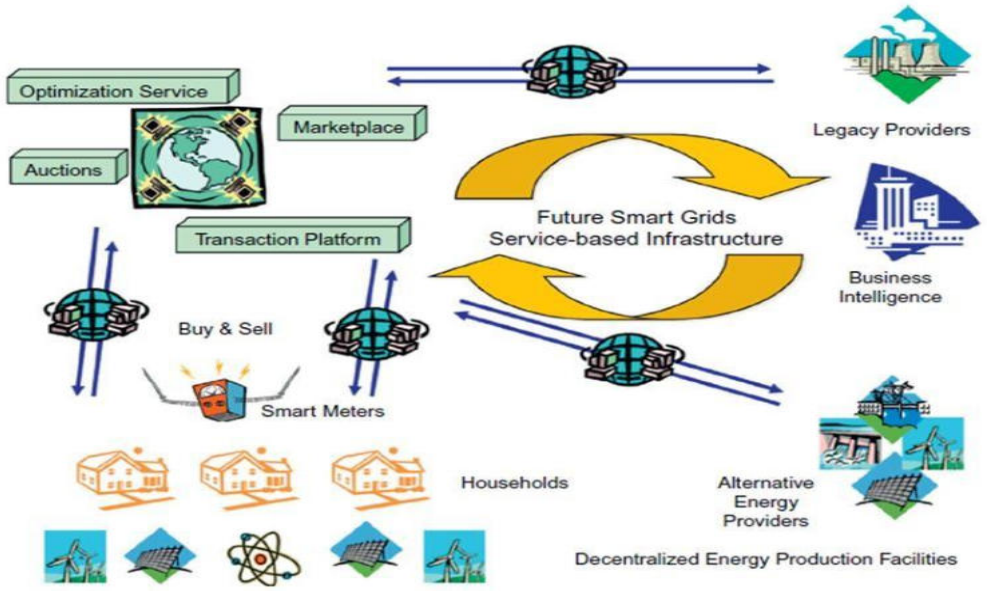


FIGURE 12.1

The future ICT-empowered interaction-rich Smart Grid.

The integration of small, highly distributed energy production sources and their coupling with advanced, information-driven services will give rise to a new infrastructure (i.e. the Internet of Energy), whose key technology building blocks are the IoT and Internet of Services (IoS). As shown in Figure 12.2, (below) IoT and IoS make possible diverse M2M interactions locally and over the Internet, empowering traditional business relationships with fine-grained monitoring and control of the large energy infrastructure. The result is the Internet of Energy (IoE), which is the vision of the Smart Grid, spanning not only the technical grid infrastructure, but covering all energy-related aspects of the grid, its devices (including appliances, etc.), their interactions, as well as high-level applications and systems that depend on their data.

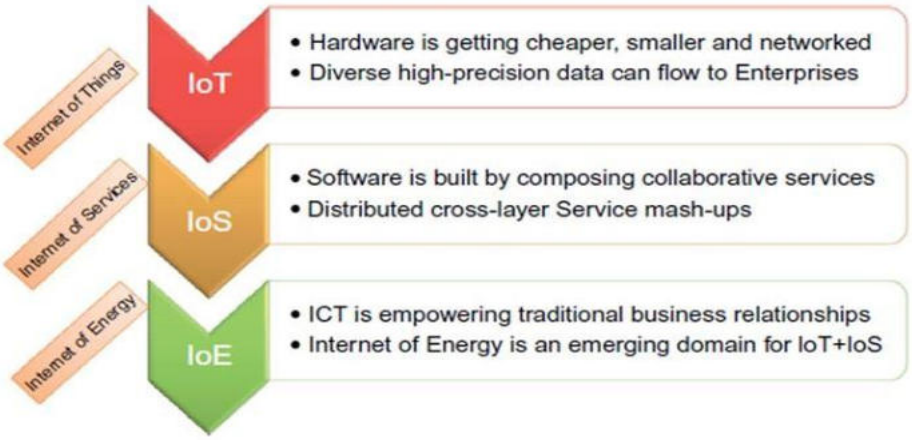


FIGURE 12.2

The Smart Grid technology based on a combination of IoT and IoS empowering old and new innovative energy-related interactions.

Smart metering:

Traditional utility management processes today involve the collection of metering data from a centralized point (the meter) in the household once or twice a year. With the emergence of smart metering, the collection of data from households has increased, with the aim to have measurements (or less) at 15- minute intervals.

Figure 12.3 depicts the paradigm change from an infrastructure delivering metering data for billing purposes towards a general-purpose monitoring infrastructure that acts as an enabler for a multitude of stakeholders and value- added services. Data generated include not only the energy metering data such asconsumption (or production of energy), but also other data (e.g. related to power quality, device status, etc.), which may provide additional added value towards asset management. Such smart metering infrastructures are currently under investigation for their performance, scalability, as well as the value-added information they can deliver beyond smart metering (e.g. towards energy management, asset management, integration of heterogeneous systems, etc.).The new smart meters, or their extensions (such as dedicated devices depicting energy consumption and other info), are used as communication media where the utilities push information (for example) about the current energy tariff, upcoming maintenance, costs, etc.

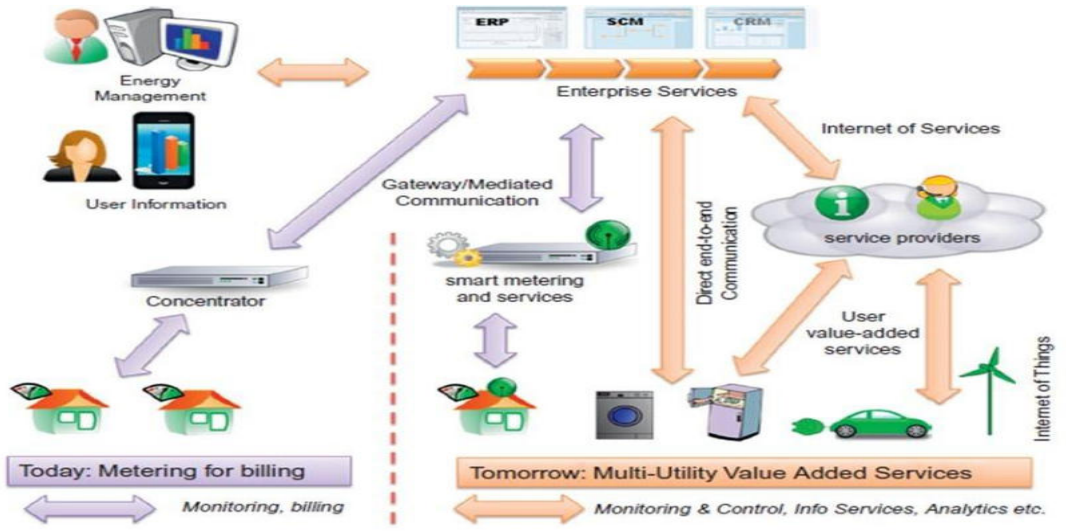


FIGURE 12.3

Smart Metering enabling multi-utility value-added services.

COMMERCIAL BUILDING AUTOMATION:

Introduction

A Building Automation System (BAS) is a computerized, intelligent system that controls and measures lighting, climate, security, and other mechanical and electrical systems in a building. The purpose of a BAS is typically to reduce energy and maintenance costs, as well as to increase control, comfort, reliability, and ease of use for maintenance staff and tenants.

Some examples use cases:

- Control of heating, cooling, and ventilation based on time of day, outside temperature, and occupancy (e.g. Morning Warm-up).
- Automatic control of air handlers to optimize mix of outside air in ventilation based on, for example, inside temperature, pressure, and time of day.
- Supervisory control and monitoring to allow maintenance staff to quickly detect problems and perform adjustments.
- Out-sourcing of monitoring and operations to a remote operations center.
- Data collection to provide statistics and facilitate efficiency improvements.
- Alarms for high carbon monoxide and carbon dioxide levels.
- Individual metering per apartment (to give incentive to save energy in multi-tenant buildings).
- Intrusion and fire detection.
- Building access control.

A BAS is normally distributed by nature to allow every sub-system to continue operation in case of failure in another system. A BAS consists of the following components (Figure 13.1):

- Sensors (i.e. devices that measure, such as thermometers, motion sensors, and air pressure sensors).
- Actuators (i.e. controllable devices, such as power switches, thermostats,and valves).
- Programmable logic controllers (PLCs) that can handle multiple inputs and outputs in real time and perform regulating functions, for example.
- A server which monitors and automatically adjusts the parameters of the system, while allowing an operator to observe and perform supervisory control.
- One or more network buses (e.g. KNX, LonWorks, or BACnet).

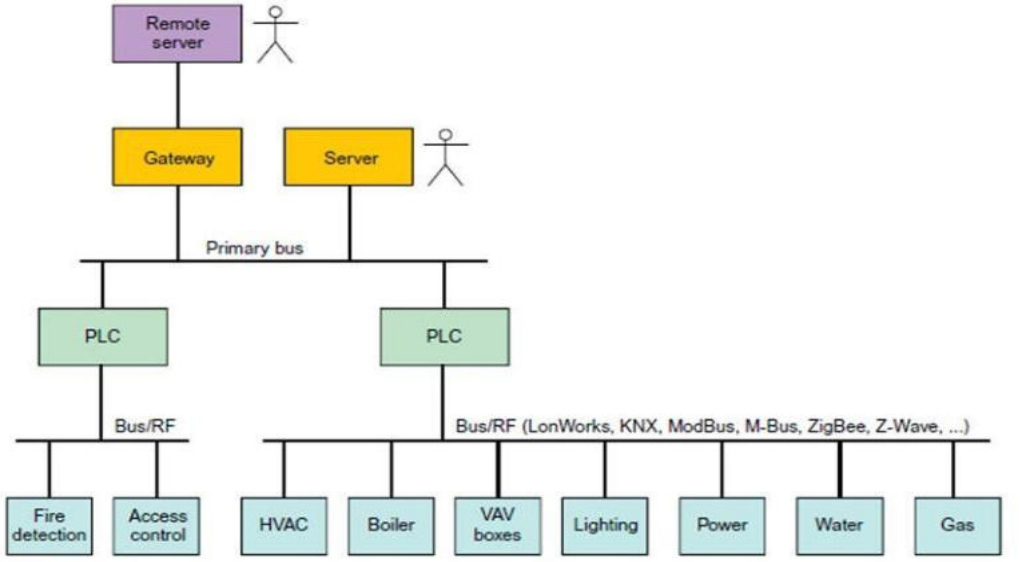


FIGURE 13.1

Central parts in a BAS.

Case study: phase I_ commercial building automation today:Background

Company A wants to improve energy efficiency in their buildings and become Green Building Partner certified, which requires lowering their energy consumption by at least 25%.

After discussions with a building automation company (Company B), they have come to understand that this is a very good investment that will quickly justify itself in terms of reduced energy costs. They agree on a five-step plan that starts with collecting data from the buildings, followed by analysis, adjustments, and connecting the systems in the buildings to a local-server, and finally connecting the buildings to a remote operations center.

They can now start with collecting data from existing systems. In some cases, this requires new meters to be installed. Everything from water usage to heat and electricity consumption is logged continuously, as well as performance of the ventilation and room temperatures.

By comparing the key performance indicators with comparative figures, the need for corrective actions is assessed and used as a basis for an action plan that consists of adjusting the existing systems and installing new software. These adjustments quickly increase the efficiency of the systems and are continuously optimized during the project. Examples of adjustments are hot water temperature, improved control of indoor temperatures as well as better control of fan and pump operation to avoid unnecessary operation.

One of the most central features of the improved system is the new web-based E-report. It provides information about current energy consumption and other key parameters from the buildings. This information is used to make both short- term decisions as well as long-term planning.

Everyone has access to the web portal because it's not only important for the maintenance staff, but also needed to create awareness across everyone in the company.

The next phase of the project consists of connecting the systems in the buildingsand analysing the dynamics to be able to perform intelligent control. This both improves performance as well as reduces maintenance costs.

The final step to completion involves setting up a web-based Supervisory Control and Data Acquisition (SCADA) system for remote monitoring of the building systems. Through the web portal, the users can access information from the buildings in a coherent manner. Company A decided to outsource the operations and daily maintenance of the systems to Company B by utilizing their cloud-based offering. Company B's remote operations centre is continuously monitoring the building systems. When

building system operations deviate from their expected behaviour, Company A's maintenance staff and their supervisors are notified by SMS and email. Typical events that can trigger a notification are, for example, mechanical failures or undesirable temperature deviations. Apart from notifications, Company B can also assist with equipment operation and adjustments remotely. For Company A, this arrangement is perfect because their in-house maintenance staff can respond to an alert 24 hours a day.

