OPEN Elective Course (OEC-4)

SEM: VIII	IoT IN AUTOMATION	L	T	P	C
BRANCH: EIE		3			3
CODE:		Category: OEC			

PRE-REQUISITE **(45 hrs)**

1. Digital Electronics.

COURSE OBJECTIVES

The course will understand the concepts of IoT:

1. Introduction to IoT
2. IoT and Machine to Machine.
3. Network and Communication aspects
4. Challenges and Applications of IoT.
5. Developing IoT using Python Programming.

UNIT 1. INTRODUCTION TO IoT **(9hrs)**

Defining IoT – Characteristics of IoT, Physical design of IoT, Logical design of IoT, Functional blocks of IoT, Communication models and APIs.

UNIT 2. IoT & M2M **(9hrs)**

Machine to Machine, Difference between IoT and M2M, Software define Network Iot Design Methodology,

UNIT 3. NETWORK AND COMMUNICATION **(9hrs)**

Wireless medium access issues, MAC protocol survey, Survey routing protocols, Sensor deployment and Node discovery, Data aggregation and dissemination

UNIT 4. CHALLENGES AND APPLICATIONS **(9hrs)**

Design challenges, Development challenges, Security challenges, Other Challenges, Home Automation, Cities and Environment and Health Monitoring.

UNIT 5. DEVELOPING IoTS **(9hrs)**

Introduction to Python, Introduction to different IoT tools, developing applications through IoT tools, developing sensor based application through embedded system platform, Implementing IoT concepts with Python

COURSE OUTCOMES

The Students should be able to:

CO1. Understand the concepts of Internet of Things.
CO2. Analyze basic protocols in wireless sensor network.
CO3. Design IoT applications in different domain and be able to analyze their performance
CO4. Implement basic IoT applications on embedded platform.

TEXTBOOK

1. Vijay Madisetti, Arshdeeo Bahga, "Internet of Things: A Hands-On Approach"
2. Waltenegus Dargie, Christian Poellabauer, "Fundamentals of Wireless Sensor Networks: Theory and Practice"

REFERENCES

1. Francis daCosta, "Rethinking the Internet of Things: A Scalable Approach to Connecting Everything", 1st Edition, A press Publications, 2013

Mapping of COs with POs												
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	✓	✓										
CO2	✓			✓					✓	✓	✓	
CO3		✓									✓	✓
CO4	✓				✓					✓		
CO5		✓									✓	✓

UNIT-I
INTRODUCTION To 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:

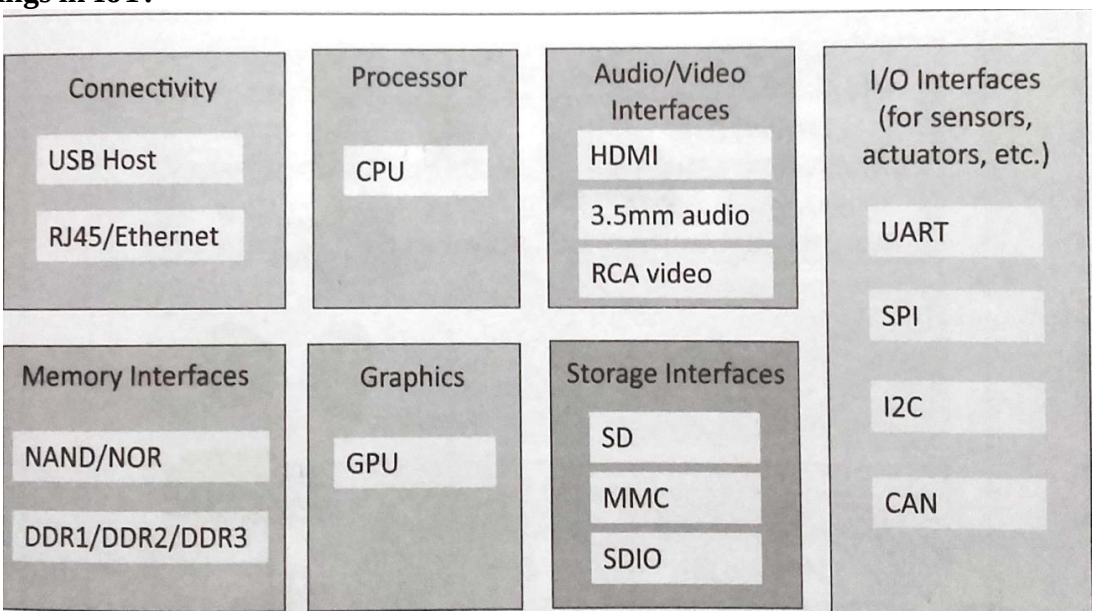
1) **Dynamic & Self Adapting**: IoT devices and systems may have the capability to dynamically adapt with the changing contexts and take actions based on their operating conditions, user‗s context or sensed environment.
   Eg: the surveillance system is adapting itself based on context and changing conditions.
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.

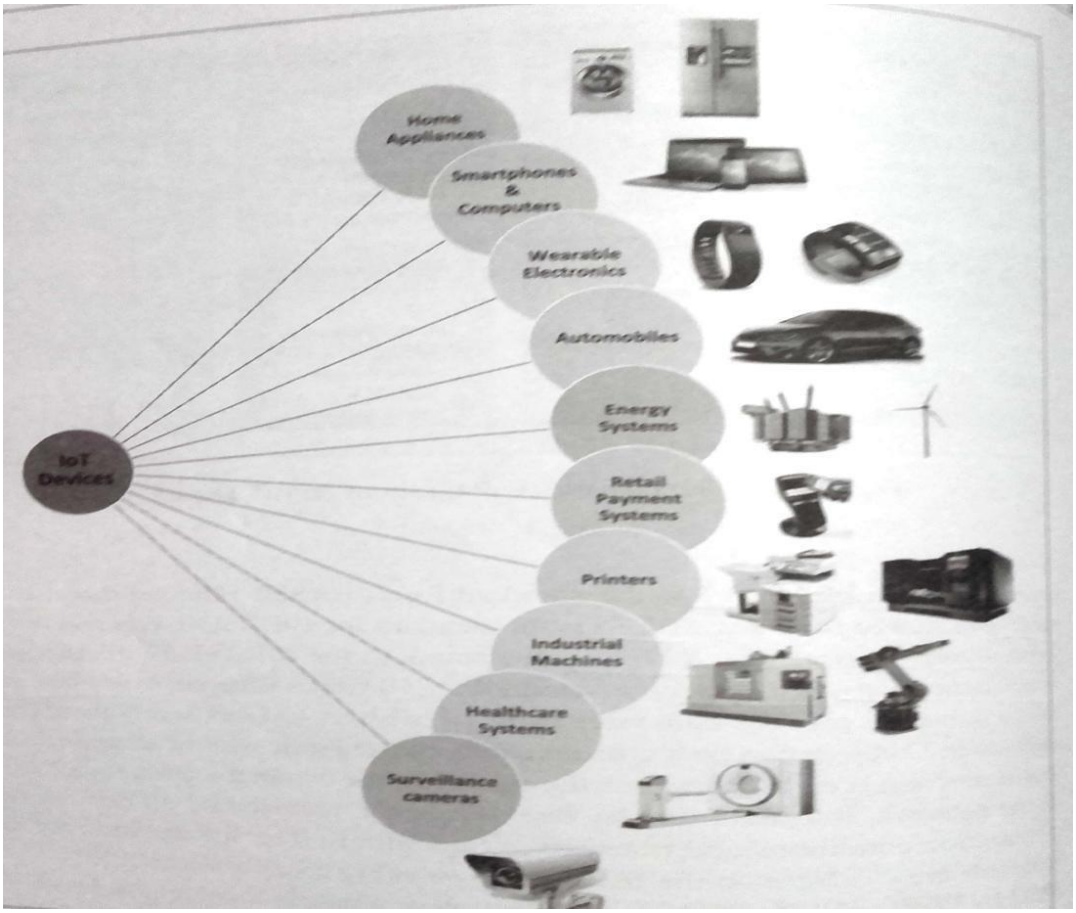
Applications of IoT:

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:**

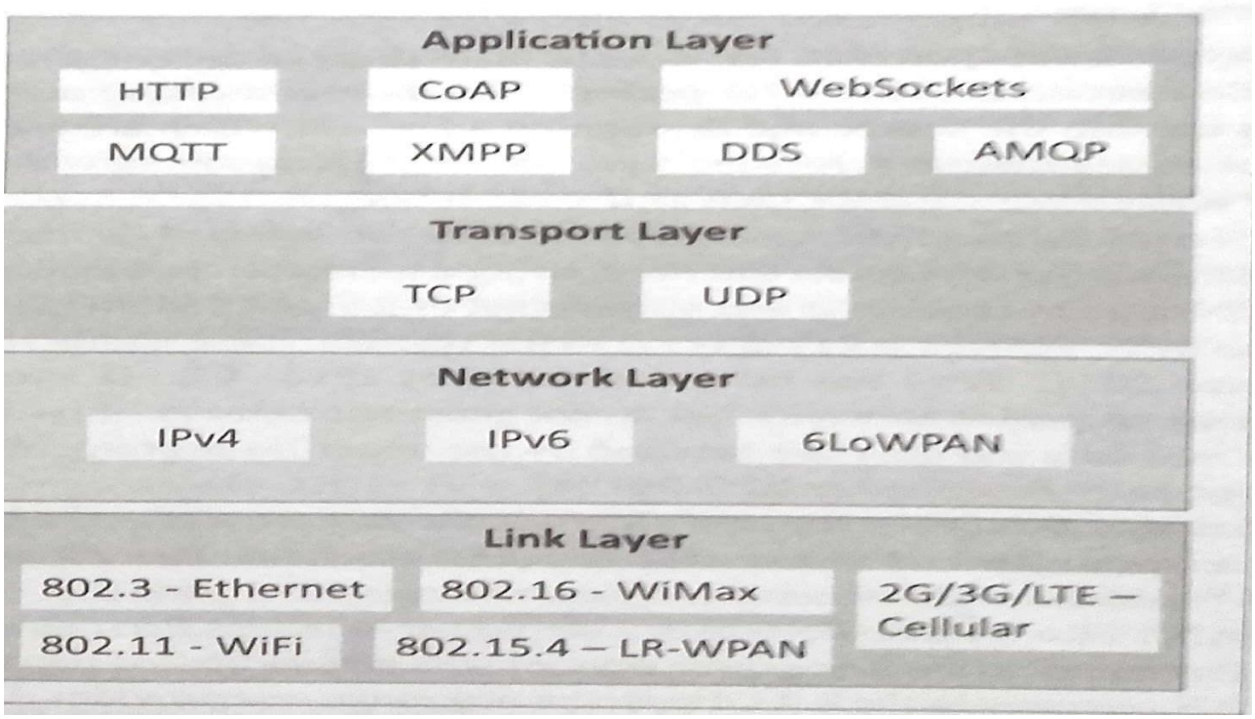




The things in IoT refers to IoT devices which have unique identities and perform remote sensing, actuating and monitoring capabilities. IoT devices can exchange data with other connected devices applications. It collects data from other devices and process data either locally or remotely. An IoT device 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:

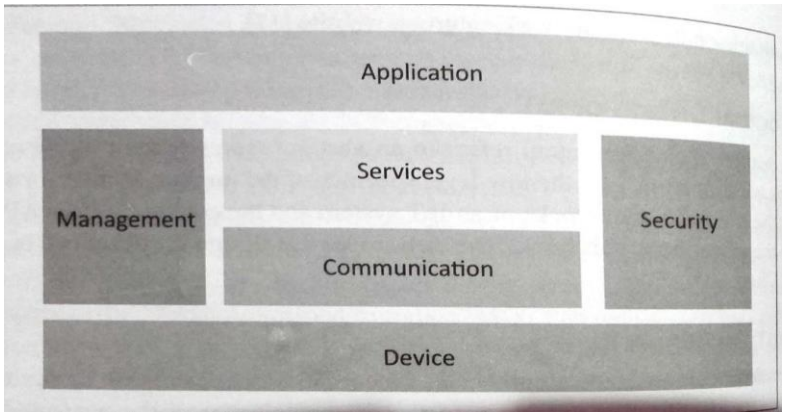
- **HTTP:** Hyper Text Transfer Protocol that forms foundation of WWW. Follow request-response model Stateless protocol.
- **CoAP:** Constrained Application Protocol for machine-to-machine (M2M) applications with constrained devices, constrained environment and constrained n/w. Uses client- server architecture.
- **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

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



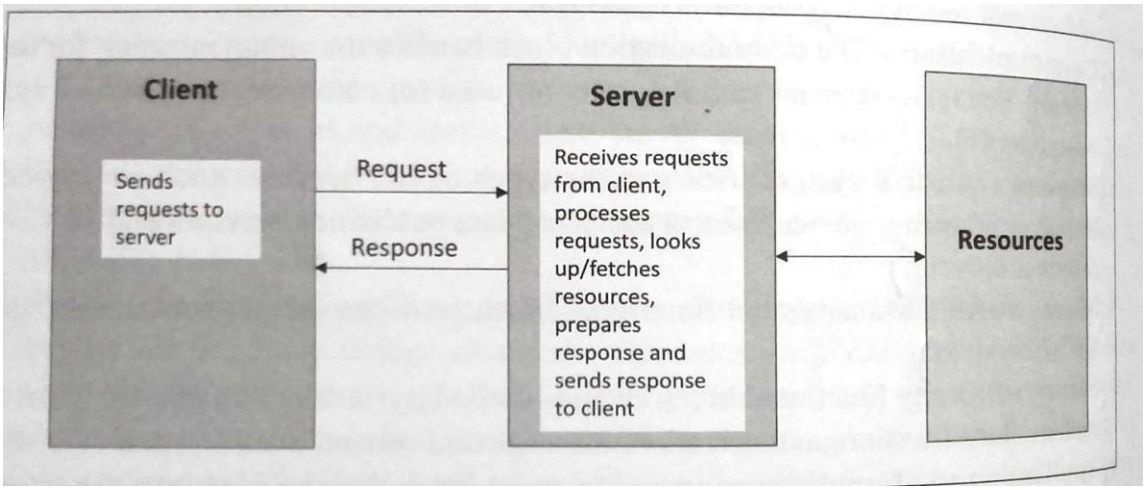
- **Device:** An IoT system comprises of devices that provide sensing, actuation, monitoring and control functions.
- **Communication:** handles the communication for IoTsystem.
- **Services:** for device monitoring, device control services, data publishing services and services for device discovery.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

2) IoT Communication Models:

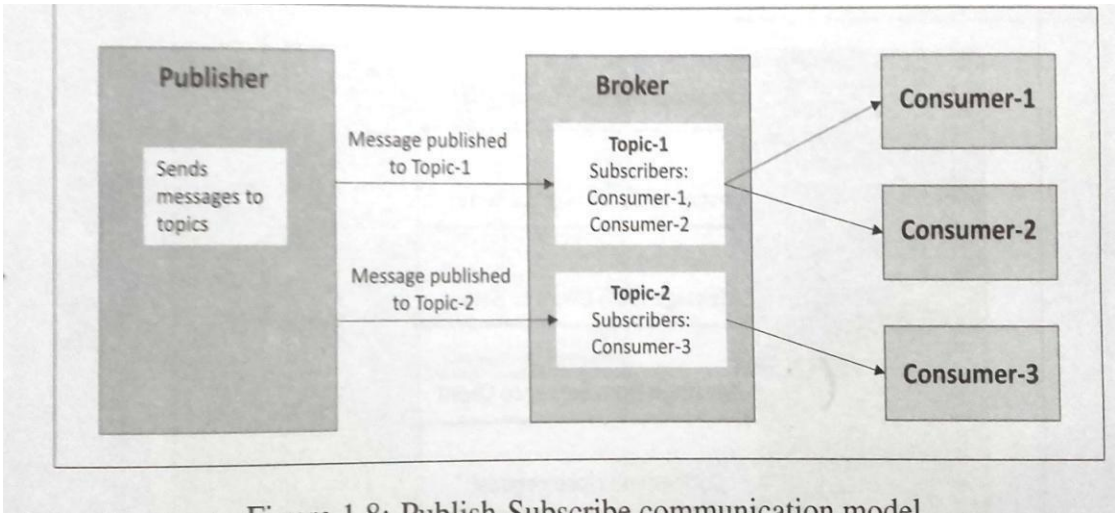
1) Request-Response 2) Publish-Subscibe 3)Push-Pull4) ExclusivePair

1) Request-Response Model:



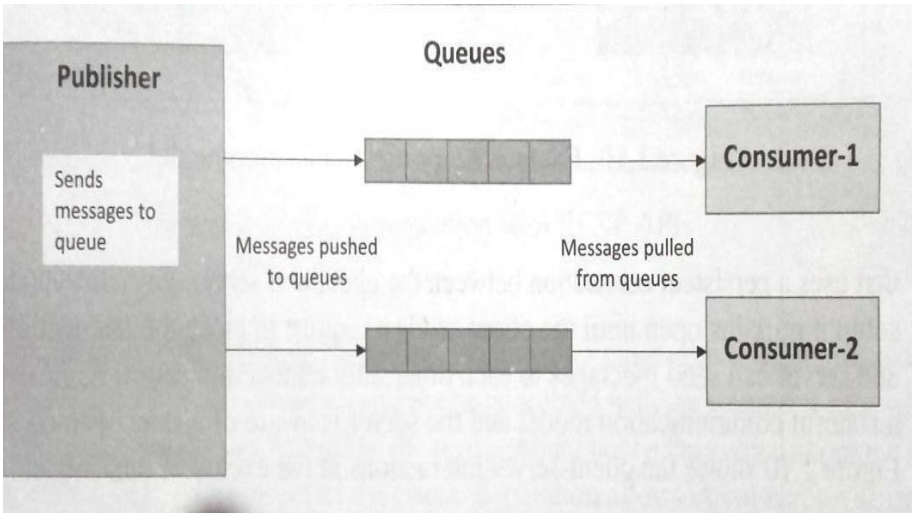
In which the client sends request to the server and the server replies to requests. Is a stateless communication model and each request-response pair is independent of others.