For Company A, the most important improvement has been the 35% energy reduction after the completion of the project. Another critical aspect has been the knowledge transfer from the experts at Company B that allows Company A to maintain the efficiency of the systems as well as the ability to continuously improve the operations of them.

Technology overview:

A setup for Company A is depicted in Figure 13.2 (below). Each building is equipped with a set of meters and sensors to measure temperature, water consumption, and power consumption, as well as one or more PLCs.

As seen in Figure 13.2, the PLCs perform real-time monitoring and control of the devices in the building. They also feature a user interface for configuration and calibration of (for example) the regulators, curves, and time relays. It is possible to remotely configure the PLCs from the Operations Centre using the PLC Control system, which is connected to the PLC via a 3G-modem and an Internet Protocol (IP) modem that converts between RS-485 networks and Transmission Control Protocol/Internet Protocol (TCP/IP) networks. The PLCs communicate with the devices using several protocols, such as M-BUS, analog, digital, and Z-Wave, which is a low-power radio mesh-network technology. All logic necessary to operate the buildings is contained within the PLCs, allowing for minimal bandwidth requirements on the connection towards the Operations

Centre as well. It also means that the building systems can remain fully operational during periods of network outage.

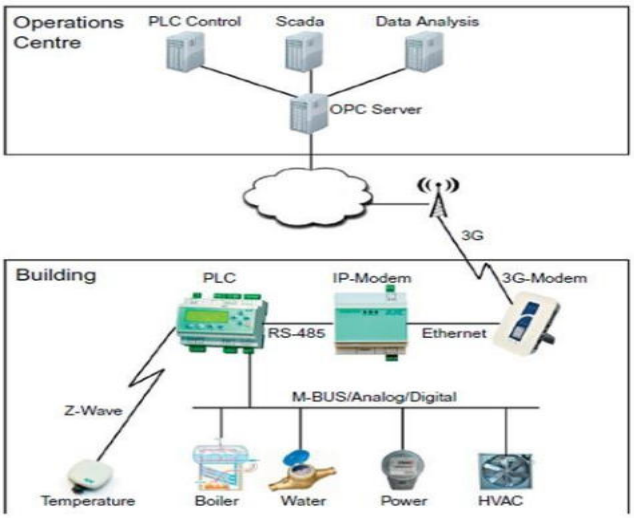


FIGURE 13.2
Illustration of the BAS.

The OLE for Process Control (OPC) server provides access to data, alarms, and statistics from the PLCs. When a value is requested from a user, a request is sent from the user's OPC Client to the OPC Server, containing an OPC Tag thatidentifies which PLC to contact and which value to ask for. The type of OPC communication used is called OPC DataAccess. The OPC server then contacts the PLC in question and asks forthe value using a protocol supported by the PLC (LonWorks or ModBus).

The SCADA system is used for operational monitoring of the buildings and provides information from all the relevant building systems. It uses the open and standardized OPC protocol, which enables integration with devices from many different vendors. The maintenance and operations staff can connect to the system using a web browser with a username and password to accessdynamic flowcharts, drawing tools, timers, set points, actual values, historic readings, alarm management, event logs, as well as configuration for notifications over email, fax, or SMS.

The Data Analysis server logs all historical readings from the buildings and makes it possible to follow up on different aspects of the energy and resource consumption, satisfying the varying needs of the tenants, economy department, and landlord. Through the OPC server it's possible togather readings from all the building systems, regardless of vendor.Typical reports include trends, cost, budget, prognosis, environment, andconsumption of electricity, heating, water, and cooling.

Value chain:

Figure 13.3 shows an applied value chain.

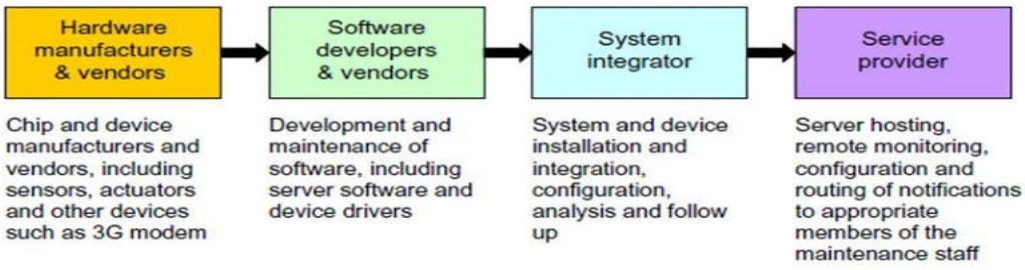


FIGURE 13.3
Applied value chain for Company A's system.

Case study: phase II commercial building automation in the future:Evolution of commercial building automation

Two major factors will drive the evolution of Building Automation: informationand legislation (Figure 13.4).

Access to well-packaged information will provide the basis needed for decisionsand behavioural changes. This can (for example) be electricity prices or where and when energy is used, and will allow for well-founded decisions that providethe best results.

Legislation, and taxes or tax credits to some degree, will provide the second driver. Legislative demands on green buildings and the Smart Grid will give rise to new opportunities, such as Demand/Response, MicroGeneration, and Time- of-Day Metering.

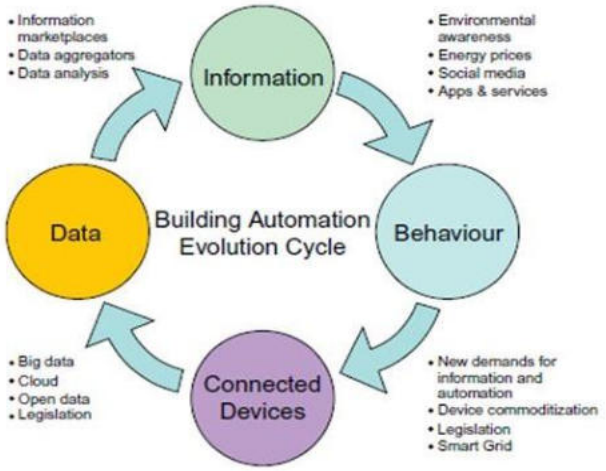


FIGURE 13.4
Building automation evolution cycle.

Background

A few years have passed and Company A has decided to outsource the maintenance of its buildings to a local contractor who provides services to several other customers in the neighbourhood. This will save money since that will enable them to utilize a shared caretaker pool. At the same time, they plan to upgrade their buildings to become fully automated with, for example, occupancy sensors, automated lighting, and integrated access control. To make this cost efficiently, they intend to make use of the existing IP infrastructure in the buildings, which also saves on operating expenditures as the network administrators can also manage the BAS infrastructure. According to studies, a converged IP and BAS network can reduce maintenance costs by around 30% while also lowering the initial investment for installation and integration by around 20% (according to studies performed by Cisco®). A shared infrastructure also leads to increased energy efficiency. New political incentivesin regards to energy efficiency have increased the development pace in the building automation area. Many neighbouring buildings in Company A's area are now fitted with building automation which allows for sharing of information and resources. The increased customer base has also enabled new niches in the value chain, which has been split up to a large degree. Where before the rule was to have one single integrator and service provider, we now see a multitude of new actors, such as specialized service providers for remote monitoring, security, optimization, data collection, and data analytics. This allows CompanyA to choose freely what combination of service providers to use, while also providing a smooth transition when moving to a new provider. This is made possible by a new niche in the value chain: the Cloud Service Broker (Figure 13.5).

The process of integrating with the maintenance contractor's systems is simplified by the service broker because it provides immediate access to Company A's BAS. The caretakers can use their own specialized softwareas theservice broker provides a bridge that can convert between several common protocols used for building automation.

When it comes to selection of devices, Company A opts for using standardized protocols to avoid vendor lock-in. They also decide to keep certainparts of the old system, as these would be too costly to replace. To still benefit from a fully integrated system, they also invest in a constrained application protocol (CoAP) gateway that translates between legacy devices and the new system.

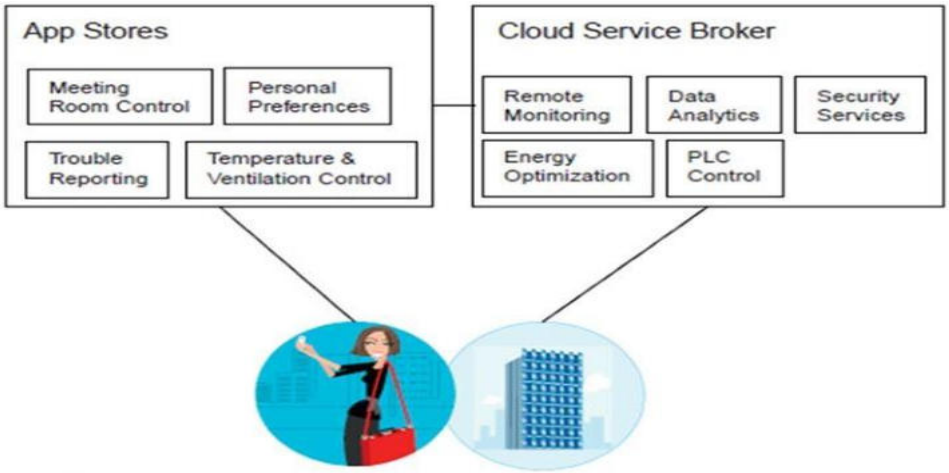


FIGURE 13.5

Cloud Service Broker.

Technology overview

The rapid development of IP technologies for Smart Objects, it is possible to use IP for both constrained devices, such as battery powered sensors and actuators.

The new system is to a large degree based on IP technology (Figure 13.6). There are several IP-based protocols to select from, but in this case CoAP and Sensor Markup Language (SenML) were selected. CoAP provides both automatic discovery as well as a semantic description of the services the device provides.

This drastically reduces installation costs, as much less configurationis needed. CoAP is similar to HypertextTransfer Protocol (HTTP), but is binary to reduce the size of the messages. It also defines a Representational State Transfer (REST) like Application Programming Interface (API) optimized for M2M applications. As with HTTP, a format for the content is also needed, in this case SenML, which is used as a format for sensor measurements and deviceparameters.

As mentioned before, there are still a few legacy devices, and these need a gateway to enable communication with the IP-based systems.

A local Resource Directory and data cache is also installed to keep track of all the devices in the company network. This allows local lookups of devices and data, and serves as a safe guard in case of failure.

To protect the system from intruders, a normal network firewall is used. For the connection towards the service broker and the service providers, a permanent Virtual Private Network (VPN) connection is established.

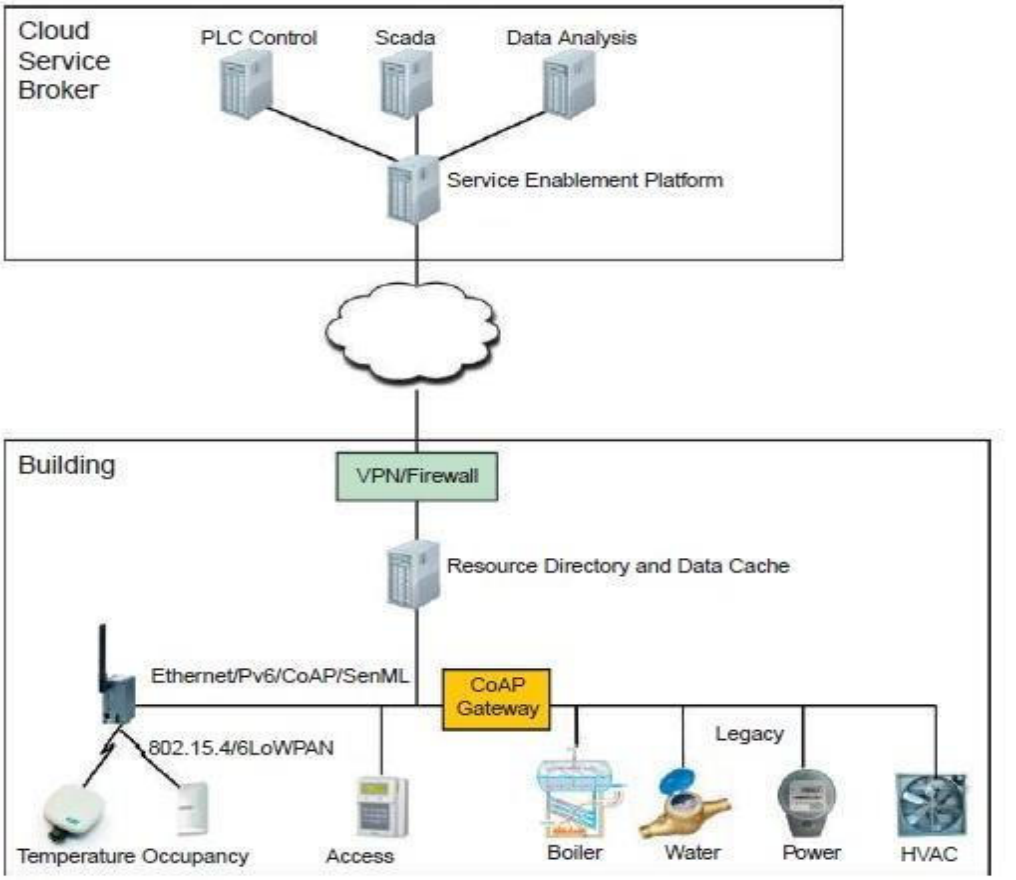


FIGURE 13.6

Architectural overview of the upgraded system.

Evolved value chain for commercial building automation:

As the demand for M2M services grows, new niches in the value chain will emerge, such as information brokers, service brokers, and service enablement providers (Figure 13.7). These will enable new use case areasby allowing vertical domains to be integrated (e.g., security, energy, waste-management, police, and public transport). It will also provide the openness needed for third- party service providers to create apps and socialmedia integration, to allow for (for example) comparisons with neighbouring buildings, competitions, and end user involvement. Privacy and security will, however, be essential to build up the trust needed for this ecosystem to develop.

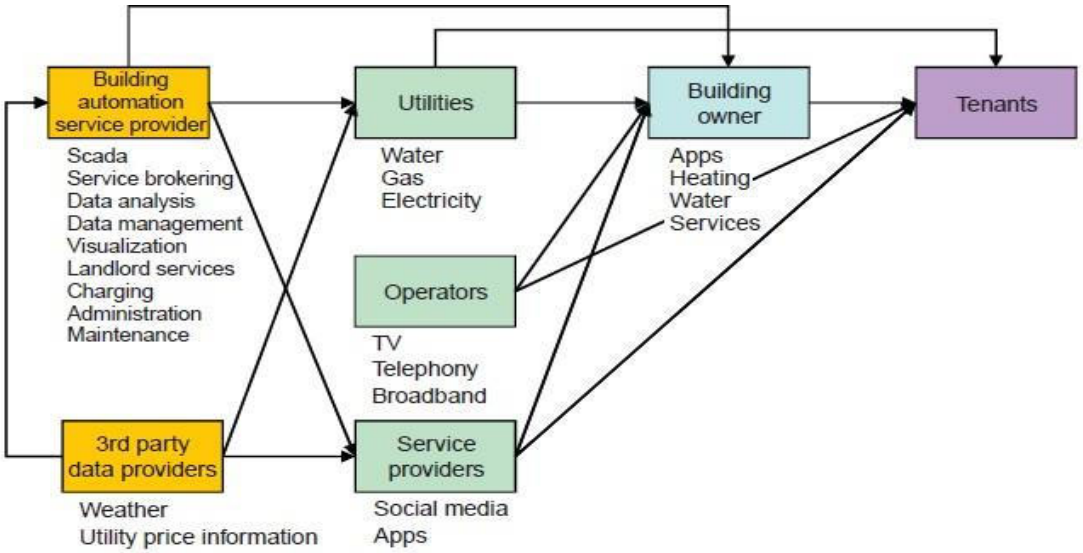


FIGURE 13.7

Evolved value chain for building automation.

SMART CITIES:

Introduction

The cities will continue to grow, it is predicted that 70% of the world's population will be living in cities by 2050. In addition to this fundamental shift in the organization of human society, we are also faced with increasing natural resource constraints, marked increases in population, and a restructuring of the global economy. Existing cities and the new ones that will be built will therefore need to handle massive tensions in three spheres simultaneously:environmental impact, economic growth, and social evolution. This will affect every city in the world; for example, failure to provide new employment opportunities in the face of massive economic change and provide adequate environmental protection will have a profound effect on the inhabitants of cities.Competition between cities is also set to increase, as cities need to create post- industrial identities that are attractive to employees and employers, and tourists, and can also provide communities the ability to combine their resources to overcome common problems rather than relying on increasingly scarce government resources. The pressure and tensions experienced within cities today are only just beginning.

As a result, so-called "Smart Cities" have rapidly become a hot topic within technology communities, and promise both improved delivery of services to endusers and reduced environmental impact in an era of unprecedented urbanization. Both large, high-tech companies and grassroots citizen-led initiatives have begun exploring the potential of these technologies. A multitudeof so-called "smart city" projects are being developed across the world.

Smart cities-a working definition:

The smart cities are a complex and multi-layered problem with multiple stakeholders that go far beyond pure technology solutions. For the purposes, we define a smart city as one that uses data and ICT in order to:

- Manage and optimize existing infrastructure investments and plan for new investments more effectively.
- Provide more efficient, new, or enhanced services to citizens.
- Reduce organizational silos in cities' service delivery and create newlevels of cross-sector collaboration.
- Assist city and government's progress toward meeting climate changemitigation and adaptation goals.
- Enable innovative business models for public and private sector serviceprovision.

By aligning the interests of stakeholders and employing new technologies and new market mechanisms, cities will be better able to capture the full value of ICT investments made in order to become a smart city.

Smart cities-some examples:

There are many different types of "smart city" solutions; these include:

- Smart Transportation solutions for parking, traffic monitoring,public transport, or municipal fleet management.
- Smart Healthcare, which includes electronic records management,hospital asset management, and remote monitoring systems.
- Smart Education, including eLearning, MOOCs, and connectedcampuses.
- Security and Public Safety, such as video surveillance, enhancedanalytics, and workflows for emergency services.
- Smart Buildings, which include smart meters, light, heating, internalsecurity systems, and water and waste management.

Roles, actors, engagement:

One of the biggest complexities associated with cities is the sheer number of actors involved in them. Moreover, many of these actors have multiple roles within a city context. In comparison to the other use cases, smart cities are much more difficult to implement due to the sheer number of actors and stakeholders. In fact, smart city solutions are often the combination of several of the other use cases described for example, a combination of a participatory sensing and a smart grid scenario. Figure 14.1 illustrates some of the complexity within cities. A city may be viewed as a nexus of several infrastructures and social constructs.

For example, cities have buildings not just for corporations, but for citizens and government services as well. Infrastructure covers a broad range of meanings, from the actual pipes laid down for transmission of water or the cables for transmission of electricity, all the way through to the garbage collection and recycling mechanisms.

In contrast to the other use cases, people are both key barriers and key opportunities for change within a smart city use case. A citizen, for example, holds a number of different roles within a city; for example, as a parent who needs to interact with schools in the district, an employee who needs to access city infrastructure in order to get to work, a voter, or as an end user of city services such as waste management.

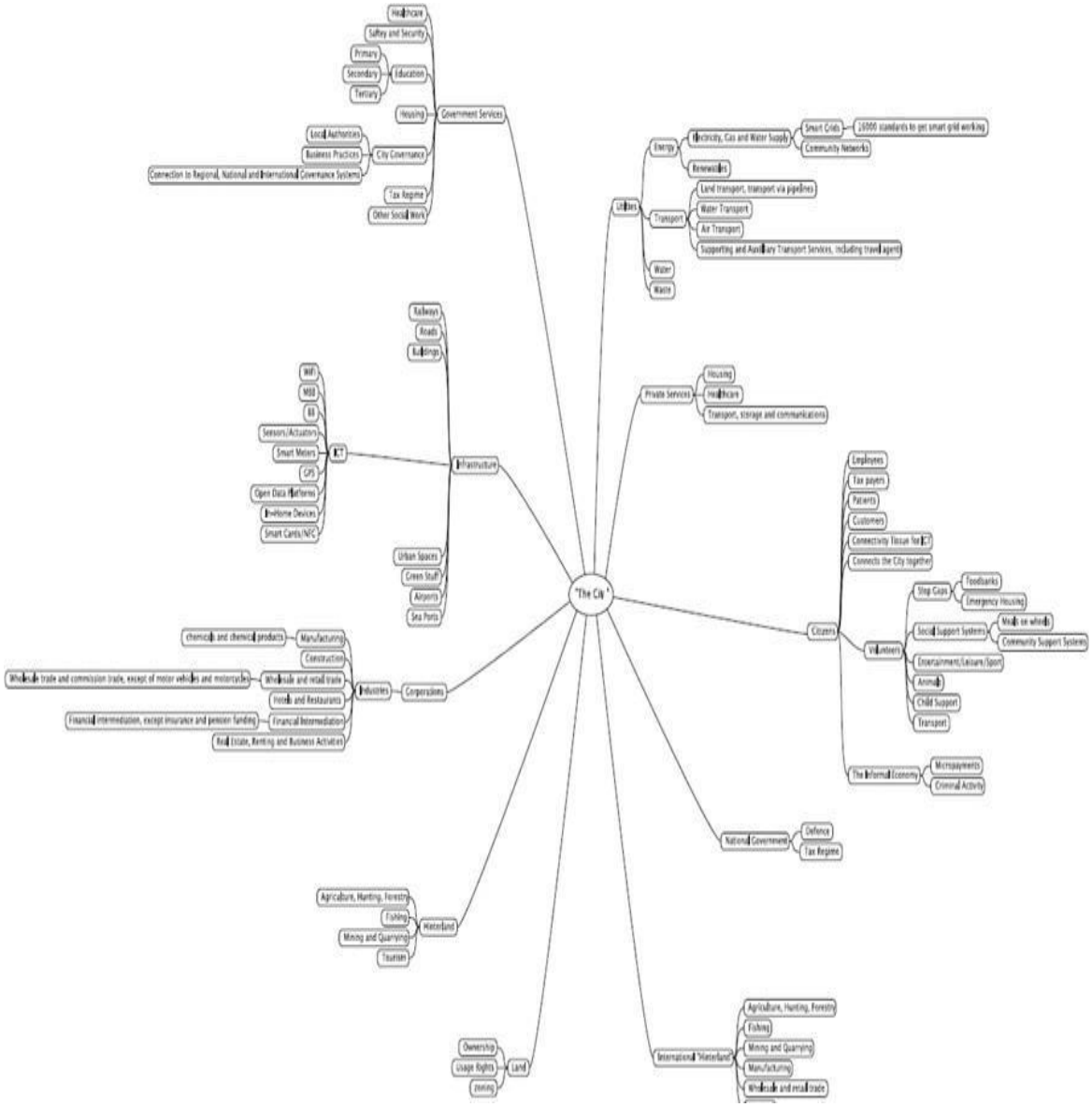
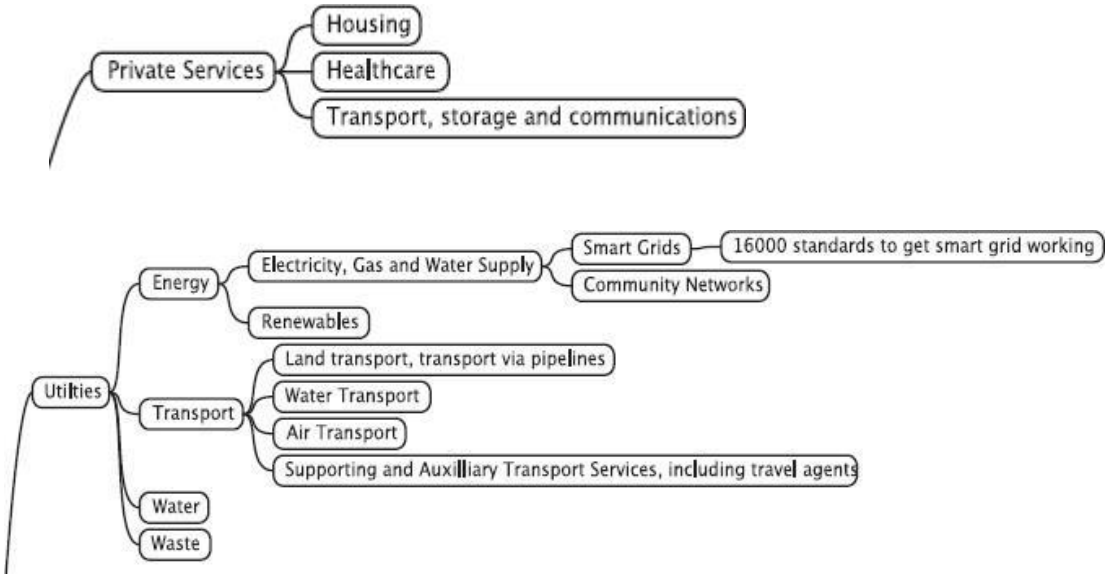
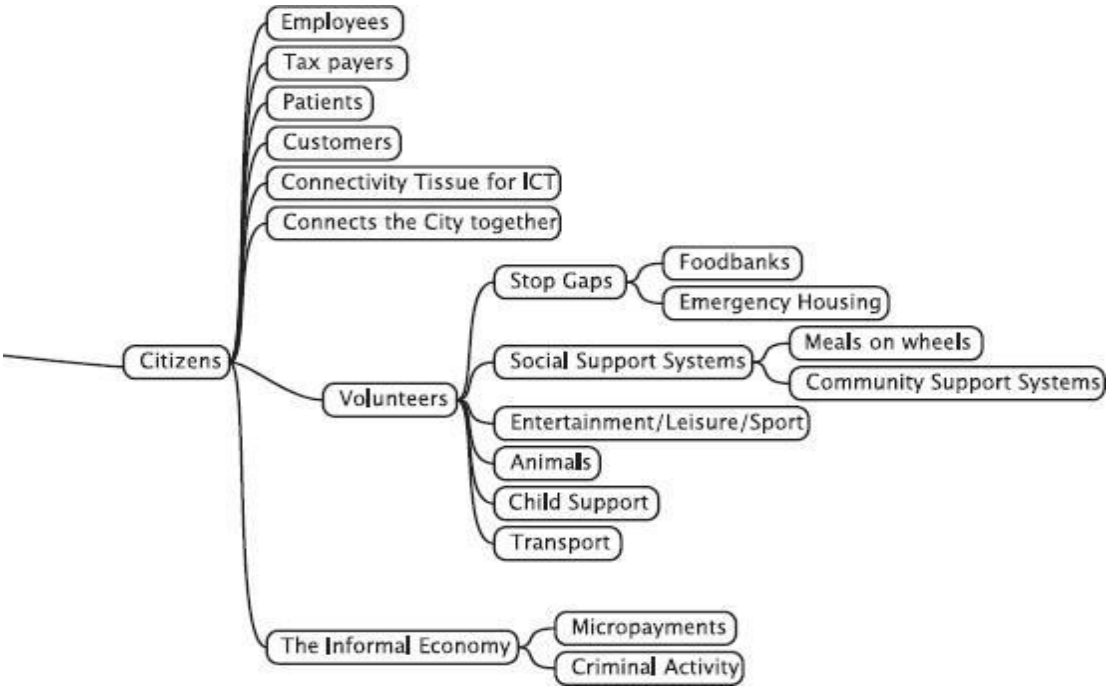
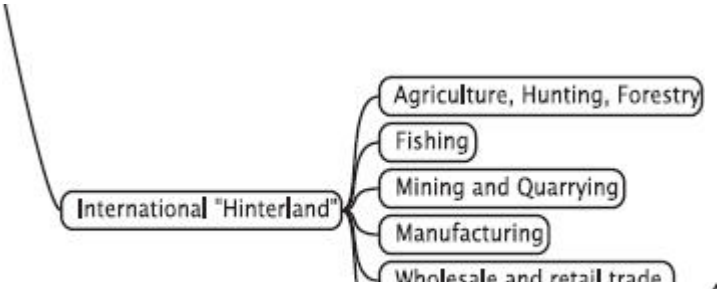