2) Publish-Subscibe Model:

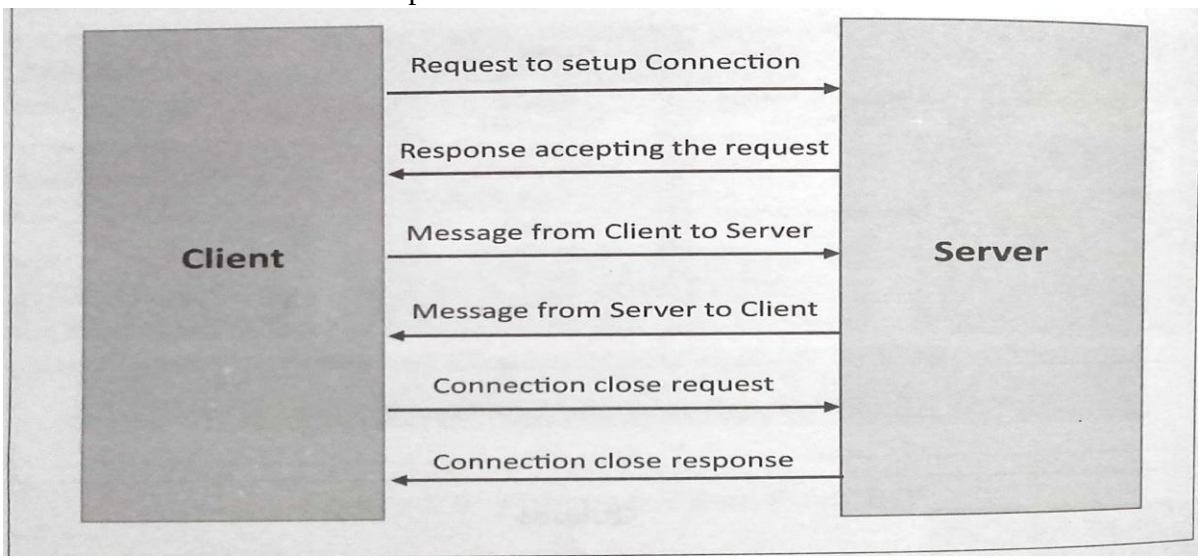


Involves publishers, brokers and consumers. Publishers are source of data. Publishers send data to the topics which are managed by the broker. Publishers are not aware of the consumers. Consumers subscribe to the topics which are managed by the broker. When the broker receives data for a topic from the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



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.



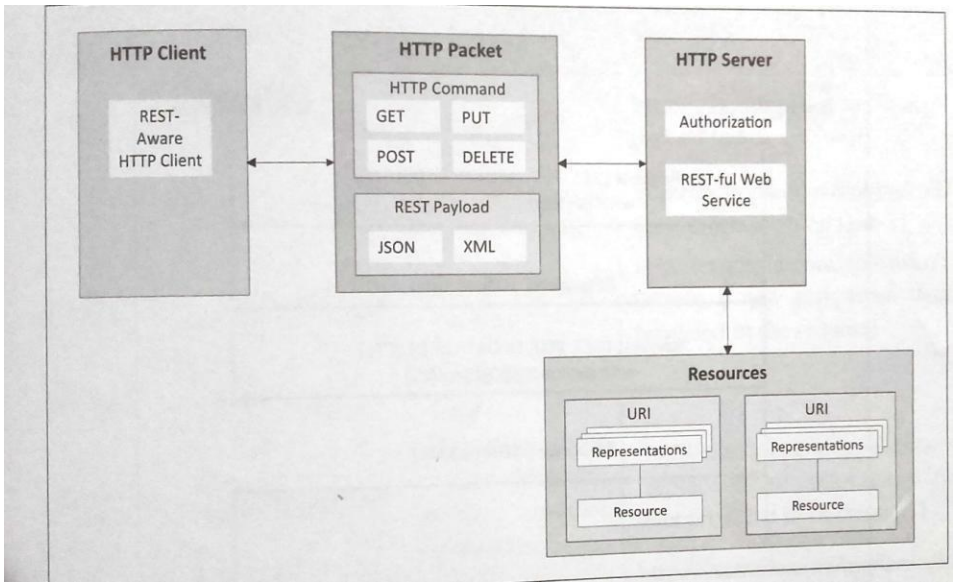
3) IoT Communication APIs:

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

b) WebSocket based Communication APIs(Exclusive PairBased Model)

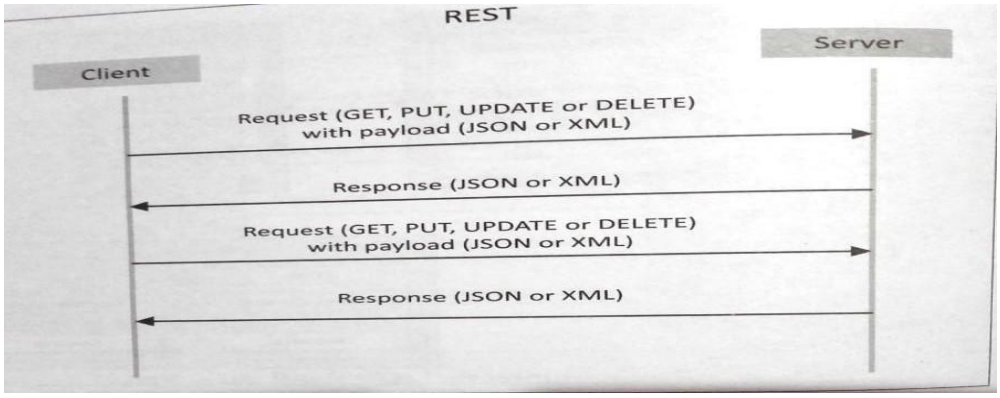
a) REST based communication APIs: Representational State Transfer(REST) is a set of architectural principles by which we can design web services and web APIs that focus on a system_s resources and have resource states are addressed and transferred.

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



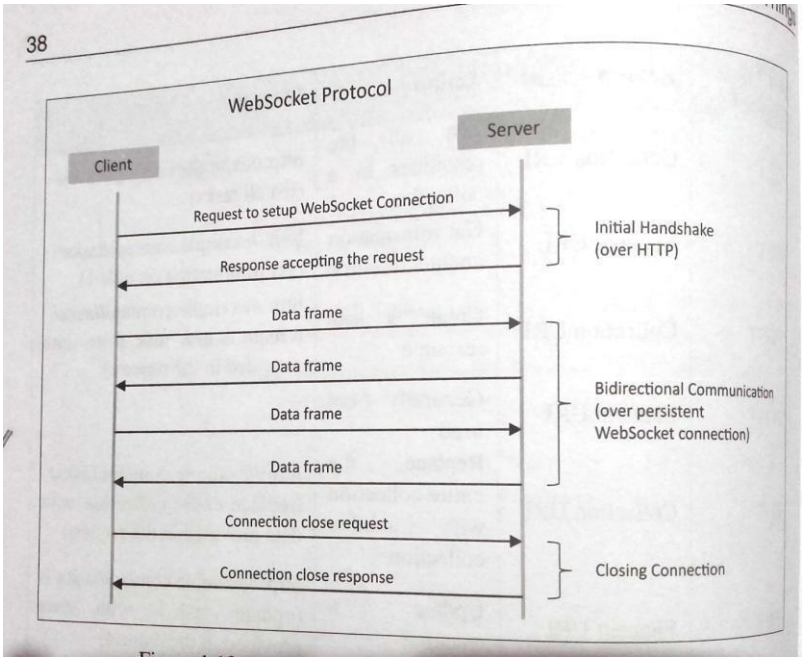
Client-Server: The principle behind client-server constraint is the separation of concerns. Separation allows client and server to be independently developed and updated.
Stateless: Each request from client to server must contain all the info. Necessary to understand the request, and cannot take advantage 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:



RESTful 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). ARESTful 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.

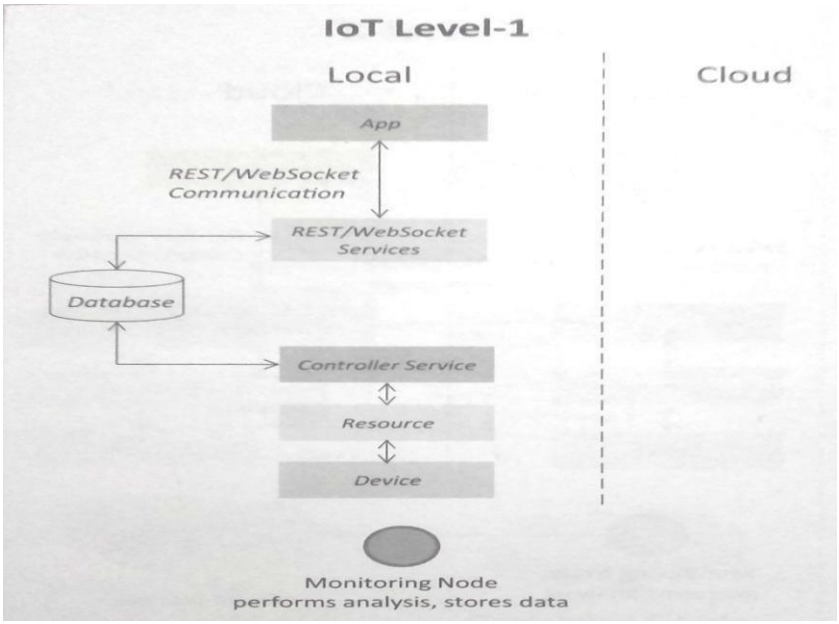
3) **Big Data Analytics:** Some examples of big data generated by IoT are
   - Sensor data generated by IoT systems.
   - Machine sensor data collected from sensors established in industrial and energy systems.
   - Health and fitness data generated IoT devices.
   - Data generated by IoT systems for 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.

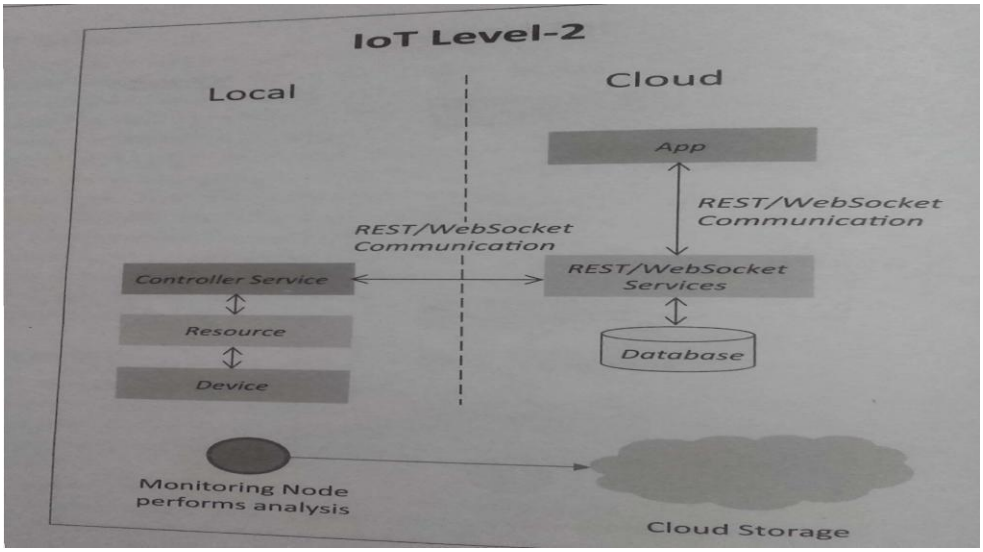
5) **Embedded Systems:** is a computer system that has computer hardware and software embedded to perform specific tasks. Embedded System range from low cost miniaturized devices such as digital watches to devices such as digital cameras, POS terminals,vending machines, appliances etc.,

IoT Levels and Deployment Templates

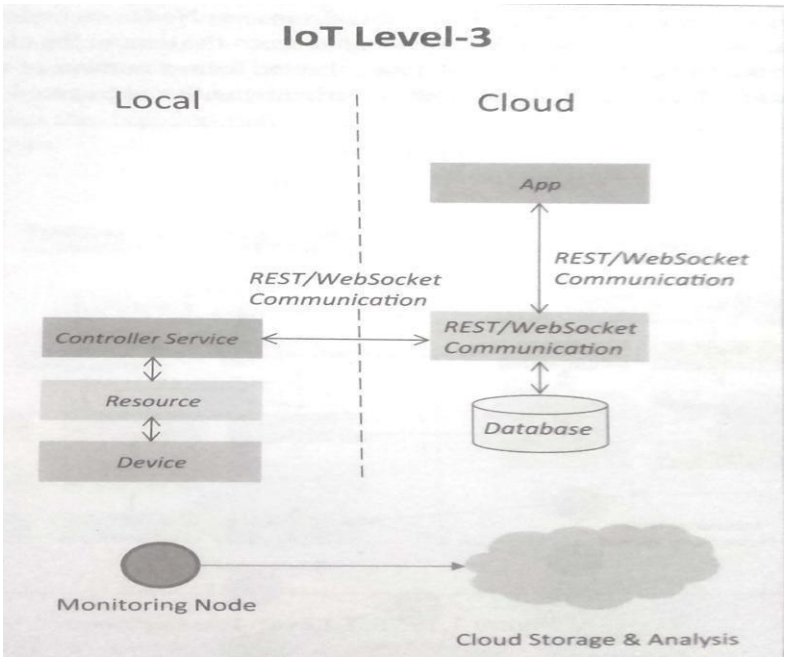
1) **IoT Level1:** System has a single node that performs sensing and/or actuation, stores data, performs analysis and host the application as shown in fig. Suitable for modeling low cost and low complexity solutions where the data involved is not big and analysisrequirement are not computationally intensive.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



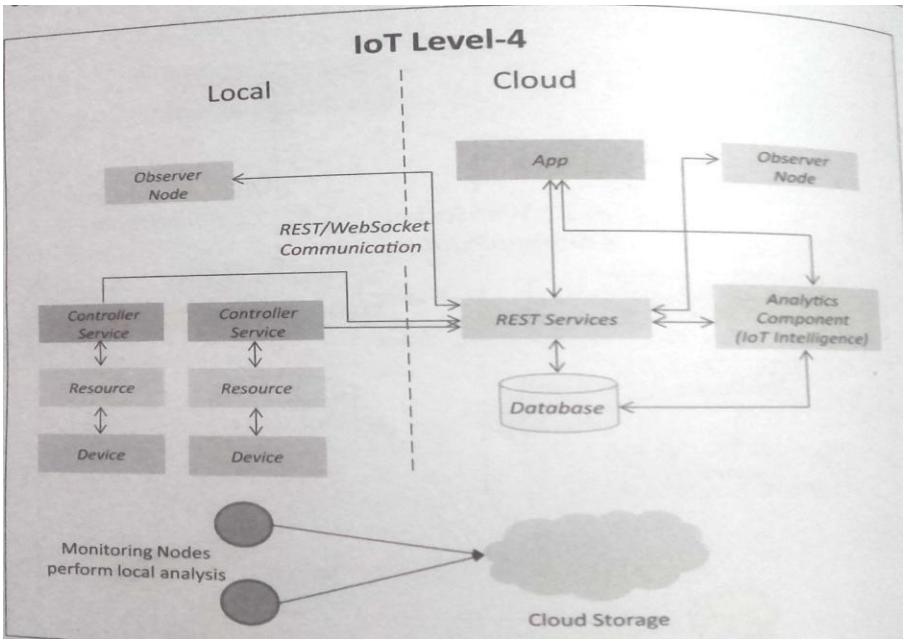
2) **IoT Level2:** has a single node that performs sensing and/or actuating and local analysisas shown in fig. Data is stored in cloud and application is usually cloud based. Level2 IoT systems are suitable for solutions where data are involved is big, however, the primary analysis requirement is not 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 handling.