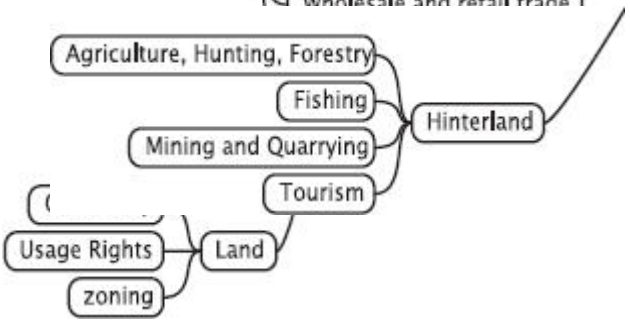
Figure 14.1Complexity within cities.[Beginning of Rough work:

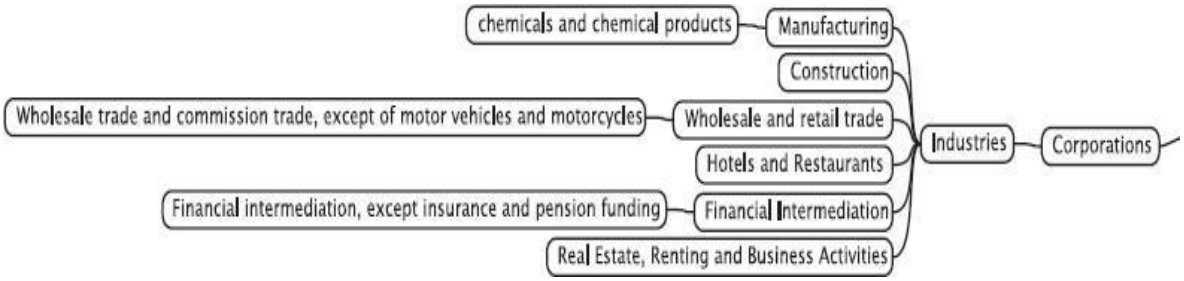


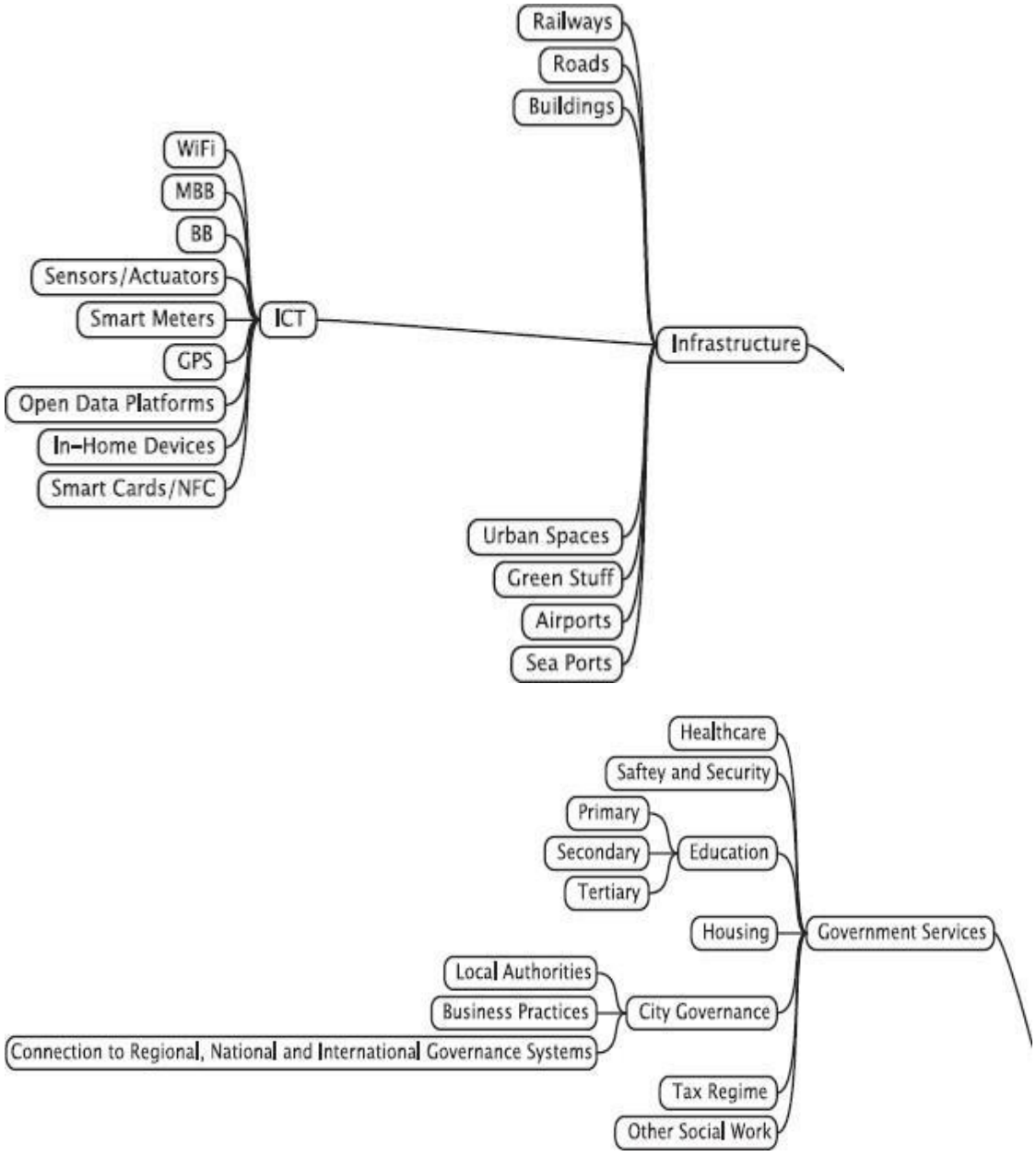

Citizens
Employees
Tax payers
Patients
Customers
Connectivity Tissue for ICT
Connects the City together
Volunteers
Stop Gaps
Foodbanks
Emergency Housing
Social Support Systems
Meals on wheels
Community Support Systems
Entertainment/Leisure/Sport
Animalls
Child Support
Transport
The Informal Economy
Micropayments
Criminal Activity


National Government
Defence
Tax Regime


International "Hinterland"
Agriculture, Hunting, Forestry
Fishing
Mining and Quarrying
Manufacturing


Hinterland
Agriculture, Hunting, Forestry
Fishing
Mining and Quarrying
Tourism
Land
Usage Rights
zoning


Industries
Corporations
Manufacturing
chemicals and chemical products
Construction
Wholesale and retail trade
Wholesale trade and commission trade, except of motor vehicles and motorcycles
Hotels and Restaurants
Financial Intermediation
Financial intermediation, except insurance and pension funding
Real Estate, Renting and Business Activities



Transport and logistics an IoT perspective:

"Transport" generally corresponds to the delivery of people and goods via road, rail, air, and sea (Classifications 49, 50, 51, 53 in the United Nations Statistics Division's International Standard Industrial Classification of All Economic Activities). "Transport infrastructure and services" refers to all infrastructure and services that support transport, as well as the manufacture of vehicles and equipment for monitoring and servicing transport.

These sectors are those that deliver day-to-day travel and delivery services in and around cities for home, work, and leisure activities. For the sake of simplicity, these are collectively called "intraurban" transport to distinguish it from long distance intercity transport.

Cities face dramatic challenges in the coming decades with the management of transport systems. It is predicted that soon there will be nearly 6 billion cars on the planet; in addition to this, 180,000 people move into cities each day, placing pressure on already stretched transport infrastructures.

Physical infrastructure for transport:

Transport infrastructure is, as mentioned previously, complex and deeply embedded in the natural environment

of a city. From roads to rail networks, the decisions made about how to make decisions about when and where to implement transport infrastructure are critical to the future success of a city. Today's modern transport infrastructure industry has often been split up into several areas of control; while there are national differences between how transport is implemented, there are generally some broad categories. Each of these actors may implement IoT in a different way:

Manufacturers: Manufacturers build and sell infrastructure assets (e.g. trains, tracks, buses). Example companies include Siemens. Infrastructure Managers: Infrastructure managers buy infrastructure from manufacturers and implement, manage, and maintain the infrastructure on behalf of a city or a government. Examples include the rail networks in the UK or the road infrastructure authorities responsible for building and maintaining national road infrastructure.**Operators:** Operators are those companies that run the actual services on the infrastructure. These companies often set timetables and are the ones that most end-user consumers know and have customer relationships with. Examples include Virgin Trains in the UK rail industry, or Greyhound coaches, who operate coaches on the road infrastructure in the U.S.

Maintainers: Maintainers are those entities that are responsible for the day-to- day running of different transport infrastructures. For example, ensuring that thetrains are kept serviceable and that roads are properly maintained. In many countries, these are outsourced operations paid for by the Operators.

End Users: End users are the people who are actually using the services; they can be train passengers, coach passengers, and in some cases, even drivers of their own cars.

Regulators: A key aspect of the transport infrastructure in any nation is the regulator. These provide regulations that companies need to abide by in a variety of situations. Examples include anti-monopoly regulations, anti-price setting, management of timetabling information, and also setting health and safety standards for end users and staff working on the infrastructure itself.

The actors involved in the physical infrastructure are illustrated in Figure 14.2.