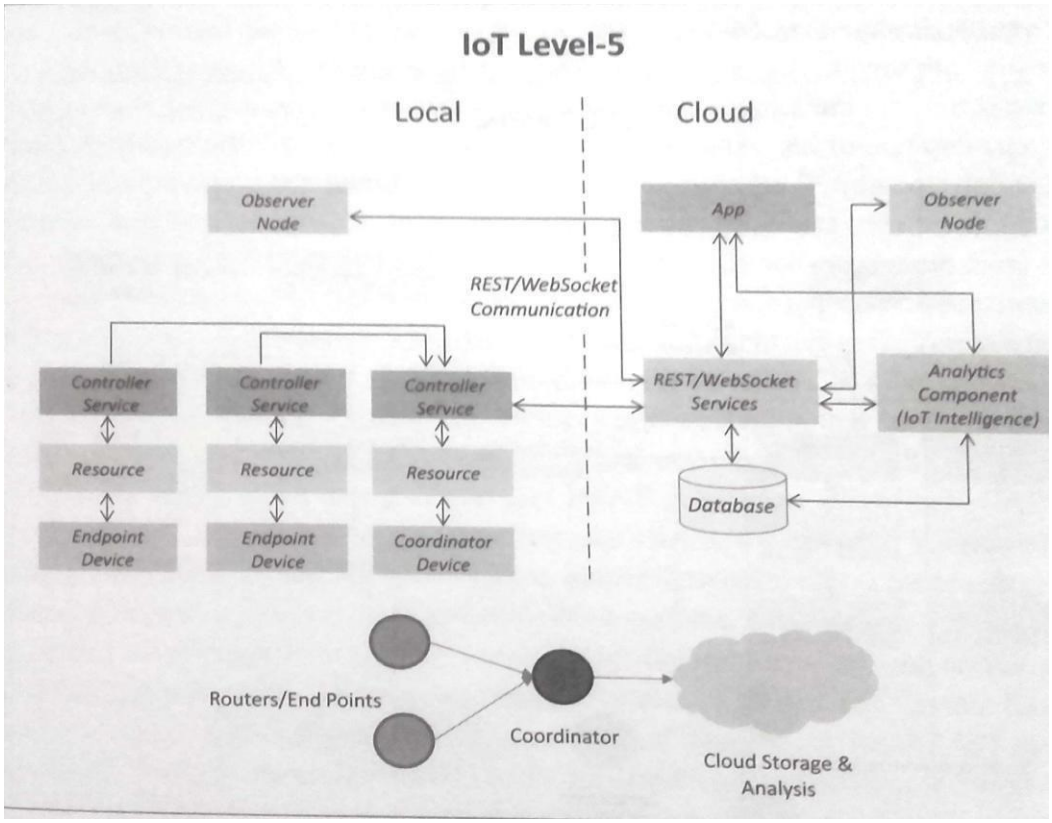


4) **IoT Level4:** System has multiple nodes that perform local analysis. Data is stored in the cloud and application is cloud based as shown in fig. Level4 contains local and cloud based observer nodes which can subscribe to and receive information collected in the cloud from IoT devices. An example of a Level4 IoT system for Noise Monitoring.

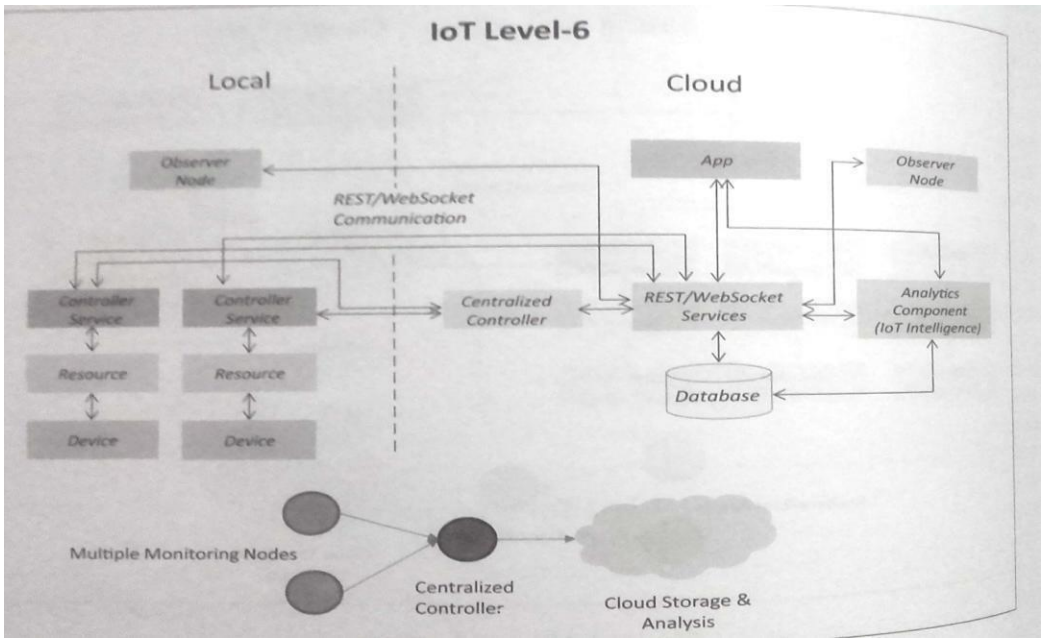


5) **IoT Level5:** System has multiple end nodes and one coordinator node as shown in fig. The end nodes that perform sensing and/or actuation. Coordinator node collects data from theendnodesandsendstothecloud.Dataisstoredandanalyzedinthecloudand

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 controlcommands to nodes. An example of a Level6 IoT system for Weather Monitoring System.



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 basednoise 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 Measurment Units(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

UNIT-II

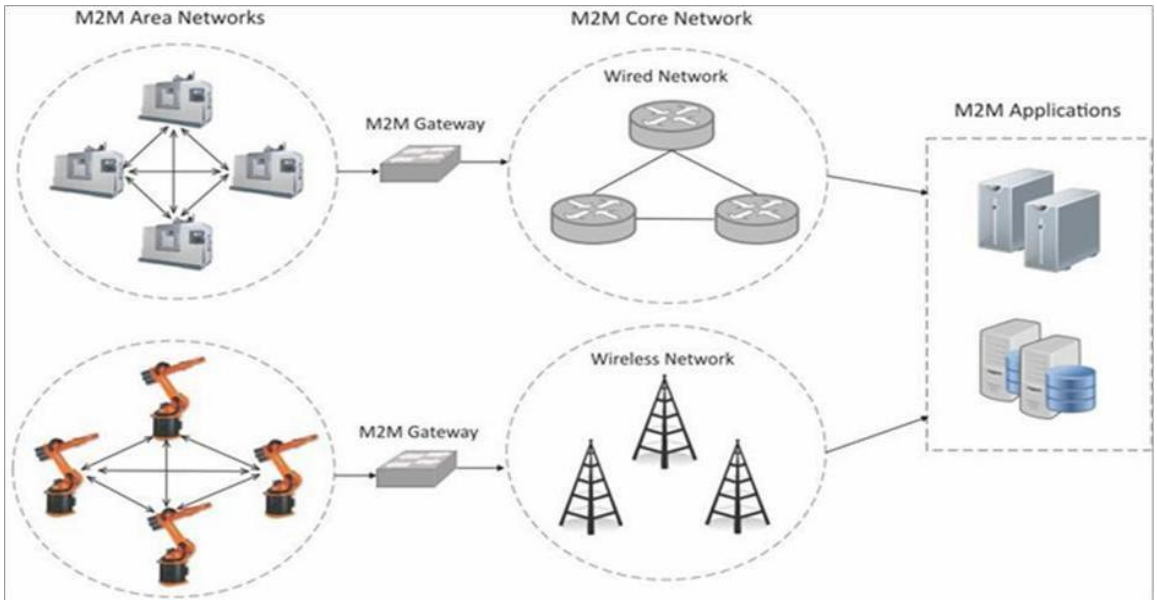
IoT and M2M

M2M:

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

- Term which is often synonymous with IoT is Machine-to-Machine (M2M).
- IoT and M2M are often used interchangeably.

Fig. Shows the end-to-end architecture of M2M systems comprises of M2M area networks, communication networks and application domain.



- An M2M area network comprises of machines(or M2M nodes) which have embedded network modules for sensing, actuation and communicating various communication protocols can be used for M2M LAN such as ZigBee, Bluetooth, M-bus, Wireless M-Bus etc., These protocols provide connectivity between M2M nodes within an M2M area network.
- The communication network provides connectivity to remote M2M area networks. The communication network provides connectivity to remote M2M area network. The communication network can use either wired or wireless network(IP based). While the M2M are networks use either properietorary or non-IP based communication protocols, the communication network uses IP-based network. Since non-IP based protocols areused within M2M area network, the M2M nodes within one network cannot communicate with nodes in an external network.
- To enable the communication between remote M2M are network, M2M gateways are used.

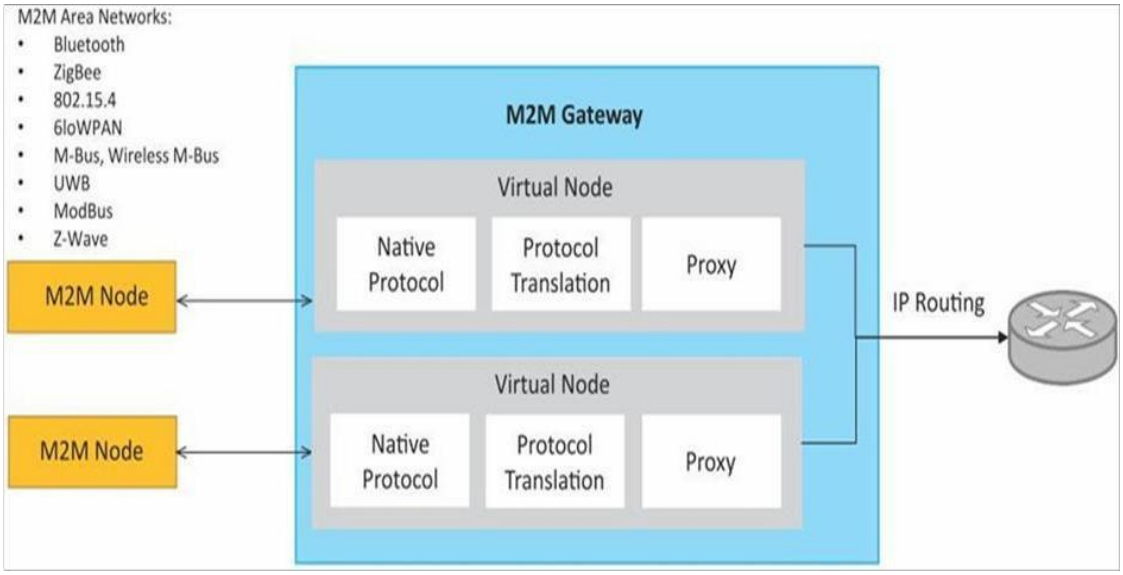


Fig. Shows a block diagram of an M2M gateway. The communication between M2M nodes and the M2M gateway is based on the communication protocols which are naive to the M2M are network. M2M gateway performs protocol translations to enable Ip-connectivity for M2M are networks. M2M gateway acts as a proxy performing translations from/to native protocols to/from Internet Protocol(IP). With an M2M gateway, each mode in an M2M area network appears as a virtualized node for external M2M area networks.

Differences between IoT and M2M

1) **Communication Protocols:**
   - Commonly uses M2M protocols include ZigBee, Bluetooth, ModBus, M-Bus, Wireless M-Bus tec.,
   - In IoT uses HTTP, CoAP, WebSocket , MQTT ,XMPP ,DDS ,AMQP etc.,
2) **Machines in M2M Vs Things in IoT:**
   - Machines in M2M will be homogenous whereas Things in IoT will be heterogeneous.
3) **Hardware Vs Software Emphasis:**
   - the emphasis of M2M is more on hardware with embedded modules, the emphasis of IoT is more on software.
4) **Data Collection &Analysis**
   - M2M data is collected in point solutions and often in on-premises storage infrastructure.
   - The data in IoT is collected in the cloud (can be public, private orhybrid cloud).

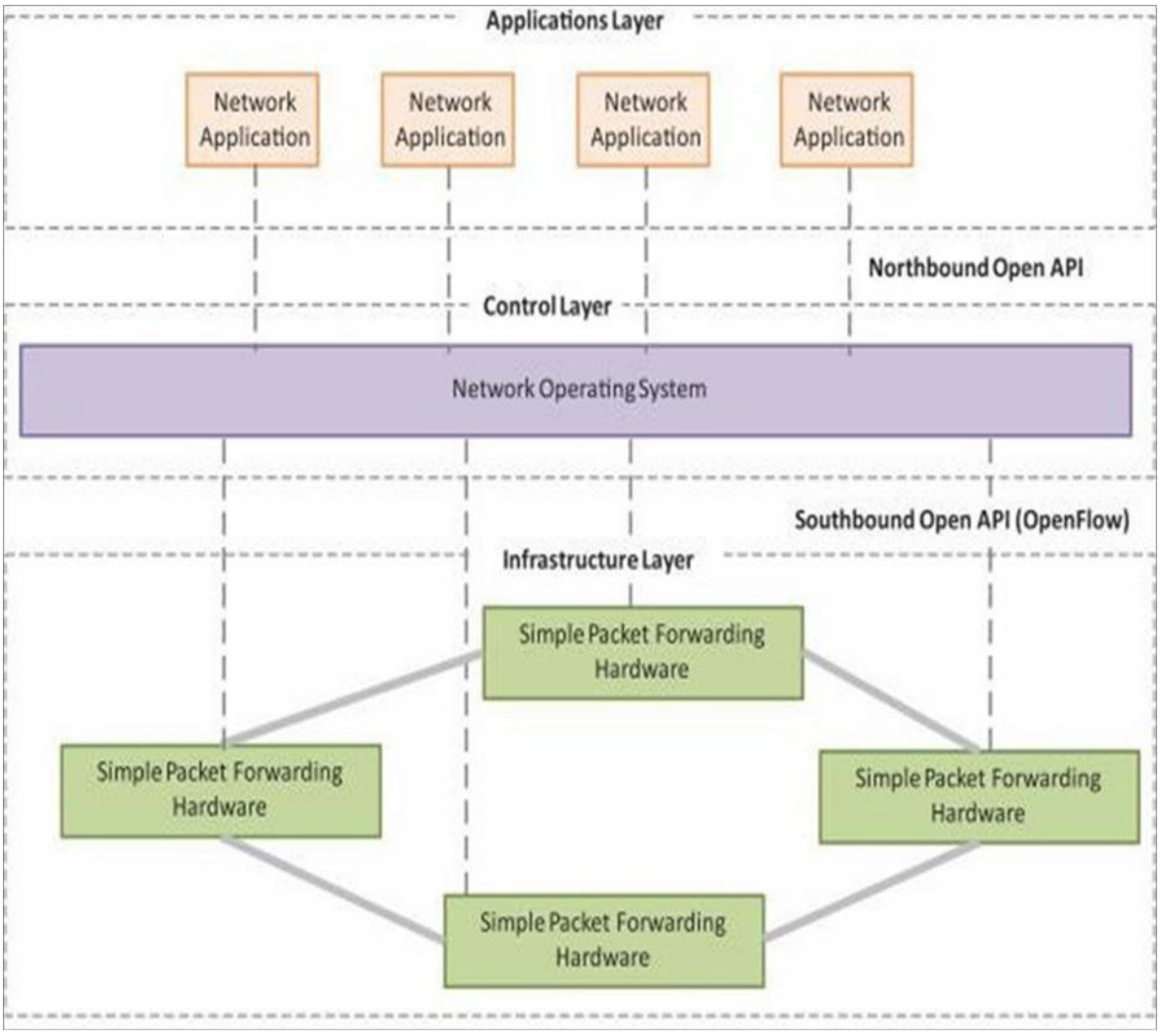
5) **Applications**

- □ M2M data is collected in point solutions and can be accessed by on-premises applications such as diagnosis applications, service management applications, and on- premisis enterprise applications.
- □ IoT data is collected in the cloud and can be accessed by cloud applications such as analytics applications, enterprise applications, remote diagnosis and management applications, etc.

SDN and NVF for IoT

Software Defined Networking(SDN):

- Software-Defined Networking (SDN) is a networking architecture that separates the control plane from the data plane and centralizes the network controller.
- Software-based SDN controllers maintain a united view of the network
- The underlying infrastructure in SDN uses simple packet forwarding hardware as opposed to specialized hardware in conventional networks.



SDN Architecture

Key elements of SDN:

1) **Centralized Network Controller**

   With decoupled control and data planes and centralized network controller, the network administrators can rapidly configure the network.

2) **Programmable Open APIs**

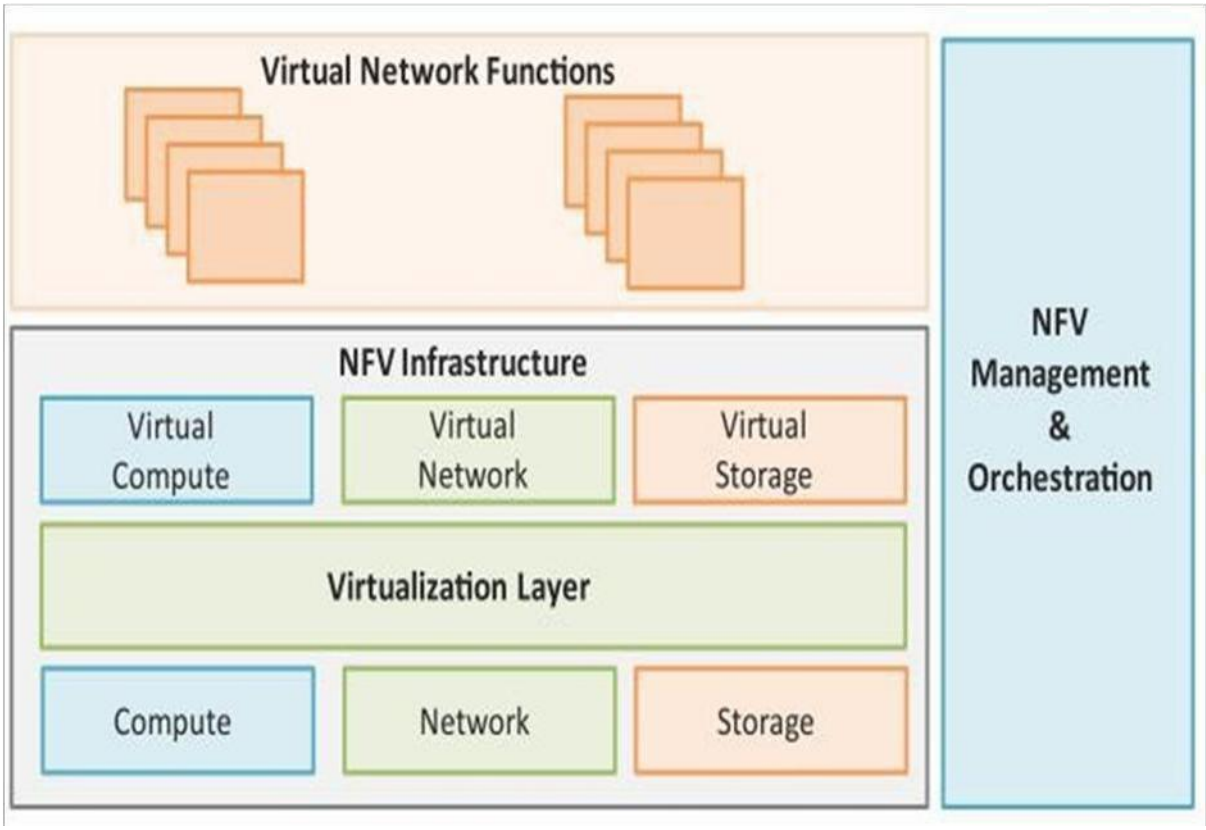
   SDN architecture supports programmable open APIs for interface between the SDN application and control layers (Northbound interface).

3) **Standard Communication Interface(OpenFlow)**

   SDN architecture uses a standard communication interface between the control and infrastructure layers (Southbound interface). OpenFlow, which is defined by the Open Networking Foundation (ONF) is the broadly accepted SDN protocolfor the Southbound interface.

Network Function Virtualization(NFV)

- Network Function Virtualization (NFV) is a technology that leverages virtualization to consolidate the heterogeneous network devices onto industry standard high volume servers, switches and storage.
- NFV is complementary to SDN as NFV can provide the infrastructure on which SDN can run.



NFV Architecture

Key elements of NFV:

1) **Virtualized Network Function(VNF):**

   VNF is a software implementation of a network function which is capable of running over the NFV Infrastructure (NFVI).

2) **NFV Infrastructure(NFVI):**

   NFVI includes compute, network and storage resources that are virtualized.

3) **NFV Management and Orchestration:**

   NFV Management and Orchestration focuses on all virtualization-specific management tasks and covers the orchestration and life-cycle management of physical and/or software resources that support the infrastructure virtualization, and the life-cycle management of VNFs.

Need for IoT Systems Management

Managing multiple devices within a single system requires advanced management capabilities.

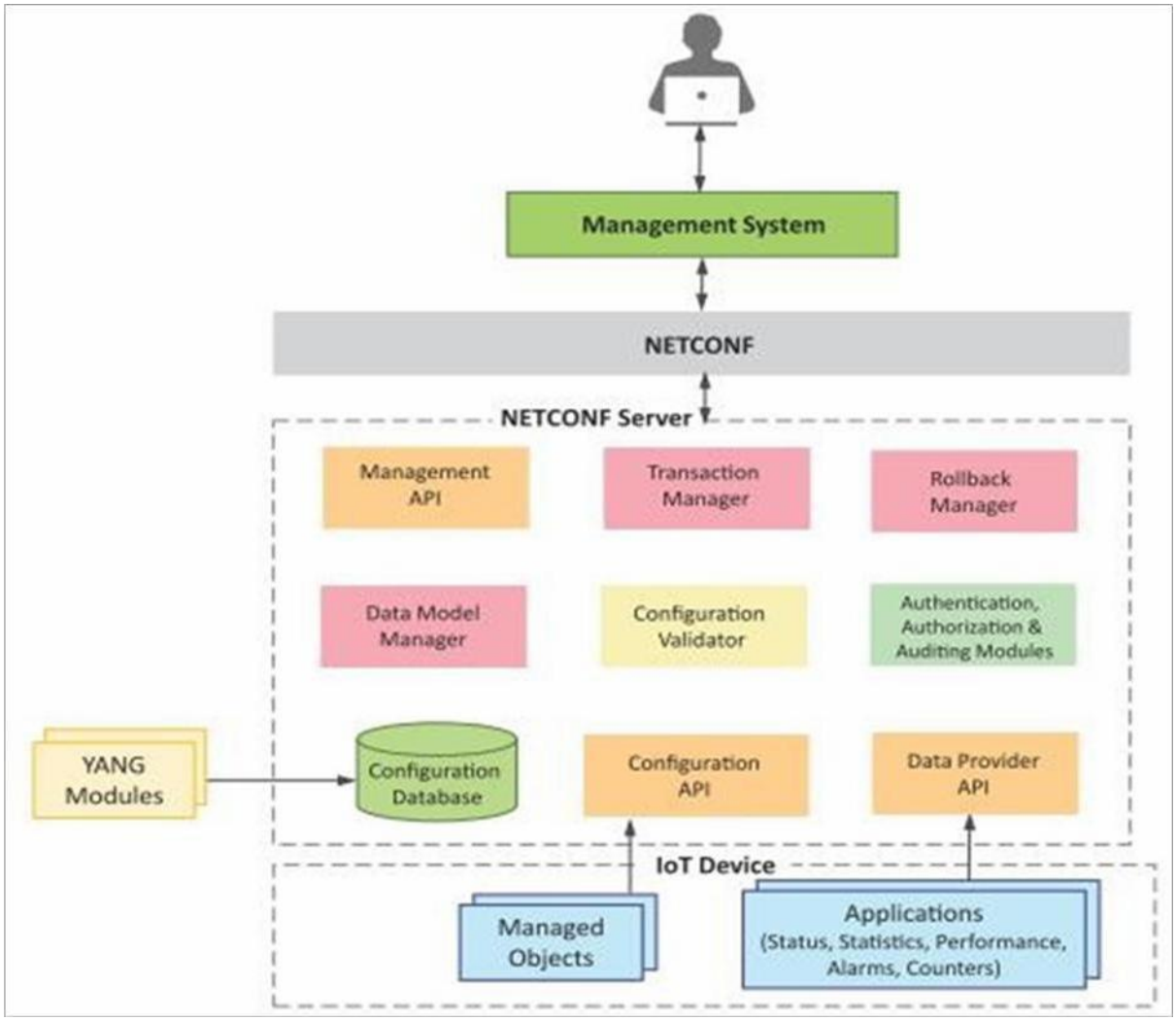
1) **Automating Configuration :** IoT system management capabilities can helpin automating the system configuration.
2) **Monitoring Operational & Statistical Data :** Management systems can help in monitoring opeartional and statistical data of a system. This data can be used for fault diagnosis or prognosis.
3) **Improved Reliability:** A management system that allows validating the system configurations before they are put into effect can help in improving the system reliability.
4) **System Wide Configurations :** For IoT systems that consists of multiple devices or nodes, ensuring system wide configuration can be critical for the correct functioning of the system.
5) **Multiple System Configurations :** For some systems it may be desirable to have multiple valid configurations which are applied at different times or in certain conditions.
6) **Retrieving & Reusing Configurations :** Management systems which have the capability of retrieving configurations from devices can help in reusing the configurations for other devices of the same type.

IoT Systems Management with NETCONF-YANG

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

The generic approach of IoT device management weith NETCONF-YANG. Roles of various componentsare:

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 torollback a current configuration to its original 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 datastore and write operational data to the operational datastore.
9) **Data Provider API:** Applications on the 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 operational ldata.

Steps for IoT device Management with NETCONF-YANG

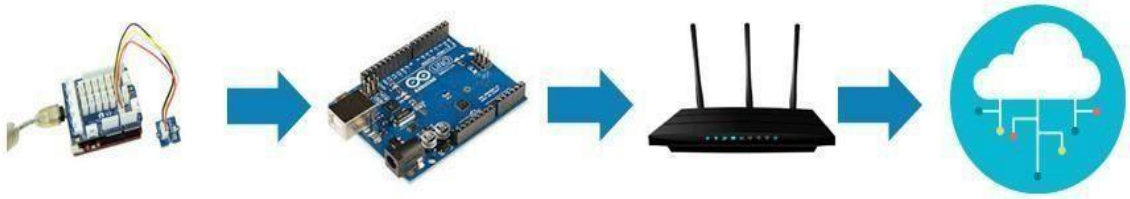
1) Create a YANG model of the system that defines the configuration and state data of the system.
2) Complete the YANG model with the _Inctool_ which comes withLibnetconf.
3) Fill in the IoT device management code in the TransAPImodule.
4) Build the callbacks C file to generate the libraryfile.
5) Load the YANG module and the TransAPImodule into the Netopeer server using Netopeer manager tool.
6) The operator can now connect from the management system to the Netopeer server using the NetopeerCLI.
7) Operator can issue NETCONF commands from the Netopeer CLI. Command can be issued to change the configuration data, get operational data or execute an RPC on the IoTdevice.

UNIT-III

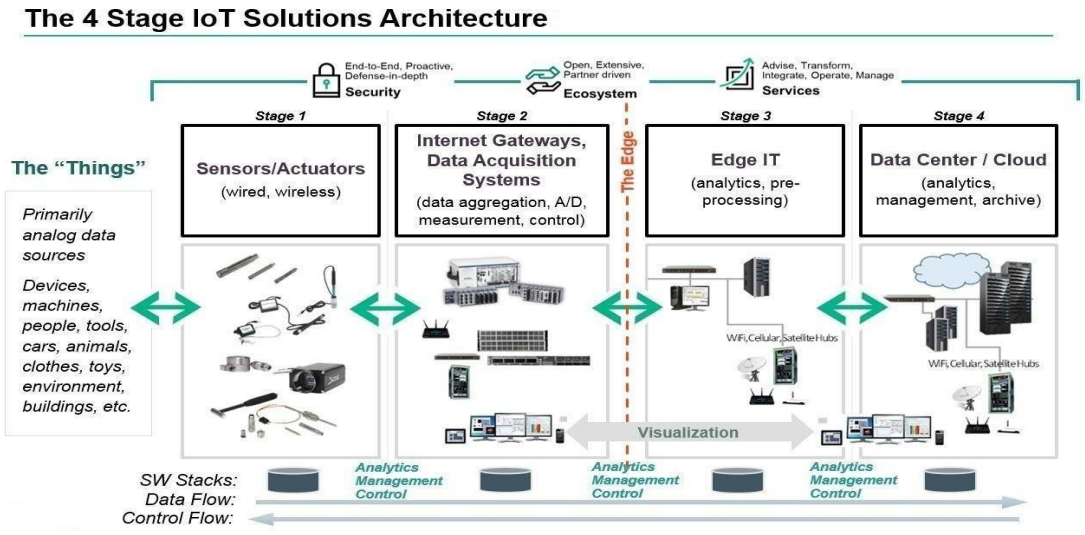
State of the art

IoT architecture varies from solution to solution, based on the type of solution which we intend to build. IoT as a technology majorly consists of four main components, over which an architecture is framed.