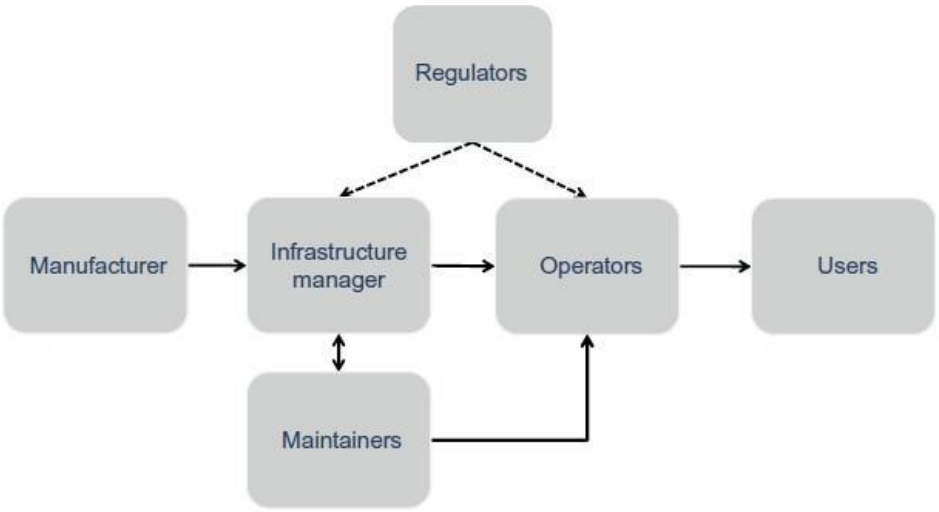


FIGURE 14.2

Actors in the Transport Value Chain.

Information marketplace for transport and logistics:

A framework of IoT Information Marketplace for Transport in Cities is shown in Figure 14.3.

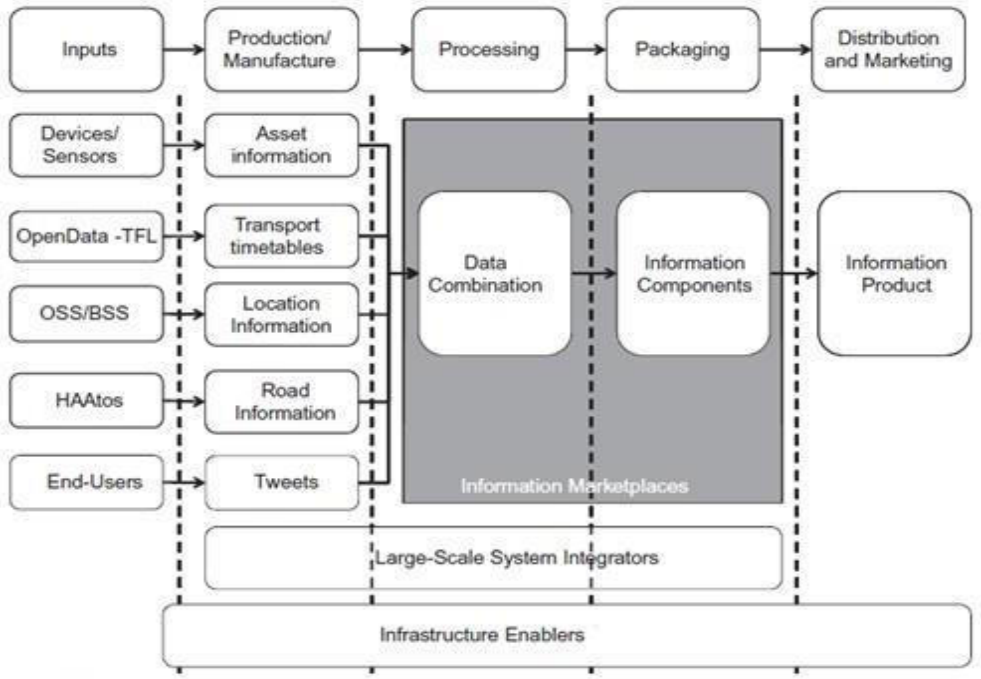


FIGURE 14.3

IoT Information Marketplace for Transport in Cities.

We focus on how they connect together to provide a unified information value chain.

Inputs: The devices and sensors here include the sensors on, for example, rail tracks that provide information about the following:

- **Traffic**: Amount and weight of traffic over the road or on a rail track.
- **Subsidence:** Whether the infrastructure in question has moved or changed. These sensors can help indicate when a critical problem on infrastructure may cause a major incident.
- **Infrastructure Temperature and Operating Conditions:** These sensors measure heat, humidity, and other environmental parameters that may affect the asset performance or longevity.
- **Open Data:** Open data may be used as an input into a transport information value chain in the form of maps, transport data, timetable data, and the current performance of the transport network in question.
- **OSS/BSS**: The Operational Support Systems and Business Support Systems of mobile operator networks are also important inputs to information value chains.
- **Corporate Databases:** A significant number of corporate databases are available within the transport networks of countries. These include the manufacturers' corporate databases of which operators they have sold to, and how the transport asset has performed.
- **End Users:** Perhaps the most significant input for travel in the new era ofIoT is the end user. End users now have the ability to input into the Information Marketplace for transport via, for example, tweeting about the performance of the transport infrastructure.

Processing: During the processing stage, data from various sources is mixed together. These data sources are combined together to create insight and information necessary for transport decision-making.

Packaging: After the data from various inputs has been combined together, the packaging section of the information value chain creates information components. These components could be produced as charts or other traditional methods of communicating information to end users.

PARTICIPATORY SENSING:

Introduction

Participatory Sensing (PS) also known in the literature as Urban, Citizen, or People- Centric Sensing, is a form of citizen engagement for the purposes of capturing the city surrounding environment and daily life. The first PS projects appeared in the early 2000s and focused more on city dweller campaigns to capture problematic situations(such as road faults, air pollution, low-lit parts of the city), daily routines(such as commuting by bike/car), personal individual health, or even combinations thereof. More recently, the concept has gone through a transformation in terms of branding, people engagement practices, target audience, and business model. Nevertheless, the main constituent of any PS activity is the city inhabitants acting as human sensors, and often a send users of an end PS product.

Roles, actors, engagement:

The main potential actors in a PS system are "individuals" and "city authorities." The city inhabitants have access to sensing devices and can use them to capture their environment. Individuals can be one group of the receivers of the collected informationand the target for environmental changes based on PS. The city authorities can be the receiver of the collected information and potentially the organizer of the sensing/collection campaigns. City authorities can also analyse the collected information, potentially with individual citizens, set plans of actions, and follow-up on the actions.

There are several roles for the citizens or city authorities in a PS system. An individual can potentially act as the Data Collector using her mobile phone to collect sensor data. An individual or city authority can play the role of the Collection System Operatorwho owns and operates the collection system from multiple data collectors. An Analysis Provider that processes, stores, and analyses the collected data can be assumed by any party (individuals, cities) that could also prepare plans of, and execute actions based on, the conclusions of the campaign (Action Responsible).As a general observation, the city authorities typically assume the non-data collector roles (Collection System Operator, the Analysis Provider, and the Action Responsible) while the citizen's major role is that of a Data Collector, not excluding, however, any other combination of assignments of roles to actors.

Collective design and investigation:

In this model individual citizens design the sensing campaign, participate in the data collection, and analyse and interpret the collected data. Therefore, the citizens are fullyempowered to contribute to the change that they would like to see in their living environment.

Public contribution:

Individual citizens only take active participation in the data collection phase organized by another individual or organization (e.g., city authorities), but they don't necessarily analyse or interpret the results.

Personal use and reflection:

Individual citizens monitor and record their daily lives for themselves without any organized campaign. A person may choose to refrain from sharing personal information and details, or may choose to share certain specific information or aggregation of collected information. Therefore, the collected data are used mainly for personal reflection or sharing and reflection within a very small and private group (e.g., individual's relatives).

Participatory sensing process:

The basic steps of a typical PS process are shown in Figure 15.1.During the Coordination phase the participants need to either organize themselves, or be recruited by some other entity (e.g. city authorities) within the context of a sensing campaign, and the objective of the campaign needs to be communicated among all of them. Then the participants spend some predetermined amount of time to capture (Capture phase) the desired sensing modalities using their mobile phone applications or custom designed applications for the sensing campaign. The data entering a PS system does not need to originate only from the data collectors.

Several other publicly available sources, such as weather, air quality, and traffic reports, could be used for drawing richer conclusions. The collected data are transferred (Transfer & Storage phase) to the data collection system through the phone connectivity options and stored in Internet servers (private or public).

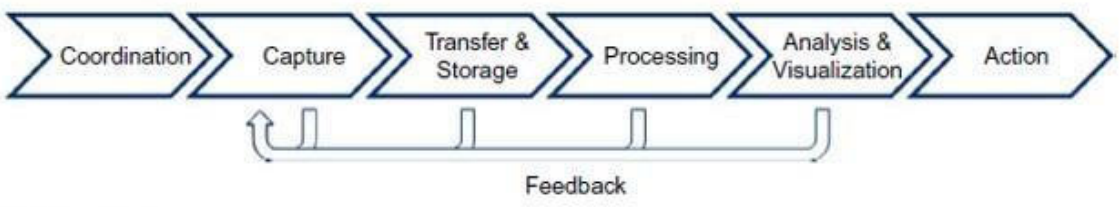


FIGURE 15.1

Typical Participatory Process steps.

The data are then subject to pre-processing (Process phase)so that the privacy of the data collectors is preserved, and access control rules are added so that the data can be accessed anytime by only authorized individuals or services. The collected data are analysed by relevant analysis tools, aggregated (if possible), correlated with each other in order to detect patterns, and in the end visualized for better understanding for the target group of the campaign (Analysis and Visualization phase). Last but not least, certain actions (Action phase) may be taken by individuals or city authorities. Feedback is present throughout the whole process and typically assists the Capture phase of the processes. If, for example, the captured data transfer and storage fails, then the participant may be notified to re-transmit or re-capture the target environment.If Processing and Analysis and Visualization result in very little or ambiguous information (e.g., when the processed picture has a bad quality), or participants enter in an area of interest, then they may be notified to (re)capture the situation if possible (Figure 15.1).

An early scenario:

In a modern developed city, people move from place to place in order to commute to home, work, school, and extracurricular activities. They walk, drive, or ride private or public means of transportation to get from one point of the city to another, and their mobility is the perfect solution that can ensure as close to complete sensing coverageas possible. The specific use case that we present here involves bikers moving in the city (e.g. commuting between home and work), carrying their mobile phones on themselves, and possibly several other sensor devices on their bikes (Figure 15.2)

FIGURE 15.2
CycleSense/BikeNet use case.

A modern example:

More recent examples of PS activities focus on exploiting citizen journalist reports from a disaster site in order to produce richer information content and potentially help others who are close to the disaster area. The use case that we describe here consists of three citizen journalists that observe a fire on the corner of Oak and Birch Street near Oak Park (Figure 15.3). At different time instances, each of the participants posts a short message (tweet) with a different description, different intention, and probably spelling and language errors. For example, CityJour#1 names the location of the fire as“Oak Park”CityJour2# as “oak parl,” and the third Tweeter as “corner of Oak and Birch street,” while there are typos or omissions such as “wins” instead of “winds” As the Twitter feeds allow only short messages, there are typically links to more content for the more interested Tweeter followers/readers; for example, a photo taken from the site annotated with the GPS location. The two Tweeters include such links in their tweet with the links redirecting to another site where photos from the fire are stored. In terms of the PS process (Figure 15.4), all or most of the spontaneous tweet by the citizen journalists need to be discovered and collected, which is not an easy task. For identifying the related reports analysis on the actual text, metadata and linked contentat off-site servers may need to take place in order to select the relevant

tweets. Moreover, third party, publicly available data sources such as maps or city authority buildings and addresses are used to enable transformation of location information from one format to another (e.g., from GPS coordinates to road names or building addresses).