1) Sensors
2) Devices
3) Gateway
4) Cloud



Stages of IoT Architecture



Stage 1:-

Sensors/actuators

Sensors collect data from the environment or object under measurement and turn it into useful data. Think of the specialized structures in your cell phone that detect the directional pull of gravity and the phone's relative position to the ‒thing| we call the earth and convert it into data that your phone can use to orient the device.

Actuators can also intervene to change the physical conditions that generate the data. An actuator might, for example, shut off a power supply, adjust an air flow valve, or move a robotic gripper in an assembly process.

The sensing/actuating stage covers everything from legacy industrial devices to robotic camera systems, water level detectors, air quality sensors, accelerometers, and heart rate monitors. And the scope of the IoT is expanding rapidly, thanks in part to low-power wireless sensor network technologies and Power over Ethernet, which enable devices on a wired LAN to operate without the need for an A/C power source.

Stage 2:-

The Internet gateway

The data from the sensors starts in analog form. That data needs to be aggregated and converted into digital streams for further processing downstream. Data acquisition systems(DAS) perform these data aggregation and conversion functions. The DAS connects to the sensornetwork, aggregates outputs, and performs the analog-to-digital conversion. The Internet gateway receives the aggregated and digitized data and routes it over Wi-Fi, wired LANs, or the Internet, to Stage 3 systems for further processing. Stage 2 systems often sit in close proximity tothe sensors and actuators.

For example, a pump might contain a half-dozen sensors and actuators that feed data into a data aggregation device that also digitizes the data. This device might be physically attached to the pump. An adjacent gateway device or server would then process the data and forward it to the Stage 3 or Stage 4 systems. Intelligent gateways can build on additional, basic gateway functionality by adding such capabilities as analytics, malware protection, and data management services. These systems enable the analysis of data streams in real time.

Stage 3:-

Edge IT

Once IoT data has been digitized and aggregated, it's ready to cross into the realm of IT. However, the data may require further processing before it enters the data center. This is where edge IT systems, which perform more analysis, come into play. Edge IT processing systems maybe located in remote offices or other edge locations, but generally these sit in the facility or location where the sensors reside closer to the sensors, such as in a wiring closet. Because IoT data can easily eat up network bandwidth and swamp your data center resources, it's best to have systems at the edge capable of performing analytics as a way to lessen the burden on core IT infrastructure. You'd also face security concerns, storage issues, and delays processing the data. With a staged approach, you can preprocess the data, generate meaningful results, and pass only those on. For example, rather than passing on raw vibration data for the pumps, you could

aggregate and convert the data, analyze it, and send only projections as to when each device will fail or need service.

Stage 4:-

The data center and cloud

Data that needs more in-depth processing, and where feedback doesn't have to be immediate, gets forwarded to physical data center or cloud-based systems, where more powerful IT systems can analyze, manage, and securely store the data. It takes longer to get results when you wait until data reaches Stage 4, but you can execute a more in-depth analysis, as well as combine yoursensor data with data from other sources for deeper insights. Stage 4 processing may take place on-premises, in the cloud, or in a hybrid cloud system, but the type of processing executed in thisstage remains the same, regardless of the platform.

REFERENCE MODEL AND ARCHITECTURE

Reference Architecture that describes essential building blocks as well as design choices to deal with conflicting requirements regarding functionality, performance, deployment and security. Interfaces should be standardized, best practices in terms of functionality and information usage need to be provided.

The central choice of the IoT-A project was to base its work on the current state of the art, rather than using a clean-slate approach. Due to this choice, common traits are derived to form the base line of the Architectural Reference Model (ARM). This has the major advantageof ensuring backward compatibility of the model and also the adoption of established, working solutions to various aspects of the IoT. With the help of end users, organised into a stakeholders group, new requirements for IoT have been collected and introduced in the main model building process. This work was conducted according to established architecturemethodology.

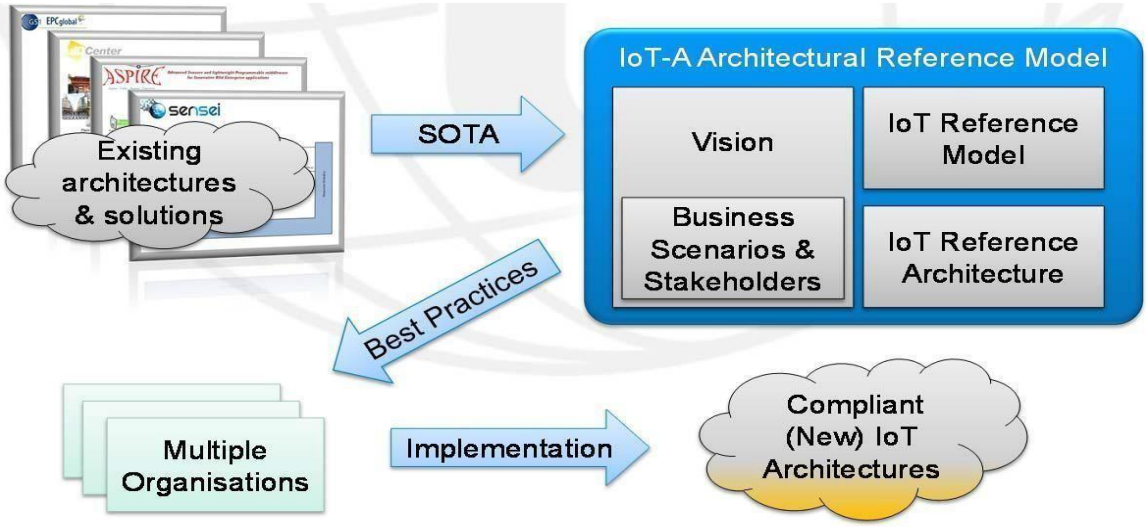
A*Reference Architecture (RA) can be visualised as*the –Matrixthat eventuallygives birth ideally to all concrete architectures. For establishing such a Matrix, based on a strong and exhaustive analysis of the State of the Art, we need to envisage the superset of all possible functionalities, mechanisms and protocols that can be used for building such concrete architecture and to show how interconnections could take place between selected ones (as no concrete system is likely to use all of the functional possibilities). Giving such a foundationalong with a set of design-choices, based on the characterisation of the targeted system w.r.t. various dimensions (like distribution, security, real-time, semantics) it becomes possible for a system architect to select the protocols, functional components, architectural options, needed to build their IoT systems.

As any metaphoric representation, this tree does not claim to be fully consistent in its depiction; it should therefore not be interpreted too strictly. On the one hand, the roots of this tree are spanning across a selected set of communication protocols (6LoWPAN, Zigbee, IPv6,…) and device technologies (sensors, actuators, tags,..) while on the other hand the blossoms / leaves of the tree represent the whole set of IoT applications that can be built from the sap (i.e., data and information) coming from the roots. The trunk of the tree is of utmost importance here, as it represent the Architectural Reference Model (ARM). The ARM is the combination of the Reference Model and the Reference Architecture, the set of models, guidelines, best practices, views and perspectives that can be usedfor building fully

interoperable concrete IoT architectures and systems. In this tree, we aim at selecting a minimal set of interoperable technologies (the roots) and proposing the potentially necessary set of enablers or building blocks (the trunk) that enable the creation of a maximal set of interoperable IoT systems (the leaves).

The IOT-A Tree

IoT-A architectural reference model building blocks.



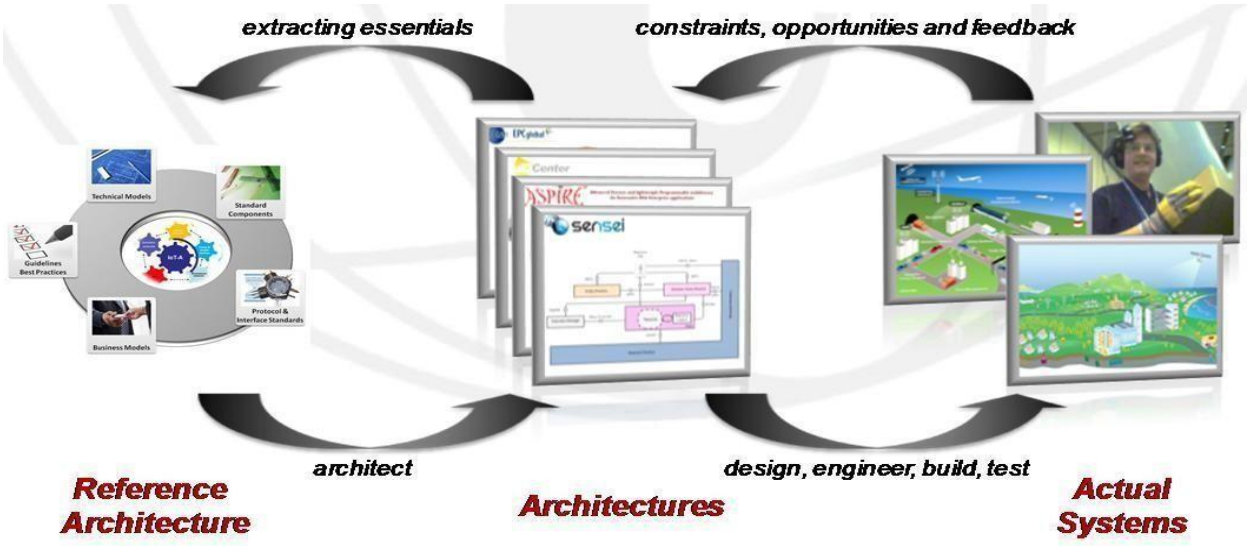
Starting with existing architectures and solutions, generic baseline requirements can be extracted and used as an input to the design. The IoT-A ARM consists of four parts:

The vision summarizes the rationale for providing an architectural reference model for the IoT. At the same time it discusses underlying assumptions, such as motivations. It also discusses how the architectural reference model can be used, the methodology applied to the architecture modeling, and the business scenarios and stakeholders addressed.

Business scenarios defined as requirements by stakeholders are the drivers of the architecture work. With the knowledge of businesses aspirations, a holistic view of IoT architectures can be derived.

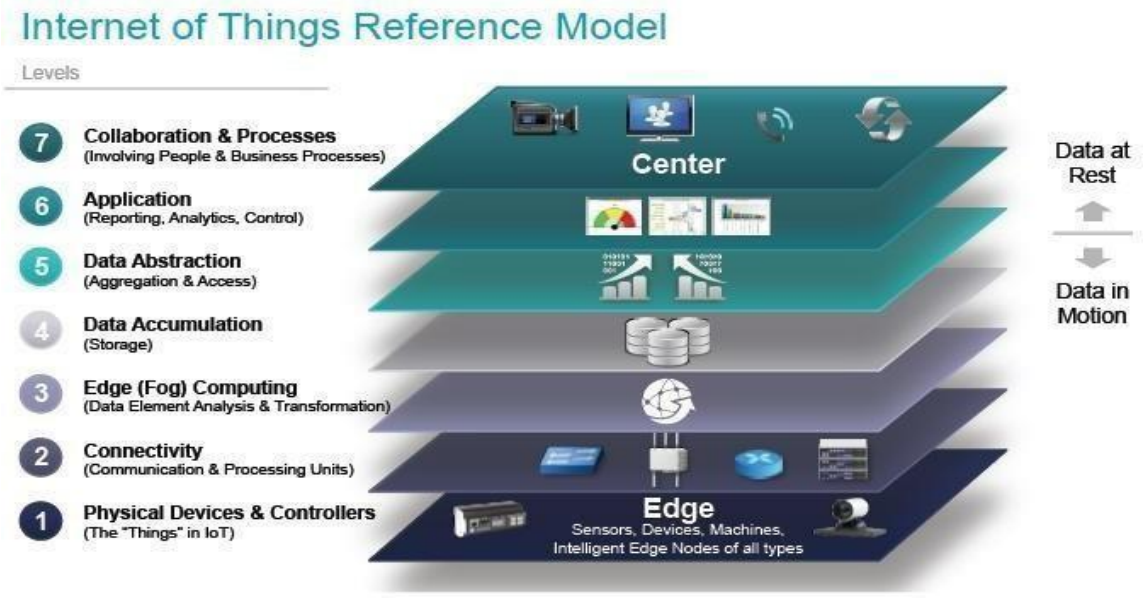
The IoT Reference Model provides the highest abstraction level for the definition of the IoT-A Architectural Reference Model. It promotes a common understanding of the IoT domain. The description of the IoT Reference Model includes a general discourse on the IoT domain, an IoT Domain Model as a top-level description, an IoT Information Model explaining how IoT information is going to be modeled, and an IoT Communication Model in order to understand specifics about communication between many heterogeneous IoT devices and the Internet as a whole.

The IoT Reference Architecture is the reference for building compliant IoT architectures. As such, it provides views and perspectives on different architectural aspects that are of concern to stakeholders of the IoT. The terms view and perspectives are used according to the general literature and standards the creation of the IoT Reference Architecture focuses on abstract setsof mechanisms rather than concrete application architectures. To organizations, an important aspect is the compliance of their technologies with standards and best practices, so that interoperability across organizations is ensured.



In an IoT system, data is generated by multiple kinds of devices, processed in different ways, transmitted to different locations, and acted upon by applications. The proposed IoT reference model is comprised of seven levels. Each level is defined with terminology that can be standardized to create a globally accepted frame of reference.

- ✓ **Simplifies**: It helps break down complex systems so that each part is moreunderstandable. **Clarifies**: It provides additional information to precisely identify levelsof the IoT and to establish common terminology.
- ✓ **Identifies**: It identifies where specific types of processing is optimized across different parts of the system.
- ✓ **Standardizes**: It provides a first step in enabling vendors to create IoT products that work with each other.
- ✓ **Organizes**: It makes the IoT real and approachable, instead of simply conceptual.



Level 1: Physical Devices and Controllers

The IoT Reference Model starts with Level 1: physical devices and controllers that might control multiple devices.These are the ‑things‑in the IoT, and theyinclude a wide range of endpoint devices that send and receive information. Today, the list of devices is already extensive. It will become almost unlimited as more equipment is added to the IoT over time. Devices are diverse, and there are no rules about size, location, form factor, or origin. Some devices will be the size of a silicon chip. Some will be as large as vehicles. The IoT must supportthe entire range. Dozens or hundreds of equipment manufacturers will produce IoT devices. To simplify compatibility and support manufacturability, the IoT Reference Model generally describes the level of processing needed from Level 1devices.

Level 2: Connectivity

Communications and connectivity are concentrated in one level—Level 2. The most important function of Level 2 is reliable, timely information transmission. This includes transmissions:

- Between devices (Level 1) and the network
- Across networks(east-west)
- Between the network (Level 2) and low-level information processing occurring at Level 3

Traditionaldatacommunicationnetworkshavemultiplefunctions,asevidencedbythe International Organization for Standardization (ISO) 7-layer reference model. However, a complete IoT system contains many levels in addition to the communications network. One objective of the IoT Reference Model is for communications and processing to be executed by existing networks. The IoT Reference Model does not require or indicate creation of a different network—it relies on existing networks. As Level 1 devices proliferate, the ways in which they interact with Level 2 connectivity equipment may change. Regardless of the details, Level 1 devices communicate through the IoT system by interacting with Level 2 connectivity equipment.