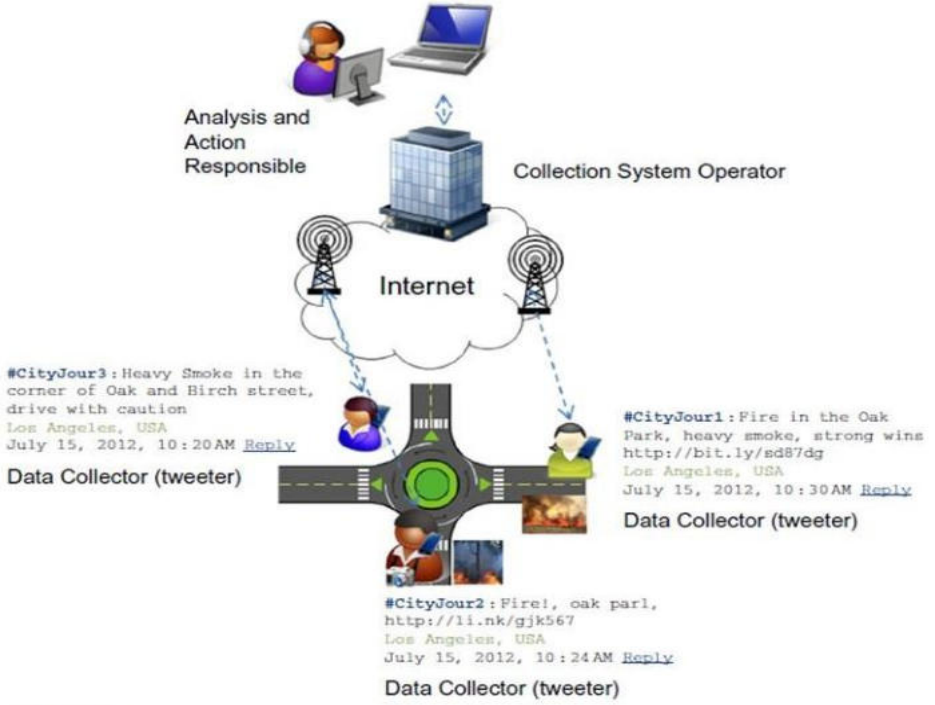


FIGURE 15.3

Citizen Sensing use case.

In addition, weather information provides a potential for prediction of the course of thefire. All these sources of information are subject to correlation and analysis before any visualization and action steps are taken.

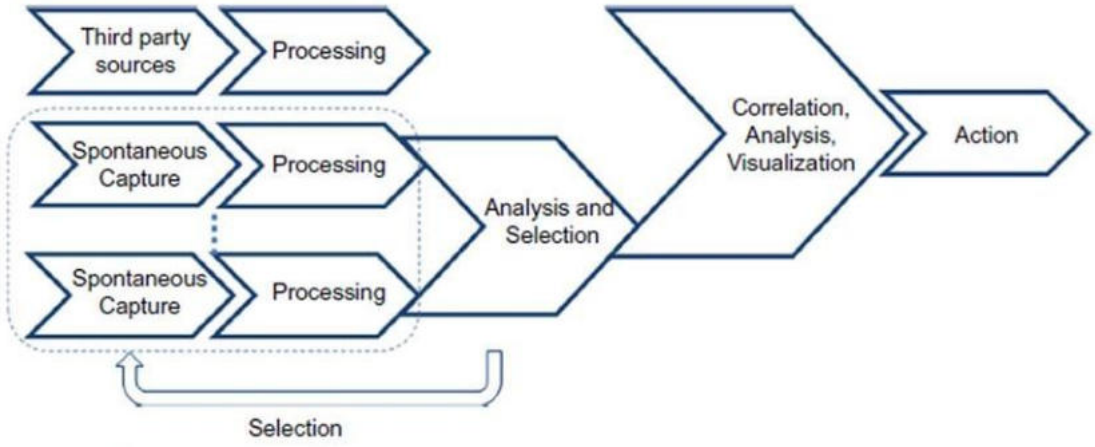


FIGURE 15.4

Citizen Sensing process.

DATA ANALYTICS FOR IOT:

The true importance of IoT data from smart objects is realized only when the analysis of the data leads to actionable business intelligence and insights. Data analysis is typically broken down by the types of results that are produced. As shown in Figure 7- 3, there are four types of data analysis results:

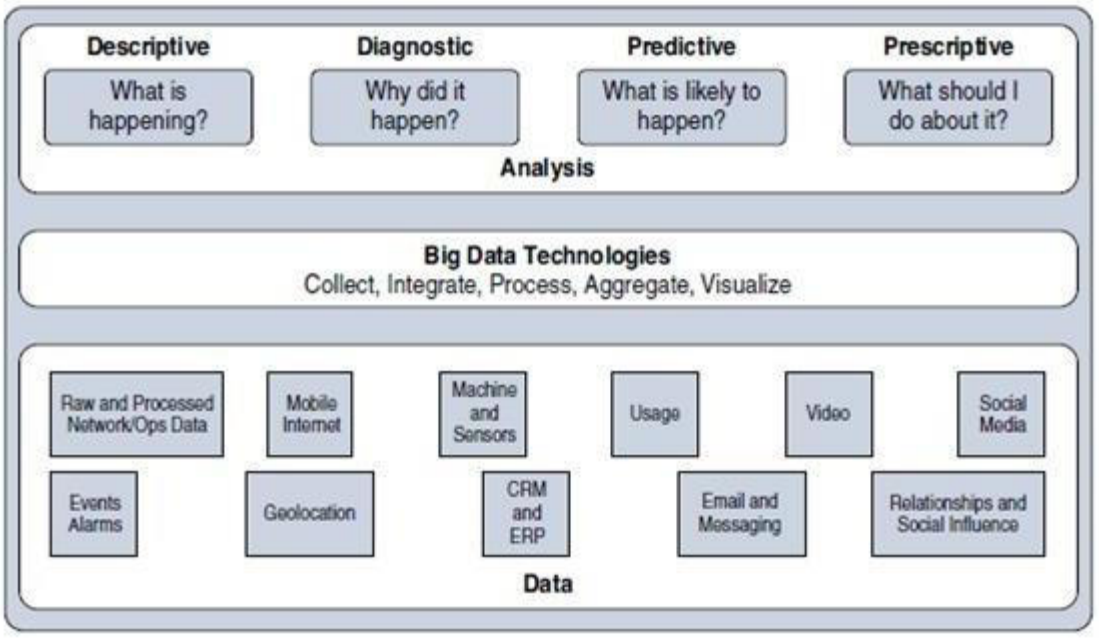


Figure 7-3 *Types of Data Analysis Results*

- **Descriptive:** Descriptive data analysis tells you what is happening, either nowor in the past. For example, a thermometer in a truck engine reports temperature values every second. From a descriptive analysis perspective, you can pull this data at any moment to gain insight into the current operating condition of the truck engine. If the temperature value is too high, then there may be a cooling problem or the engine may be experiencing too much load.
- **Diagnostic:** When you are interested in the "why," diagnostic data analysis can provide the answer. Continuing with the example of the temperature sensor in the truck engine, you might wonder why the truck engine failed. Diagnostic analysis might show that the temperature of the engine was too high,and the engine overheated. Applying diagnostic analysis across the data generated by a wide range of smart objects can provide a clear picture of why a problem or an event occurred.
- **Predictive:** Predictive analysis aims to foretell problems or issues before they occur. For example, with historical values of temperatures for the truck engine, predictive analysis could provide an estimate on the remaining life of certain components in the engine. These components could then be proactively replaced before failure occurs. Or perhaps if temperature values of the truck engine start to rise slowly over time, this could indicate the need for an oil change or some other sort of engine cooling maintenance.
- **Prescriptive:** Prescriptive analysis goes a step beyond predictive andrecommends solutions for upcoming problems. A prescriptive analysis of the temperature data from a truck engine might calculate various alternatives to cost-effectively maintain our truck. These calculations could range from the cost necessary for more frequent oil changes and cooling maintenance to installing new cooling equipment on the engine or upgrading to a lease on a model with a more powerful engine. Prescriptive analysis looks at a variety of factors and makes the appropriate recommendation.

Both predictive and prescriptive analyses are more resource intensive and increase complexity, but the value they provide is much greater than the value from descriptive and diagnostic analysis. Figure 7-4 illustrates the four data analysis types and how they rank as complexity and value increase

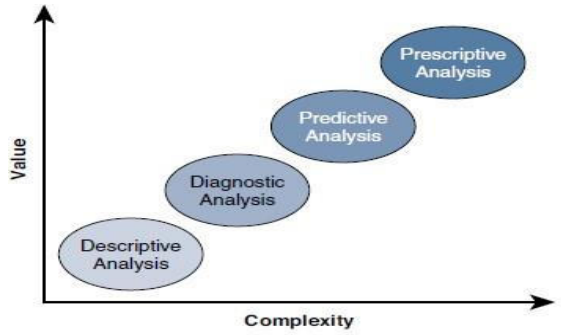


Figure 7-4 *Application of Value and Complexity Factors to the Types of Data Analysis*

IoT Data Analytics Challenges:

As IoT has grown and evolved, it has become clear that traditional dataanalytics solutions were not always adequate. For example, traditional data analytics typically employs a standard RDBMS and corresponding tools, but the world of IoT is much more demanding. While relational databases are still used for certain data types and applications, they often struggle with the nature of IoT data. IoT data places two specific challenges on a relational database:

- Scaling problems: Due to the large number of smart objects in most IoT networks that continually send data, relational databases can grow incredibly large very quickly. This can result in performance issues that can be costly to resolve,often requiring more hardware and architecture changes.
- Volatility of data: With relational databases, it is critical that the schema be designedcorrectly from the beginning. Changing it later can slow or stop the database from operating. Due to the lack of flexibility, revisions to the schema must be kept at a minimum. IoT data, however, is volatile in the sense that the data model is likely to change and evolve over time. A dynamic schema is often required so that data model changes can be made daily or even hourly.

To deal with challenges like scaling and data volatility, a different type of database, known as NoSQL, is being used. Structured Query Language (SQL) is the computer language used to communicate with an RDBMS. As the name implies, a NoSQL database is a database that does not use SQL. It is not set up in the traditional tabular form of are relational database. NoSQL databases do not enforce a strict schema, and they support a complex, evolving data model. These databases are also inherently much more scalable.

In addition to the relational database challenges that IoT imposes, with its highvolume of smart object data that frequently changes, IoT also brings challenges with the lives treaming nature of its data and with managing data at the network level. Streaming data, which is generated as smart objects transmit data, is challenging because it is usually ofa very high volume, and it is valuable only if it is possible to analyse and respond to it in real-time. Real-time analysis of streaming data allows you to detect patterns or anomalies that could indicate a problem or a situation that needs some kind of immediate response. To have a chance of affecting the outcome of this problem, you naturally must be able to filter and analyse the data while it is occurring, as close to the edge as possible.

The market for analysing streaming data in real-time is growing fast. Major cloud analytics providers, such as Google, Microsoft, and IBM, have streaming analytics offerings, and various other applications can be used in house.

Another challenge that IoT brings to analytics is in the area of network data, which is referred to as network analytics. With the large numbers of smart objects in IoT networks that are communicating and streaming data, it can be challenging to ensure that these data flows are effectively managed, monitored, and secure. Network analytics tools such as Flexible NetFlow and IPFIX provide the capability to detect irregular patterns or other problems in the flow of IoT data through a network.

Benefits of IoT Analytics:

- Better visibility and control result in faster decision-making.
- Scalability of business requirements and expansion into other markets.
- Automation results in lower operational costs and greater resource utilization.
- New revenue streams as a result of operational difficulties being resolved.
- Quicker solutions result from accurate problem identification.
- Issues are resolved faster, and they do not reoccur.
- Improved customer experience as a result of purchase history analysis.
- Product development that is both faster and more relevant.

SOFTWARE & MANAGEMENT TOOLS FOR IOT:

IOT SOFTWARE TOOLS:

IoT Software:

The Internet of Things is a vast network of different smart device types able toexchange data autonomously without human interference. An IoT-connected device typically has sensors to collect data, radio or other digital communication technology to send data, and software to control or automate the process. IoT software controls data collection and communication to provide real-time data to computers and applications. Complete IoT solutions, powered with real-time analytics and machine learning, then transform and present data to users as actionable information.

Following is a list of the top IoT software that helps businesses and organizations achieve autonomous data communication, perform data analysis, provide insights, andgive users access to tools to build efficient solutions.

- SAP Internet of Things
- Mule soft Any point
- Particle
- Microsoft Azure IoT Central
- Google Cloud IoT Core
- IBM Watson IoT
- Fracttal One
- AWS IoT
- Boomi
- Altair SmartWorks IoT

SAP Internet of Things:

SAP Internet of Things is a business technology platform that allows users to collect, manage, and embed data from an IoT device into business processes and uses cases such as in product design, manufacturing, supply chain, and operation. It handles a large volume of sensor data that users can easily manage and consume.

SAP's IoT platform can sense identified events within a business context to automatically guide businesses in managing their supply chain processes. The IoT software can run business processes and real-time analytics autonomously on edge devices and allow users to orchestrate a process or analysis report from the cloud.