2 Connectivity
(Communication & Processing Units)

Level 2 functionality focuses on East-West communications

Connectivity includes:

Communicating with Bnd between theLevel I devices

- Reliable delivery Bcross the network(s)
- Implementation of various protocols
- Switching and routing Translation between protocols Security at the network level

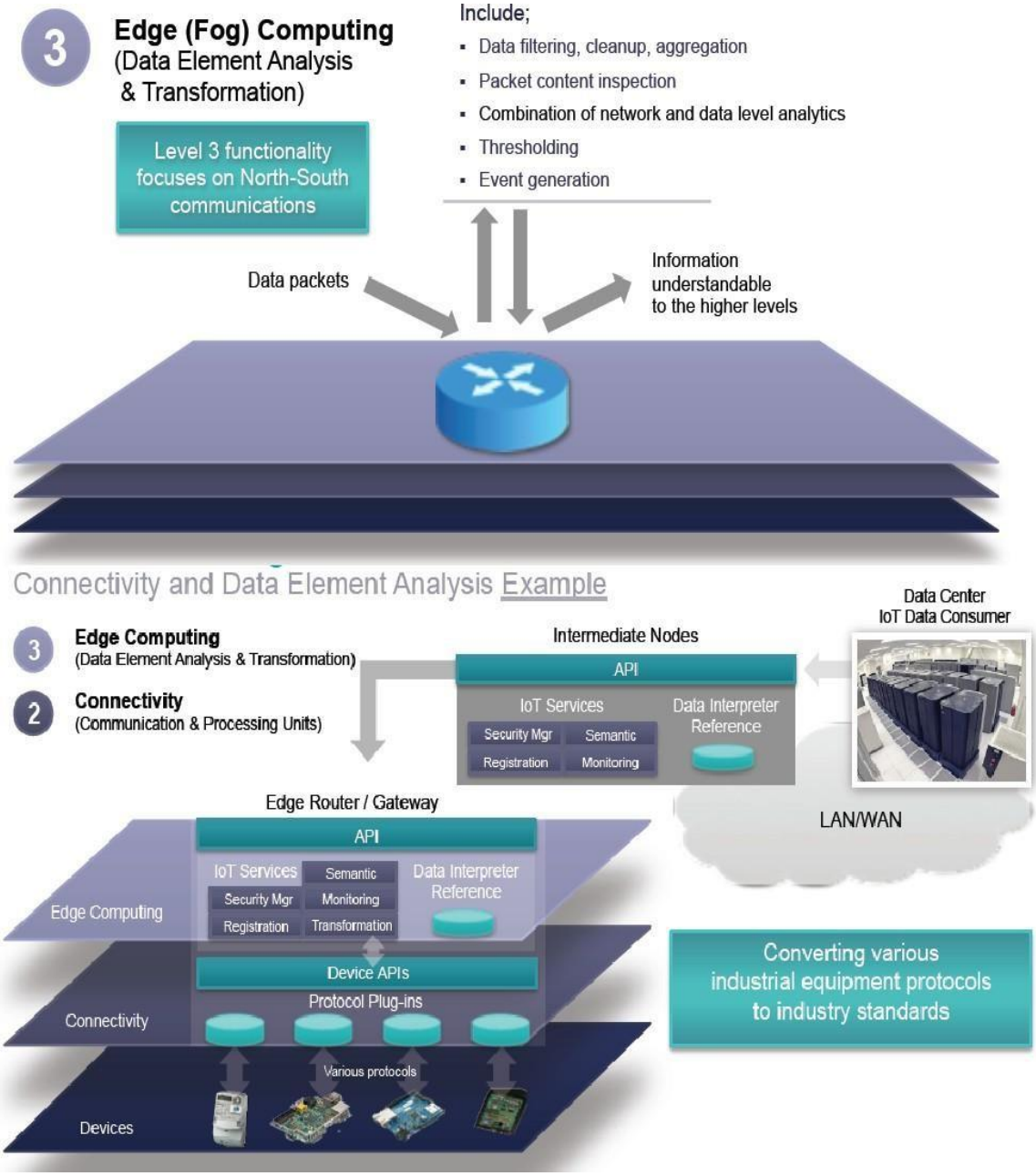
(Self Learning) Networking Analysis

Level 3: Edge (Fog) Computing

The functions of Level 3 are driven by the need to convert network data flows into information that is suitable for storage and higher level processing at Level 4 (data accumulation). This means that Level 3 activities focus on high-volume data analysis and transformation. For example, a Level 1 sensor device might generate data samples multiple times per second, 24 hours a day, 365 days a year. A basic tenet of the IoT Reference Model is that the most intelligent system initiates information processing as early and as close to the edge of the network as possible. This is sometimes referred to as fog computing. Level 3 is where this occurs.

Given that data is usually submitted to the connectivity level (Level 2) networking equipment by devices in small units, Level 3 processing is performed on a packet-by-packet basis. This processing is limited, because there is only awareness of data uni nol "sessions‖ or –transactions." Level 3 processing can encompass many examples, such as:

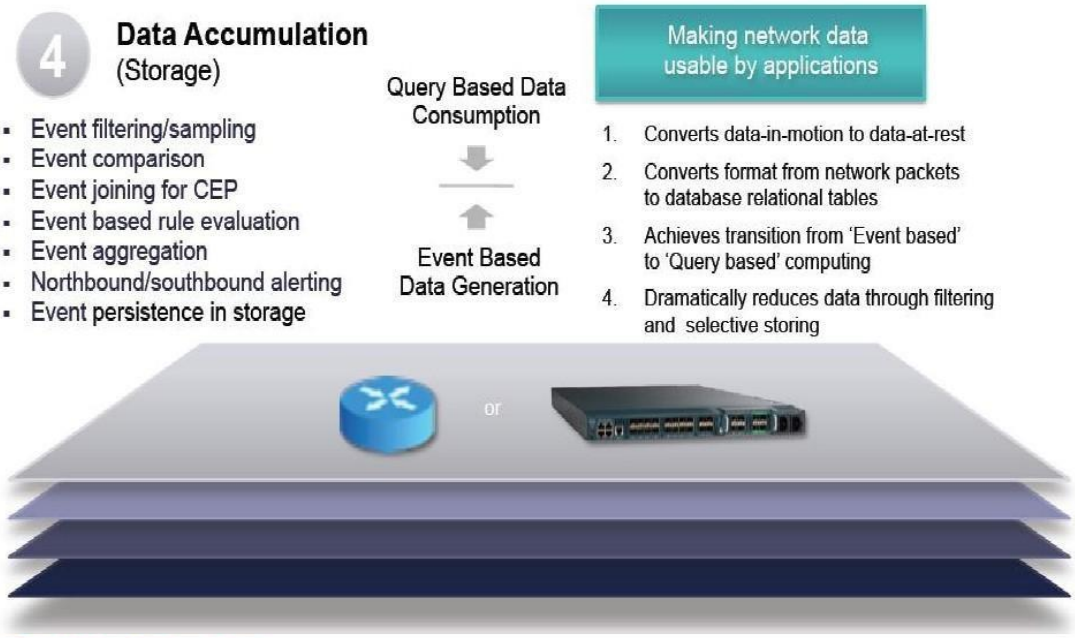
- Evaluation: Evaluating data for criteria as to whether it should be processed at a higher level
- Formatting: Reformatting data for consistent higher-level processing
- Expanding/decoding. Handling cryptic data with additional context (such as the origin)
- Distillation/reduction: Reducing and/or summarizing data to minimize the impact of data and traffic on the network and higher-level processingsystems
- Assessment. Determining whether data represents a threshold or alert; this could include redirecting data to additional destinations.


3
Edge (Fog) Computing
(Data Element Analysis
& Transformation)
Include;
Data filtering, cleanup, aggregation
Packet content inspection
Combination of network and data level analytics
Thresholding
Event generation
Level 3 functionality focuses on North-South communications
Data packets
Information understandable to the higher levels
Connectivity and Data Element Analysis Example
3
Edge Computing
(Data Element Analysis & Transformation)
2
Connectivity
(Communication & Processing Units)
Intermediate Nodes
API
IoT Services
Security Mgr
Semantic
Registration
Monitoring
Data Interpreter Reference
Data Center
IoT Data Consumer
LAN/WAN
Edge Router / Gateway
API
IoT Services
Semantic
Security Mgr
Monitoring
Registration
Transformation
Data Interpreter Reference
Edge Computing
Device APIs
Protocol Plug-ins
Connectivity
Various protocols
Devices
Converting various industrial equipment protocols to industry standards

Level 4: Data Accumulation

Networking systems are built to reliably move data. The data is "in motion." Prior to Level 4, data is moving through the network at the rate and organization determined by the devices generating the data. The model is event driven. As defined earlier, Level 1 devices do not include computing capabilities themselves. However, some computational activities could occur at Level 2, such as protocol translation or application of network security policy. Additional compute tasks can be performed at Level 3, such as packet inspection. Driving computational tasks as close to the edge of the IoT as possible, with heterogeneous systems distributed across multiple management domains represents an example of fog computing. Fog computing and fog services will be a distinguishing characteristic of the IoT.

As Level 4 captures data and puts it at rest, it is now usable by applications on a non-real-time basis. Applications access the data when necessary. In short, Level 4 converts event-based data to query-based processing. This is a crucial step in bridging the differences between the real-time networking world and the non-real-time application world. Figure 6 summarizes the activities that occur at Level 4.



Level 5: Data Abstraction

IoT systems will need to scale to a corporate—or even global—level and will require multiple storage systems to accommodate IoT device data and data from traditional enterprise ERP, HRMS, CRM, and other systems. The data abstraction functions of Level 5 are focused on rendering data and its storage in ways that enable developing simpler, performance-enhanced applications.

With multiple devices generating data, there are many reasons why this data may not land in the same data storage:

- There might be too much data to put in one place.
- Moving data into a database might consume too much processing power, so that retrieving it must be separated from the data generation process. This is done today with online transaction processing (OLTP) databases and data warehouses.
- Devices might be geographically separated, and processing is optimized locally.
- Levels 3 and 4 might separate "continuous streams of raw data" from "data that represents an event." Data storage for streaming data may be a big data system, such as Hadoop. Storage for event data may be a relational database management system (RDBMS) with faster query times.
- Different kinds of data processing might be required. For example, in-store processing will focus on different things than across-all-stores summary processing.

For these reasons, the data abstraction level must process many different things. These include:

- Reconciling multiple data formats from different sources
- Assuring consistent semantics of data across sources
- Confirming that data is complete to the higher-level application

Level 6: Application

Level 6 is the application level, where information **interpretation occurs.** Software at th is level interacts with Level 5 and data at rest, so it does not have to operate at network **speeds.**

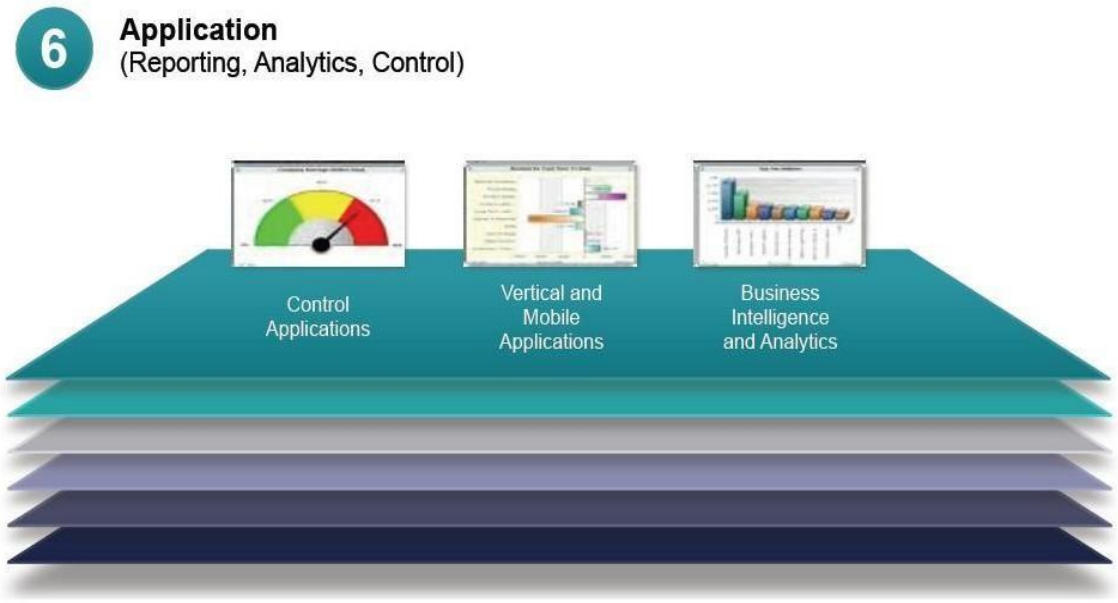
The IoT Reference Model does not strictly define an application. Applications vary based on vertical markets, the nature of device data, and **business** needs. For example, some applications will focus an monitoring device data. Some will focus on controlling devices. Some will combine device and non-device data. Monitoring and control applications represent many different application models, programming patterns, ancJ software stacks, leading to discussions of operating systems, mobility, application servers, hypervisors, multi-threading, multi-tenancy, etc. These topics are beyond the scope of the IoT Reference Model discussion. Suffice it to say that application complexity will vary widely.

Examples include:

Mission-critical business applications, such as generalized ERP or specialized industry solutions Mobile applications that handle simple interactions

- Business intelligence reports, where the application is the BI server Analytic applications that interpret data for business decisions

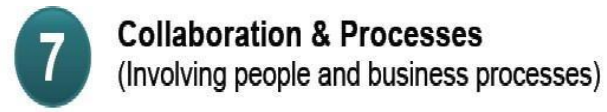
- System management/con1rol center applications that control the IoT system ibelf and don't act on the data produced by it

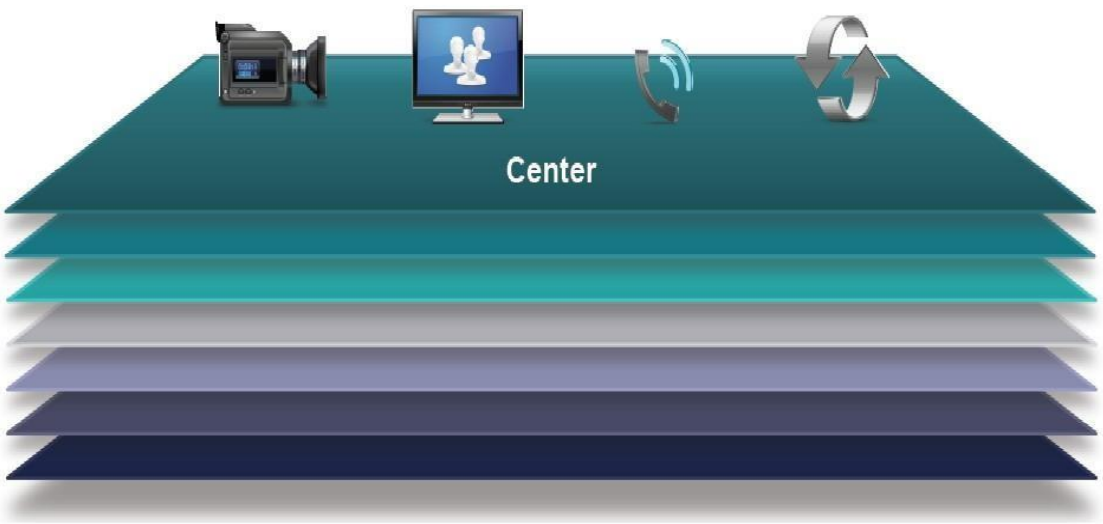


Level 7: Collaboration and Processes

One of the main distinctions between the Internet of Things (IoT) and IoT is that IoT includes people and processes. This difference becomes particularly clear at Level 7: Collaboration and Processes. The IoT system, and the information it creates, is of little value unless it yields action, which often requires people and processes. Applications execute business logic to empower people. People use applications and associated data for their specific needs. Often, multiple people use the same application for a range of different purposes. So the objective is not the application—it is to empower people to do their work better. Applications (Level 6) give business people the right data, at the right time, so they can do the right thing.