Pros:

- Simple and fast IoT development
- Easy device integration
- Responsive customer support

Cons:

- Dashboard needs improvement
- Higher license cost compared with other IoT solutions

MuleSoft Anypoint:

MuleSoft Anypoint Platform is an enterprise hybrid integration platform that lets usersconnect applications, data, and devices. The integrated solution accelerates applicationdevelopment by letting users build APIs, use any existing asset, or discover APIs builtby others. From the same platform, users can test APIs and integrations, deploy in an IoT architecture, ensure security, and manage APIS and services comprehensively.

MuleSoft Anypoint lets companies integrate IoT devices using APIs and a gateway that supports various transport protocols and ready-to-use connectors. Users can transform any device like vending machines and billboards into smart devices with its runtime engine. Layers of RESTful APIs provide an interface for IoT software development and interaction with new IoT solutions.

Pros:

- Ease of setup and use
- Comprehensive documentation
- All-in-one solution

Cons:

- Pricing is on the high end
- Older versions lack support

Particle:

Particle is an integrated IoT Platform-as-a-Service (PaaS) that provides tools to help users prototype, scale, and manage IoT products. It is a full-stack IoT system that lets users build connected products to help businesses grow and optimize their operations. It provides IoT software, connectivity, and hardware and makes all the elements work together in an integrated PaaS.

Particle tools are reprogrammable and reconfigurable to allow users to develop custom IoT solutions for specific use cases. Some of the industries where Particle helps customers grow and gain advantages are in light electric vehicles, smart energy, HVAC, emissions monitoring, and industrial equipment monitoring.

Pros:

- Easy setup and use
- Cost-effective IoT solution
- End-to-end tools from app development to deployment

Cons:

- Users reports occasional glitches
- No longer supporting mesh networking solution

Microsoft Azure IoT Central:

Microsoft Azure provides numerous cloud computing services including an IoT platform. Users can create and develop industry-specific cloud solutions using edge- to-cloud technology with built-in IoT security, privacy, and compliance. Azure IoT offers platform services for companies with cloud solutions and device expertise.

Microsoft also offers a managed app platform through Azure IoT Central so companies can quickly build IoT apps with a fully managed solution. It is highly secure, scales as the business grows, and integrates with existing business apps. It provides a ready-made web experience and API surface that simplify IoT development services. Tools include a job scheduler, data storage, custom dashboards,rules, data export, and a public REST API.

Pros:

- Easy device management
- All tools in one place
- Responsive customer support

Cons:

- High learning curve
- Users wish for more how-to videos

Google Cloud IoT Core:

Google Cloud IoT Core is a managed service for connecting, managing, and ingesting data from dispersed devices in an easy and secure way. In combination with other cloud services, users get a complete solution to collect, process, analyse, and visualize IoT data in real time so operations can be more efficient.

Google Cloud IoT Core can aggregate dispersed device data and integrate it with data analytics services. IoT data stream can be further analysed and visualized using machine learning to improve operations, anticipate problems, and build new business models. The platform supports standard MQTT and HTTP protocols, 2-way communication with devices, and central management that works with leading hardware manufacturers.

Pros:

- Ease of setup
- Support for MQTT and HTTP protocols
- Responsive customer support

Cons:

- High learning curve
- Planned retirement of the software on August 2023

MANAGEMENT TOOLS FOR IOT:

IoT Management Platforms enable businesses to manage multiple connected devices remotely over the network. They entail provisioning and authentication of IoT devicesfor better security and to keep the attackers away from the smart devices. They also provide advanced features to manage the entire network of devices.

Key Features:

- Remote troubleshooting
- Error Handling
- Patch Security Holes
- Monitoring IoT devices
- Configuration of IoT devices
- Device Health Diagnosis
- Monitors usage and Performance Metrics

To eliminate the threat and easy management of IoT fleets, it needs update Management Tools that can helps to accelerate the performance. A few top IoT Update Management platforms on the basis of their features and functionalities are listed below.

Syxsense: is a cloud-based IoT update management platform designed for managed service providers (MSPs) and IT professionals to prevent threats and scan vulnerabilities. It comprises various advanced features, including third-party patching, team collaboration, Network Device Map solutions, Two-Factor Authentication, and more to protect IoT devices at all times.

Further, their products and tools enable users to keep their desktops, laptops, and servers up-to-date with the latest security features.

Syxsense is ideal for both SMB and large enterprises giving an overview of networks to analyze, predict and detect threats in real-time. Using the dashboard, users can viewand ensure all endpoints are secure within your environment. It also provides the location of your devices on a Network map and Alert Monitors.

Key Features:

- Centralized IoT device management solution for tracking devices
- Network Device Map solutions
- Remote Desktop Access
- Provides endpoint security
- Software Distribution
- Remote Control
- Automated IoT Management
- Allows users to view anything new added to the networks in the console
- Alerts and notifications
- Two-Factor Authentication
- Audit Management
- Asset Discovery
- Compliance Tracking
- Configuration Management
- Approval Process Control
- Inventory History
- Server Patch Management
- Third-Party Patch Management
- Manages risk exposure

- Effective patch management solution
- Compatible with Windows, Mac, and Linux
- Supports security patches, software updates, and feature upgrades to keep usersup-to-date at all times
- Vulnerability Scanner
- Patching for Legacy OS
- Patching for Virtual machines

JFrog Connect: is a leading tool that helps control, monitor, and secure all IoT devices with a single click. Connecting with any type of Linux or IoT device is much easier and quick with JFrog Connect. It is an all-in-one device management platform with various advanced features that make it a top choice.

Users can access different remote product fleet versions and locations using the tool. Further, they comprise high-end deployment capabilities that help keep the software solution updated at all times for IoT and edge Linux devices.

Get Real-time monitoring of your connected devices, data, and application with JFrog Connect and save your devices from unwanted threats and cybersecurity challenges. Today, most enterprises and startups projects use JFrog Connect to help eliminate product recalls and software bugs, further adding more visibility to your IoT devicesat scale.

Key Features

- Monitors all IoT devices at scale with a click
- The tool hardly takes any time to connect with IoT devices
- Agile and easy-to-use IoT update management solution
- Supports high-end deployment capabilities
- Allows tools to control and manage devices from any location
- Supports port forwarding, SSH, VNC, and remote Bash commands
- Offers real-time monitoring
- Anomaly detection engine
- Provides high visibility
- Reduces product recalls and software bugs
- Highly scalable and flexible IoT solution
- Allows users to access networks remotely
- Process Monitor and Device Management
- Log Management
- Remote control and commands
- Fully integrated into software update process and DevOps tools
- Provides end-to-end automation

DIGI Remote Manager: helps businesses in IoT device monitoring and network management. Its advanced features and solution support integration with network infrastructure to automate processes. Further, DIGI Remote Manager helps keep a track record of all activities across the network and adds commands over groups and individually.

With the help of Digi Remote Manager® (Digi RM), users can easily activate, monitor, and diagnose multiple devices using a single point of command. Further, youhave full access to edit configurations and update the firmware.

It is one of the best tools that provides automatic monitoring and remediation to the network's device security. DIGI Remote Manager provides instant updates if any issue arises by sending alerts and delivering reports to monitor and fix the issues.

Key Features

- Manages Remote Assets Securely
- Rapid device deployment
- IoT device monitoring and checking asset performance
- Track security with bi-directional communications
- Automates mass firmware
- Provides software updates to scale deployment and stay in compliance
- Use open APIs for better insights
- Configuration Management
- Integrate device data and control third-party applications
- Real-time alerts
- Offers detailed reports on network health
- View and group device inventory
- Allows users to update device settings in bulk and load configuration files
- Provides installation features
- Real-time device location
- Uses the File Management page to load files
- Supports SM/UDP (Short Message/User Datagram Protocol) features
- Allows storing time-series data

AWS IoT Device Management: allows you to deploy, monitor, and manage thousands of connected devices. It helps businesses detect and remotely troubleshoot problems, provide regular software and firmware updates. It also helps in organizing devices and manages access policies for the created groups.

Another benefit of using AWS IoT Device Management is it helps find and discover IoT devices using a combination of attributes like ID, device state, type across the entire device fleet in real-time. It comprises various advanced features like AWS IoT Device Defender that help users to conduct audits, send timely alerts, monitor connected device fleets, and take mitigating actions.

AWS IoT Device Management also enables users to monitor usage and performance metrics.

Key Features

- Allows quick registration of connected devices in bulk
- Allows grouping of device fleet based on functions and security requirements
- Allows users to view operational metrics and manage access policies across thegroup
- Allows querying a group of devices and aggregate statistics
- Advanced Search Fleet Indexing
- Users can collect device logs and configure them to identify issues quickly
- Remotely push software and firmware updates
- Troubleshoot issues on devices
- control deployment velocity
- set failure thresholds
- FreeRTOS over-the-air (OTA) update job
- Secure Tunneling
- Fleet Hub feature helps visualize and interact with device fleet
- IoT Core

- IoT Device Defender for auditing, monitoring, and sending alerts

CLOUD STORAGE MODELS & COMMUNICATION APIS: INTRODUCTION TO CLOUD STORAGE MODELS:

In truth, cloud computing and IoT are tightly coupled. The growth of IoT and the rapid development of associated technologies create a widespread connection of things. This has lead to the production of large amounts of data, which needs to be stored, processed and accessed. Cloud computing as a paradigm for big data storage and analytics. While IoT is exciting on its own, the real innovation will come from combining it with cloud computing. The combination of cloud computing and IoT willenable new monitoring services and powerful processing of sensory data streams.

For example, sensory data can be uploaded and stored with cloud computing, later to be used intelligently for smart monitoring and actuation with other smart devices. Ultimately, the goal is to be able to transform data to insight and drive productive, cost-effective action from those insights. The cloud effectively serves as the brain to improved decision-making and optimized internet-based interactions. However, when IoT meets cloud, new challenges arise. There is an urgent need for novel network architectures that seamlessly integrate them. The critical concerns during integration are quality of service (QoS) and quality of experience (QoE), as well as data security, privacy and reliability.

Popular Models are

- Amazon Web Service (AWS)
- XivelyCloud (PaaS)

COMMUNICATION APIS:

Three basic forms of APIs are local, web-like and program-like.

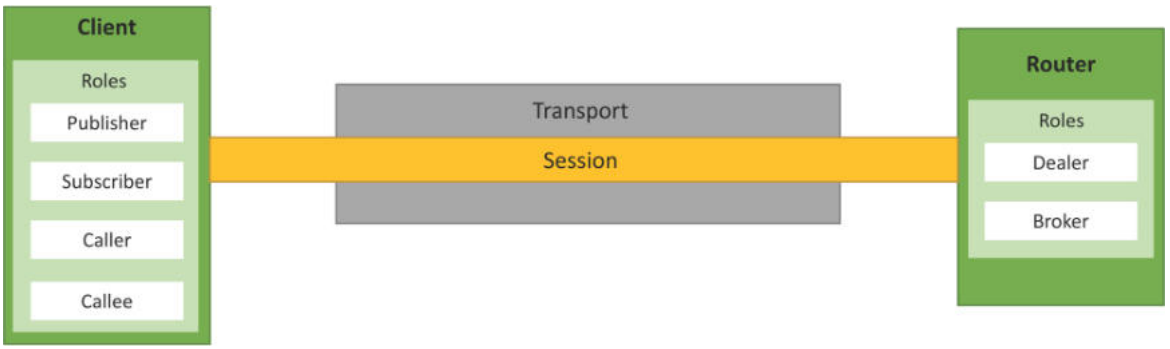
1. **Local APIs:** is the original form, from which the name came. They offer OS or middleware services to application programs. Microsoft's .NET APIs, the TAPI (Telephony API) for voice applications, and database access APIs are examples of the local API form.
2. **Web APIs** are designed to represent widely used resources like HTML pages and are accessed using a simple HTTP protocol. Any web URL activates a web API. Web APIs are often called REST (representational state transfer) or RESTful because the publisher of REST interfaces doesn't save any data internally between requests. As such, requests from many users can be intermingled as they would be on the internet.
3. **Program APIs** are based on remote procedure call (RPC) technology that makes a remote program component appear to be local to the rest of the software. Service oriented architecture (SOA) APIs, such as Microsoft's WS-series of APIs are programming APIs.
    - Cloud Models are relied on Communication API.
    - Communications API facilitate data transfer, control information transfer fromapplication to cloud, one service to another.
    - It also exists in the form of Communication Protocols.
    - It supports RPC (Remote Procedure Call), PUBSUB (publish/subscribe protocols), and WAMP (Web Application Messaging Protocol).

WAMP – AutoBahn for IoT:

WAMP for IoT:

- Mainly used in cloud storage model for IoT& other messaging services
- WAMP is a routed protocol, with all components connecting to aWAMP Router, where the WAMP Router performs message routing between the component
- It is protocol for Web Socket (PUBSUB based protocol) : uses RPC MessagingPattern
- Some Important Key Terminologies are
    - Transport
    - Session
    - Clients (Publisher & Subscriber)
    - Router
    - Broker
    - Dealer
    - Application Code

WAMP is a sub-protocol of WebSocket which provides publish–subscribe and remoteprocedure call (RPC) messaging patterns.



WAMP –Concepts:

- **Transport:** Transport is a channel that connects two peers.
- **Session:** Session is a conversation between two peers that runs over a transport.
- **Client:** Clients are peers that can have one or more roles.
    - In the publish–subscribe model, the Client can have the following roles:
        - Publisher: Publisher publishes events (including payload) to the topicmaintained by the Broker.
        - Subscriber: Subscriber subscribes to the topics and receives theevents including the payload.
    - In the RPC model, the Client can have the following roles:
        - Caller: Caller issues calls to the remote procedures along with callarguments.
        - Callee: Callee executes the procedures to which the calls are issuedby the Caller and returns the results to the Caller.
            - **Router:** Routers are peers that perform generic call and event routing**.**
                - In the publish–subscribe model, the Router has the role of a Broker.
        - Broker: Broker acts as a Router and routes messages published to atopic to all the subscribers subscribed to the topic.
    - In the RPC model, the Router has the role of a Dealer.

- Dealer: Dealer acts a router and routes RPC calls from the Caller tothe Callee and routes results from the Callee to the Caller.

Application code: Application code runs on the Clients (Publisher, Subscriber,Calleeor Caller).

CLOUD FOR IoT:

Xively Cloud for IoT:

Use of Cloud IoT cloud-based service

- The service provides for the data collection, data points, messages andcalculation objects.
- The service also provisions for the generation and communication of alerts,triggers and feeds to the user.
- A user is an application or service. The user obtains responses or feeds fromthe cloud service.

Pachube platform: for data capture in real-time over the Internet

- Cosm: a changed domain name, where using a concept of console, one canmonitor the feeds
- Xively is the latest domain name.A commercial PaaS for the IoT/M2M
  - A data aggregator and data mining website often integrated into the Web ofThings
  - An IoT PaaS for services and business services.

Xively PaaS services:

- Data visualisation for data of connected sensors to IoT devices.
- Graphical plots of collected data.
- Generates alerts.
- Access to historical data
- Generates feeds which can be real-world objects of own or others

Xively HTTP based APIs:

- Easy to implement on device hardware acting as clients to Xively web services
- APIs connect to the web service and send data.
- APIs provides services for logging, sharing and displaying sensor data of all.

Xively Support:

- The platform supports the REST, WebSockets and MQTT protocols andconnects the devices to Xively Cloud Services
- Native SDKs for Android, Arduino, ARM mbed, Java, PHP, Ruby, and Pythonlanguages
- Developers can use the workflow of prototyping, deployment and managementthrough the tools provided at Xively

Xively APIs:

- Enable interface with Python, HTML5, HTML5 server, tornado
- Interface with WebSocket Server and WebSockets
- Interface with an RPC (Remote Procedure Call).

Xively PaaS services:

- Enables services

- Business services platform which connects the products, including collaboration products
- Rescue, Boldchat, join.me, and operations to Internet
- Data collection in real-time over Internet

Xively Methods for IoT Devices Data:

- Concept of users, feeds, data streams, data points and triggers
- Data feed typically a single location (e.g. a device or devices network),
- Data streams are of individual sensors associated with that location (forexample, ambient lights, temperatures, power consumption).
- Pull or Push (Automatic or Manual Feed)

Xively Data formats and Structures:

- Number of data formats and structures enable the interaction, data collectionand services
- Support exists for JSON , XML and CSV
- Structures: Tabular, spreadsheet, Excel, Data numbers and Text with a comma-separated values in file

Xively Uses in IoT/M2M

- Private and Public Data Access
- Data streams, Data points and Triggers
- Creating and Managing Feeds
- Visualising Data

AMAZON WEB SERVICES FOR IoT:

i) Amazon EC2:

Boto is a Python package that provides interfaces to Amazon Web Services (AWS).

- In this example, a connection to EC2 service is first established by calling boto.ec2.connect_to_region.
- The EC2 region, AWS access key and AWS secret key are passed to this function. After connecting to EC2, a new instance is launched using the conn.run_instances function.
- The AMI-ID, instance type, EC2 key handle and security group are passed to this function.

```
#Python program for launching an EC2 instance
import boto.ec2
from time import sleep
ACCESS_KEY="<enter access key>"
SECRET_KEY="<enter secret key>"

REGION="us-east-1"
AMI_ID = "ami-d0f89fb9"
EC2_KEY_HANDLE = "<enter key handle>"
INSTANCE_TYPE="t1.micro"
SECGROUP_HANDLE="default"

conn = boto.ec2.connect_to_region(REGION, aws_access_key_id=ACCESS_KEY,
                                  aws_secret_access_key=SECRET_KEY)

reservation = conn.run_instances(image_id=AMI_ID, key_name=EC2_KEY_HANDLE,
                                 instance_type=INSTANCE_TYPE,
                                 security_groups = [ SECGROUP_HANDLE, ] )
```

ii) Amazon AutoScaling:

- **AutoScaling Service:**

A connection to AutoScaling service is first established by calling boto.ec2.autoscale.connect_to_region function.

- **Launch Configuration**

After connecting to Auto Scaling service, a new launch configuration is created by calling conn.create_launch_con figuration. Launch configuration contains instructions on how to launch new instances including the AMI-ID, instance type, security groups, etc.

- **AutoScaling Group**

After creating a launch configuration, it is then associated with a new AutoScaling group. AutoScaling group is created by calling conn.create_auto_scaling_group. The

settings for AutoScaling group such as the maximum and minimum number of instances in the group, the launch configuration, availability zones, optional load balancer to use with the group, etc.

```
#Python program for creating an AutoScaling group (code excerpt)
import boto.ec2.autoscale
:
print "Connecting to Autoscaling Service"
conn = boto.ec2.autoscale.connect_to_region(REGION,
  aws_access_key_id=ACCESS_KEY,
  aws_secret_access_key=SECRET_KEY)

print "Creating launch configuration"

lc = LaunchConfiguration(name='My-Launch-Config-2',
                                                image_id=AMI_ID,

                key_name=EC2_KEY_HANDLE,

                instance_type=INSTANCE_TYPE,
                                                security_groups = [
SECGROUP_HANDLE, ])
conn.create_launch_configuration(lc)

print "Creating auto-scaling group"

ag = AutoScalingGroup(group_name='My-Group',
              availability_zones=['us-east-1b'],
              launch_config=lc, min_size=1, max_size=2,
              connection=conn)
conn.create_auto_scaling_group(ag)
```

- **AutoScaling Policies**

After creating an AutoScaling group, the policies for scaling up and scalingdown are defined.

In this example, a scale up policy with adjustment type Change In Capacity andscaling_ad justment = 1 is defined.

Similarly a scale down policy with adjustment type ChangeInCapacity andscaling_ad justment = -1 is defined.

```
#Creating AutoScaling policies
scale_up_policy = ScalingPolicy(name='scale_up',
                adjustment_type='ChangeInCapacity',
                as_name='My-Group',
                scaling_adjustment=1,
                cooldown=180)
scale_down_policy = ScalingPolicy(name='scale_down',
                adjustment_type='ChangeInCapacity',
                                                as_name='My-Group',
                scaling_adjustment=-1,
                cooldown=180)

conn.create_scaling_policy(scale_up_policy)
conn.create_scaling_policy(scale_down_policy)
```

- **CloudWatch Alarms**

With the scaling policies defined, the next step is to create Amazon CloudWatch alarms that trigger these policies.

The scale up alarm is defined using the CPUUtilization metric with the Average statistic and threshold greater 70% for a period of 60 sec. The scale uppolicy created previously is associated with this alarm. This alarm is triggered when the average CPU utilization of the instances in the group becomes greaterthan 70% for more than 60 seconds.

The scale down alarm is defined in a similar manner with a threshold less than 50%.

```
#Connecting to CloudWatch
cloudwatch = boto.ec2.cloudwatch.connect_to_region(REGION,
                                    aws_access_key_id=ACCESS_KEY,
                                    aws_secret_access_key=SECRET_KEY)
alarm_dimensions = {"AutoScalingGroupName": 'My-Group'}

#Creating scale-up alarm
scale_up_alarm = MetricAlarm(
        name='scale_up_on_cpu', namespace='AWS/EC2',
        metric='CPUUtilization', statistic='Average',
        comparison='>', threshold='70',
        period='60', evaluation_periods=2,
        alarm_actions=[scale_up_policy.policy_arn],
        dimensions=alarm_dimensions)
cloudwatch.create_alarm(scale_up_alarm)

#Creating scale-down alarm
scale_down_alarm = MetricAlarm(
        name='scale_down_on_cpu', namespace='AWS/EC2',
        metric='CPUUtilization', statistic='Average',
        comparison='<', threshold='40',
        period='60', evaluation_periods=2,
        alarm_actions=[scale_down_policy.policy_arn],
        dimensions=alarm_dimensions)
cloudwatch.create_alarm(scale_down_alarm)
```

iii) Amazon S3:

In this example, a connection to S3 service is first established by callingboto.connect_s3 function.

The upload_to_s3_bucket_path function uploads the file to the S3 bucketspecified at the specified path.

```
# Python program for uploading a file to an S3 bucket
import boto.s3

conn = boto.connect_s3(aws_access_key_id='<enter>',
    aws_secret_access_key='<enter>')

def percent_cb(complete, total):
   print ('.')

def upload_to_s3_bucket_path(bucketname, path, filename):
                mybucket = conn.get_bucket(bucketname)
                fullkeyname=os.path.join(path,filename)
                key = mybucket.new_key(fullkeyname)
                key.set_contents_from_filename(filename, cb=percent_cb,
num_cb=10)
```

iv) Amazon RDS

In this example, a connection to RDS service is first established by calling boto.rds.connect_to_region function.

- The RDS region, AWS access key and AWS secret key are passed to this function.
- After connecting to RDS service, the conn.create_dbinstance function is called to launch a new RDS instance.
- The input parameters to this function include the instance ID, databasesize, instance type, database username, database password, database port, database engine (e.g. MySQL5.1), database name, security groups, etc.

```
#Python program for launching an RDS instance (excerpt)
import boto.rds

ACCESS_KEY="<enter>"
SECRET_KEY="<enter>"
REGION="us-east-1"
INSTANCE_TYPE="db.t1.micro"
ID = "MySQL-db-instance-3"
USERNAME = 'root'
PASSWORD = 'password'
DB_PORT = 3306
DB_SIZE = 5
DB_ENGINE = 'MySQL5.1'
DB_NAME = 'mytestdb'
SECGROUP_HANDLE="default"

#Connecting to RDS
conn =  boto.rds.connect_to_region(REGION,
  aws_access_key_id=ACCESS_KEY,
  aws_secret_access_key=SECRET_KEY)


#Creating an RDS instance
db = conn.create_dbinstance(ID, DB_SIZE, INSTANCE_TYPE,
USERNAME, PASSWORD,  port=DB_PORT, engine=DB_ENGINE,
db_name=DB_NAME, security_groups = [ SECGROUP_HANDLE, ] )
```

v) Amazon Dynamo DB

In this example, a connection to DynamoDB service is first established bycalling boto.dynamodb.connect_to_region.

- After connecting to DynamoDB service, a schema for the new table iscreated by calling conn.create_schema.
- The schema includes the hash key and range key names and types.
- A DynamoDB table is then created by calling conn.create_table functionwith the table schema, read units and write units as input parameters.

```
# Python program for creating a DynamoDB table (excerpt)
import boto.dynamodb

ACCESS_KEY="<enter>"
SECRET_KEY="<enter>"
REGION="us-east-1"

#Connecting to DynamoDB
conn = boto.dynamodb.connect_to_region(REGION,
  aws_access_key_id=ACCESS_KEY,
  aws_secret_access_key=SECRET_KEY)

table_schema = conn.create_schema(
      hash_key_name='msgid',
      hash_key_proto_value=str,
      range_key_name='date',
      range_key_proto_value=str
   )

#Creating table with schema
table = conn.create_table(
      name='my-test-table',
      schema=table_schema,
      read_units=1,
      write_units=1
   )
```