But frequently, the action needed requires more than one person. People must be able to communicate and collaborate, sometimes using the traditional Internet, to make the IoT useful. Communication and collaboration often requires multiple steps. And it usually transcends multiple applications. This is why Level 7, as shown in

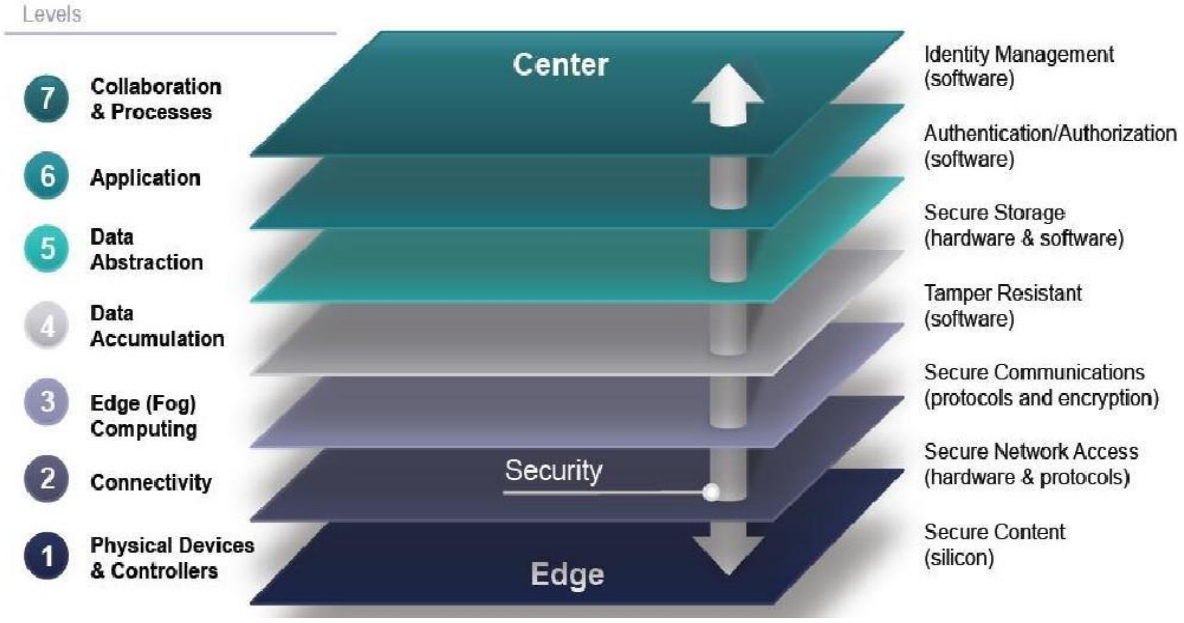




Security in the IoT

Discussions of security for each level and for the movement of data between levels could fill a multitude of papers. For the purpose of the IoT Reference Model, security measures must:

- Secure each device or system
- Provide security for all processes at each level
- Secure movement and communication between each level, whether north- or south-bound



The technologies highlighted here are the ones that are seen as having market and/or mind share. The following are the technologies for connecting smart objects:

IEEE 802.15.4: This is an older but foundational wireless protocol for connecting smart objects.

IEEE 802.15.4g and IEEE 802.15.4e: These are the result of improvements done to 802.15.4 and are mainly targeted to utilities and smart cities deployments.

IEEE 1901.2a: This is a technology for connecting smart objects over power lines.

IEEE 802.11ah: This is a technology built on the well-known 802.11 Wi-Fi standards that is specifically for smart objects.

LoRaWAN: This is a scalable technology designed for longer distances with low power requirements in the unlicensed spectrum.

NB-IoT and Other LTE Variations: These technologies are often the choice of mobile service providers looking to connect smart objects over longer distances in the licensed spectrum.

A common information set is provided about the IoT access technologies which are as listed below:

- **Standardization and alliances:** The standards bodies that maintain the protocols for a technology
- **Physical layer:** The wired or wireless methods and relevant frequencies
- **MAC layer:** Considerations at the Media Access Control (MAC) layer, which bridgesthe physical layer with data link control
- **Topology:** The topologies supported by the technology
- **Security:** Security aspects of the technology
- **Competitive technologies:** Other technologies that are similar and may be suitable alternatives to the given technology

IEEE 802.15.4

IEEE 802.15.4 is a wireless access technology for low-cost and low-data-rate devices that are powered or run on batteries. In addition to being low cost and offering a reasonable battery life, this access technology enables easy installation using a compact protocol stack while remaining both simple and flexible. Several network communication stacks, including deterministic ones, and profiles leverage this technology to address a wide range of IoT use cases in both the consumer and business markets.

IEEE 802.15.4 is commonly found in the following types of deployments:

- Home and building automation
- Automotive networks
- Industrial wireless sensor networks
- Interactive toys and remote controls

Criticisms of IEEE 802.15.4 often focus on its MAC reliability, unbounded latency, and susceptibility to interference and multipath fading.

The negatives around reliability and latency often have to do with the Collision Sense Multiple Access/Collision Avoidance (CSMA/CA) algorithm. CSMA/CA is an access method in which a device "listens" to make sure no other devices are transmitting before starting its own transmission. If another device is transmitting, a wait time (which is usually random) occurs before "listening" occurs again. Interference and multipath fading occur with IEEE 802.15.4 because it lacks a frequency-hopping technique. Later variants of 802.15.4 from the IEEE start to address these issues.

Standardization and Alliances:

IEEE 802.15.4 or IEEE 802.15 Task Group 4 defines low-data-rate PHY and MAC layer specifications for wireless personal area networks (WPAN). This standard has evolved over the years and is a well-known solution for low-complexity wireless devices with low data rates that need many months or even years of battery.

Since 2003, the IEEE has published several iterations of the IEEE 802.15.4 specification. Newer releases typically supersede older ones, integrate addendums, and add features or clarifications to previous versions.

The IEEE 802.15.4 PHY and MAC layers are the foundations for several networking protocol stacks. These protocol stacks make use of 802.15.4 at the physical and link layer levels, but the upper layers are different. These protocol stacks are promoted separately through various organizations and often commercialized. Some of the most well-known protocol stacks based on 802.15.4 are highlighted in Table.

S.No	Protocol	Description
1.	ZigBee	Promoted through the ZigBee alliance, ZigBee defines upper-layer components (network through application) as well as application profiles. Common profiles include building automation, home automation, and healthcare. ZigBee also defines device object functions such as device role, device discovery, network join and security
2.	6LoWPAN	6LoWPAN is an IPv6 adaptation layer defined by the IETF 6LoWPANworking group that describes how to transport IPv6 packets over IEEE 802.15.4 layers. RFCs document header compression and IPv6 enhancement to cope with the specific details of IEEE 802.15.4
3.	ZigBee IP	An evolution of the ZigBee protocol stack, ZigBee IP adopts the 6LoWPAN adaptation layer , IPv6 network layer, RPL routing protocol. In addition, it offers improvement in IP security.
4.	ISA100.11a	This is developed by the International Society of Automation (ISA) as "Wireless Systems for Industrial automation: Process Control and Related Applications".It is based on IEEE 802.15.4-2006. The network and transport layers are based on IETF 6LoWPAN, IPv6, and UDP standards
5.	Wireless HART	Wireless HART promoted by the HART Communication Foundation, is aprotocol stack that offers a time-synchronized, self-organizing, and self healing mesh architecture, leveraging IEEE 802.15.4-2006 over the 2.4GHz frequency band.
6.	Thread	Constructed on top of IETF 6LoWPAN /IPv6, Thread is a protocol stack for a secure and reliable mesh network to connect and control products in the home.

ZigBee

ZigBee is one of the most well-known protocols listed in Table 4-2. In addition, ZigBee has continued to evolve over time as evidenced by the release of Zigbee IP and is representativeof how IEEE 802.15.4 can be leveraged at the PHY and MAC layers, independent of the protocol layers above. The Zigbee Alliance is an industry group formed to certify interoperability between vendors and it is committed to driving and evolving ZigBee as an IoT solution for interconnecting smart objects. ZigBee solutions are aimed at smart objects and sensors that have low bandwidth and low power needs.

Furthermore, products that are ZigBee compliant and certified by the ZigBee Alliance should interoperate even though different vendors may manufacture them. The ZigBee specification has undergone several revisions.

The main areas where ZigBee is the most well-known include automation for commercial, retail, and home applications and smart energy. In the industrial and commercial automation space, ZigBee-based devices can handle various functions, from measuring temperature and humidity to tracking assets. For home automation, ZigBee can control lighting, thermostats, and security functions. ZigBee Smart Energy brings together a variety of interoperable products, such as smart meters, that can monitor and control the use and delivery of utilities, such as electricity and water. These ZigBee products are controlled by

the utility provider and can help coordinate usage between homes and businesses and the utility provider itself to provide more efficient operations.
The traditional ZigBee stack is illustrated in Figure. As mentioned previously, ZigBee utilizes the IEEE
802.15.4 Standard at the lower PHY and MAC layers. ZigBee specifies the network andsecurity layer and application support layer that sit on top of the lower layers.

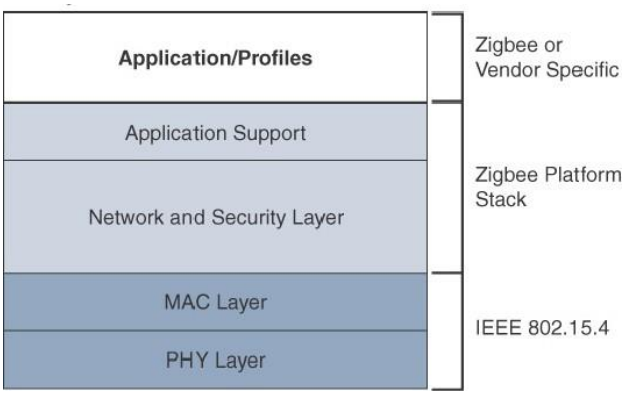


Figure. High Level ZigBee Protocol Stack

The ZigBee network and security layer provides mechanisms for network startup, configuration, routing, and securing communications. This includes calculating routing paths in what is often a changing topology, discovering neighbors, and managing the routing tables as devices join for the first time. The network layer is also responsible for forming the appropriate topology, which is often a mesh but could be a star or tree as well. From a security perspective, ZigBee utilizes 802.15.4 for security at the MAC layer, using the Advanced Encryption Standard (AES) with a 128-bit key and also provides security at the network and application layers.

The application support layer in Figure 2.1 interfaces the lower portion of the stack dealing with the networking of ZigBee devices with the higher-layer applications. ZigBee predefines many application profiles for certain industries, and vendors can optionally create their own custom ones at this layer.

ZigBee is one of the most well-known protocols built on an IEEE 802.15.4 foundation. On top of the 802.15.4 PHY and MAC layers, ZigBee specifies its own network and security layer and application profiles. While this structure has provided a fair degree of interoperability for vendors with membership in the ZigBee Alliance, it has not provided interoperability with other IoT solutions. However, this has started to change with the release of ZigBee IP

ZigBee IP

With the introduction of ZigBee IP, the support of IEEE 802.15.4 continues, but the IP and TCP/UDP protocols and various other open standards are now supported at the network and transport layers. The ZigBee-specific layers are now found only at the top of the protocol stack for the applications. ZigBee IP was created to embrace the open standards coming from the IETF's work on LLNs, such as IPv6, 6LoWPAN, and RPL. They

provide for low-bandwidth, low-power, and cost-effective communications when connecting smart objects.

ZigBee IP is a critical part of the Smart Energy (SE) Profile 2.0 specification from the ZigBee Alliance. SE 2.0 is aimed at smart metering and residential energy management systems. In fact, ZigBee IP was designed specifically for SE 2.0 but it is not limited to this use case. Any other applications that need a standards based IoT stack can utilize Zigbee IP. The ZigBee IP stack is shown in the following figure.

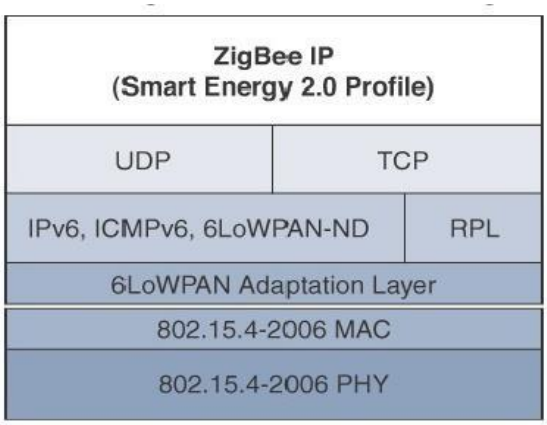


Figure : ZigBee IP protocol stack

Unlike traditional ZigBee, ZigBee IP supports 6LoWPAN as an adaptation layer. The 6LoWPAN mesh addressing header is not required as ZigBee IP utilizes the mesh-over or route-over method for forwarding packets.

ZigBee IP requires the support of 6LoWPAN's fragmentation and header compression schemes. At the network layer, all ZigBee IP nodes support IPv6, ICMPv6, and 6LoWPAN Neighbor Discovery (ND), and utilize RPL for the routing of packets across the mesh network. Both TCP and UDP are also supported, to provide both connection-oriented and connectionless service.

As you can see, ZigBee IP is a compelling protocol stack offering because it is based on current IoT standards at every layer under the application layer. This opens up opportunitiesfor ZigBee IP to integrate and interoperate on just about any 802.15.4 network with other solutions built on these open IoT standards.

Physical Layer

The 802.15.4 standard supports an extensive number of PHY options that range from 2.4 GHz to sub-GHz frequencies in ISM bands. The original IEEE 802.15.4-2003 standard specified only three PHY options based on direct sequence spread spectrum (DSSS) modulation. DSSS is a modulation technique in which a signal is intentionally spread in the frequency domain, resulting in greater bandwidth. The original physical layer transmission options were as follows:

- 2.4 GHz, 16 channels, with a data rate of 250 kbps
- 915 MHz, 10 channels, with a data rate of 40 kbps
- 868 MHz, 1 channel, with a data rate of 20 kbps

The 2.4 GHz band operates worldwide. The 915 MHz band operates mainly in North and South America, and the 868 MHz frequencies are used in Europe, the Middle East, and Africa. IEEE 802.15.4-2006, 802.15.4-2011, and IEEE 802.15.4-2015 introduced additional PHY communication options, including the following:

- **OQPSK PHY:** This is DSSS PHY, employing offset quadrature phase-shift keying (OQPSK) modulation. OQPSK is a modulation technique that uses four unique bit values that are signaled by phase changes. An offset function that is present during phase shifts allows data to be transmitted more reliably.
- **BPSK PHY:** This is DSSS PHY, employing binary phase-shift keying (BPSK) modulation. BPSK specifies two unique phase shifts as its data encoding scheme.
- **ASK PHY:** This is parallel sequence spread spectrum (PSSS) PHY, employing amplitude shift keying (ASK) and BPSK modulation. PSSS is an advanced encoding scheme that offers increased range, throughput, data rates, and signal integrity compared to DSSS. ASK uses amplitude shifts instead of phase shifts to signal different bit values.

These improvements increase the maximum data rate for both 868 MHz and 915 MHz to 100 kbps and 250 kbps, respectively. The 868 MHz support was enhanced to 3 channels, while other IEEE 802.15.4 study groups produced addendums for new frequency bands. For example, the IEEE 802.15.4c study group created the bands 314–316 MHz, 430–434 MHz, and 779–787 MHz for use in China.

Figure 2.3 shows the frame for the 802.15.4 physical layer. The synchronization header for this frame is composed of the Preamble and the Start of Frame Delimiter fields. The Preamble field is a 32-bit 4-byte (for parallel construction) pattern that identifies the start ofthe frame and is used to synchronize the data transmission. The Start of Frame Delimiter field informs the receiver that frame contents start immediately after this byte.

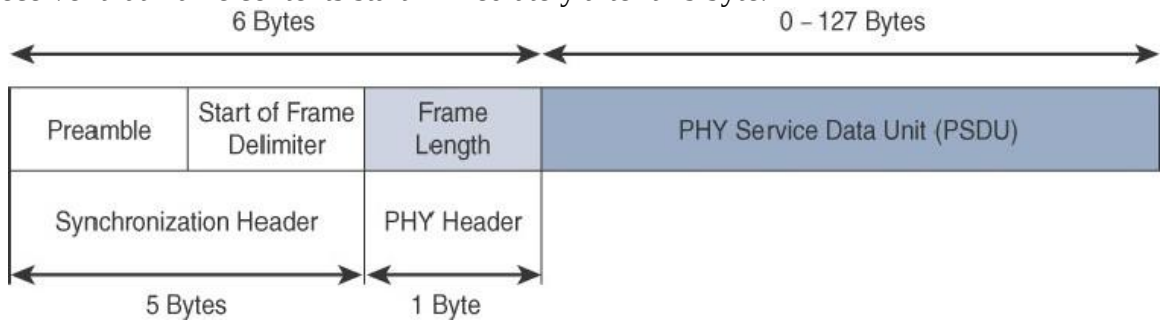


Figure 2.3 IEEE 802.15.4 PHY Format

The PHY Header portion of the PHY frame shown in Figure 2.3 is simply a frame length value. It lets the receiver know how much total data to expect in the PHY service data unit (PSDU) portion of the 802.4.15 PHY. The PSDU is the data field or payload.

MAC Layer

The IEEE 802.15.4 MAC layer manages access to the PHY channel by defining how devices in the same area will share the frequencies allocated. At this layer, the scheduling and routing of data frames are also coordinated. The 802.15.4 MAC layer performs the following tasks:

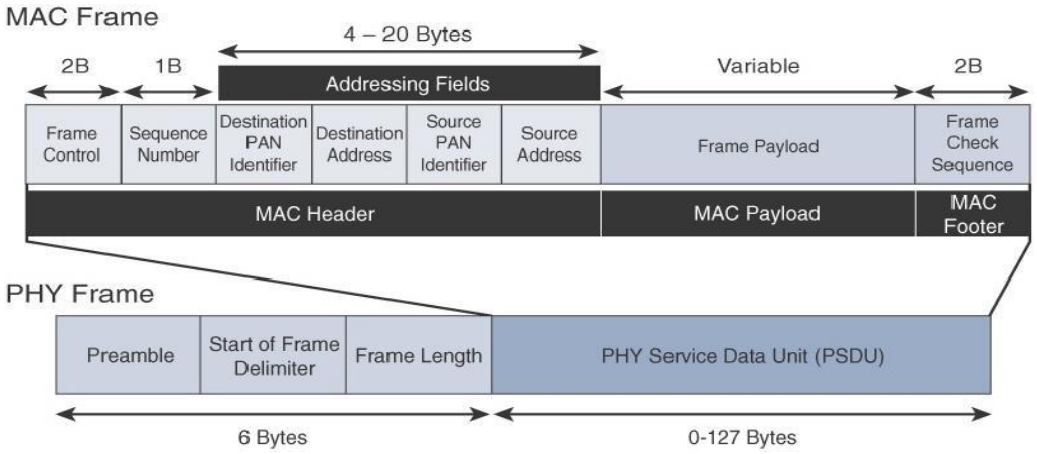
- Network beaconing for devices acting as coordinators (New devices use beacons to join an 802.15.4 network)

- PAN association and disassociation by a device
- Device security
- Reliable link communications between two peer MAC entities

The MAC layer achieves these tasks by using various predefined frame types. In fact, four types of MAC frames are specified in 802.15.4:

- **Data frame:** Handles all transfers of data
- **Beacon frame:** Used in the transmission of beacons from a PAN coordinator
- **Acknowledgement frame:** Confirms the successful reception of a frame
- **MAC command frame:** Responsible for control communication between devices

Each of these four 802.15.4 MAC frame types follows the frame format shown in Figure 2.4. In Figure 2.4, notice that the MAC frame is carried as the PHY payload. The 802.15.4 MACframe can be broken down into the MAC Header, MAC Payload, and MAC Footer



fields.

Figure 2.4 IEEE 802.15.4 MAC Format

The MAC Header field is composed of the Frame Control, Sequence Number and the Addressing fields. The Frame Control field defines attributes such as frame type, addressing modes, and other control flags. The Sequence Number field indicates the sequence identifier for the frame. The Addressing field specifies the Source and Destination PAN Identifier fields as well as the Source and Destination Address fields.

The MAC Payload field varies by individual frame type. For example, beacon frames have specific fields and payloads related to beacons, while MAC command frames have different fields present. The MAC Footer field is nothing more than a frame check sequence (FCS). An FCS is a calculation based on the data in the frame that is used by the receiving side to confirm the integrity of the data in the frame.

IEEE 802.15.4 requires all devices to support a unique 64-bit extended MAC address, based on EUI-64. However, because the maximum payload is 127 bytes, 802.15.4 alsodefines how a 16-bit “short address” is assigned to devices.

This short address is local to the PAN and substantially reduces the frame overhead compared to a 64-bit extended MAC address. However, you should be aware that the use of this short address might be limited to specific upper-layer protocol stacks.

Topology

IEEE 802.15.4–based networks can be built as star, peer-to-peer, or mesh topologies. Mesh networks tie together many nodes. This allows nodes that would be out of range if trying to communicate directly to leverage intermediary nodes to transfer communications. Please note that every 802.15.4 PAN should be set up with a unique ID. All the nodes in the same 802.15.4 network should use the same PAN ID. Figure 2.5 shows an example of an 802.15.4 mesh network with a PAN ID of 1.

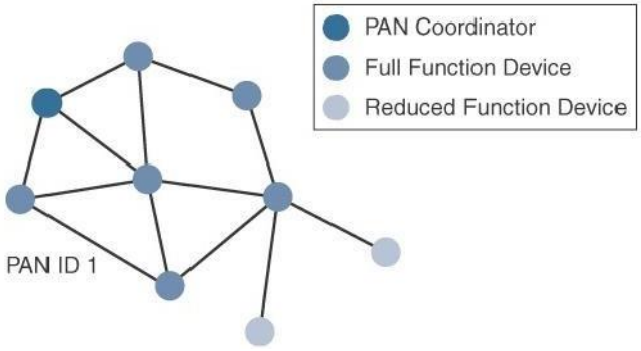


Figure 2.5: Sample Mesh Network Topology

A minimum of one FFD acting as a PAN coordinator is required to deliver services that allow other devices to associate and form a cell or PAN. Notice in Figure 2.5 that a single PAN coordinator is identified for PAN ID 1. FFD devices can communicate with any other devices, whereas RFD devices can communicate only with FFD devices.

The IEEE 802.15.4 specification does not define a path selection within the MAC layer for a mesh topology. This function can be done at Layer 2 and is known as *mesh-under*. Generally, this is based on a proprietary solution. Alternatively, the routing function can occur at Layer 3, using a routing protocol, such as the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL). This is referred to as *mesh-over*.

Security

The IEEE 802.15.4 specification uses Advanced Encryption Standard (AES) with a 128-bit key length as the base encryption algorithm for securing its data. Established by the US National Institute of Standards and Technology in 2001, AES is a block cipher, which means it operates on fixed-size blocks of data. The use of AES by the US government and its widespread adoption in the private sector has helped it become one of the most popular algorithms used in symmetric key cryptography. (A *symmetric key* means that the same key is used for both the encryption and decryption of the data.)

In addition to encrypting the data, AES in 802.15.4 also validates the data that is sent. This is accomplished by a message integrity code (MIC), which is calculated for the entire frame using the same AES key that is used for encryption.

Enabling these security features for 802.15.4 changes the frame format slightly and consumes some of the payload. Using the Security Enabled field in the Frame Controlportion of the 802.15.4 header is the first step to enabling AES encryption. This field is a single bit that is set to 1 for security. Once this bit is set, a field called the Auxiliary Security Header is created after the Source Address field, by stealing some bytes from the Payload field. Figure 2.6 shows the IEEE 802.15.4 frame format at a high level, with the Security Enabled bit set and the Auxiliary Security Header field present.

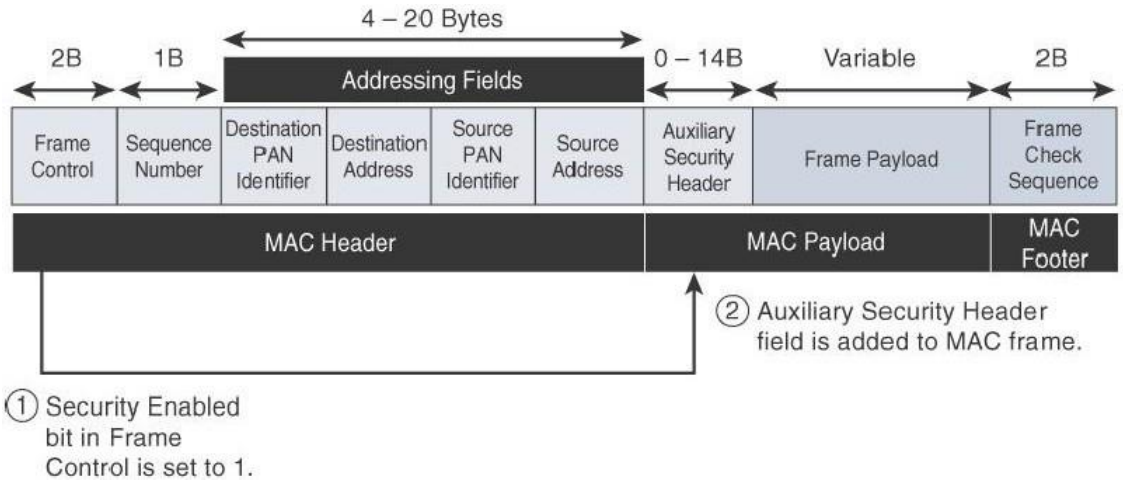


Figure 2.6: Frame format with the Auxiliary Security Header Field for 802.15.4-2006

Competitive Technologies

As detailed in Table 2.1, the IEEE 802.15.4 PHY and MAC layers are the foundations for several networking profiles that compete against each other in various IoT access environments. These various vendors and organizations build upper-layer protocol stacks on top of an 802.15.4 core. They compete and distinguish themselves based on features and capabilities in these upper layers.

A competitive radio technology that is different in its PHY and MAC layers is DASH7. DASH7 was originally based on the ISO18000-7 standard and positioned for industrial communications, whereas IEEE 802.15.4 is more generic. Commonly employed in active radio frequency identification (RFID) implementations, DASH7 was used by US military forces for many years, mainly for logistics purposes. Active RFID utilizes radio waves generated by a battery-powered tag on an object to enable continuous tracking.

The current DASH7 technology offers low power consumption, a compact protocol stack, range up to 1 mile, and AES encryption. Frequencies of 433 MHz, 868 MHz, and 915 MHz have been defined, enabling data rates up to 166.667 kbps and a maximum payload of 256 bytes. DASH7 is promoted by the DASH7 Alliance, which has evolved the protocol from its active RFID niche into a wireless sensor network technology that is aimed at the commercial market.

IEEE 802.15.4 Summary

The IEEE 802.15.4 wireless PHY and MAC layers are mature specifications that are the foundation for various industry standards and products as listed in Table 2.1. The PHY layer offers a maximum speed of up to 250 kbps, but this varies based on modulation and frequency. The MAC layer for 802.15.4 is robust and handles how data is transmitted and received over the PHY layer. Specifically, the MAC layer handles the association and disassociation of devices to/from a PAN, reliable communications between devices, security, and the formation of various topologies.

The topologies used in 802.15.4 include star, peer-to-peer, and cluster trees that allow for the formation of mesh networks. From a security perspective, 802.15.4 utilizes AES encryption to allow secure communications and also provide data integrity.

The main competitor to IEEE 802.15.4 is DASH7, another wireless technology that compares favorably. However, IEEE 802.15.4 has an edge in the marketplace through all the different vendors and organizations that utilize its PHY and MAC layers. As 802.15.4 continues to evolve, you will likely see broader adoption of the IPv6 standard at the network layer. For IoT sensor deployments requiring low power, low data rate, and low complexity, the IEEE 802.15.4 standard deserves strong consideration.

IEEE 802.11ah

In unconstrained networks, IEEE 802.11 Wi-Fi is certainly the most successfully deployed wireless technology. This standard is a key IoT wireless access technology, either for connecting endpoints such as fog computing nodes, high-data-rate sensors, and audio orvideo analytics devices or for deploying Wi-Fi backhaul infrastructures, such as outdoor Wi- Fi mesh in smart cities, oil and mining, or other environments.

However, Wi-Fi lacks sub-GHz support for better signal penetration, low power for battery-powered nodes, and the ability to support a large number of devices. For these reasons, the IEEE 802.11 working group launched a task group named IEEE 802.11ah to specify a sub-GHz version of Wi-Fi. Three main use cases are identified for IEEE 802.11ah:

- **Sensors and meters covering a smart grid:** Meter to pole, environmental/agricultural monitoring, industrial process sensors, indoor healthcare system and fitness sensors, home and building automation sensors
- **Backhaul aggregation of industrial sensors and meter data:** Potentially connecting IEEE 802.15.4g sub networks
- **Extended range Wi-Fi:** For outdoor extended-range hotspot or cellular traffic offloading when distances already covered by IEEE 802.11a/b/g/n/ac are not good enough

Standardization and Alliances

In July 2010, the IEEE 802.11 working group decided to work on an "industrial Wi-Fi"and created the IEEE 802.11ah group. The 802.11ah specification would operate in unlicensed sub-GHz frequency bands, similar to IEEE 802.15.4 and other LPWA technologies.The industry organization that promotes Wi-Fi certifications and interoperability for 2.4 GHz and 5 GHz products is the Wi-Fi Alliance. The Wi-Fi Alliance is a similar body to theWi-SUN Alliance.

For the 802.11ah standard, the Wi-Fi Alliance defined a new brand called Wi-Fi HaLow. The HaLow brand exclusively covers IEEE 802.11ah for sub-GHz device certification. You can think of Wi-Fi HaLow as a commercial designation for products incorporating IEEE 802.11ah technology.

Physical Layer

IEEE 802.11ah essentially provides an additional 802.11 physical layer operating inunlicensed sub-GHz
bands.

For example, various countries and regions use the following bands for IEEE 802.11ah: 868–868.6 MHz for EMEAR, 902–928 MHz and associated subsets for NorthAmerica and Asia-Pacific regions, and 314–316 MHz, 430–434 MHz, 470–510 MHz, and 779–787 MHz for China.

Based on OFDM modulation, IEEE 802.11ah uses channels of 2, 4, 8, or 16 MHz (and also 1 MHz for lowbandwidth transmission). This is one-tenth of the IEEE 802.11ac channels, resulting in one-tenth of the corresponding data rates of IEEE 802.11ac. The IEEE 802.11ac standard is a high-speed wireless LAN protocol at the 5 GHz band that is capable of speeds up to 1 Gbps.

While 802.11ah does not approach this transmission speed (as it uses one-tenth of 802.11ac channel width, it reaches one-tenth of 802.11ac speed), it does provide an extended range for its lower speed data. For example, at a data rate of 100 kbps, the outdoor transmission range for IEEE 802.11ah is expected to be 0.62 mile.

MAC Layer

The IEEE 802.11ah MAC layer is optimized to support the new sub-GHz Wi-Fi PHY while providing low power consumption and the ability to support a larger number of endpoints. Enhancements and features specified by IEEE 802.11ah for the MAC layer include the following:

- **Number of devices:** Has been scaled up to 8192 per access point.
- **MAC header:** Has been shortened to allow more efficient communication.
- **Null data packet (NDP) support:** Is extended to cover several control and management frames. Relevant information is concentrated in the PHY header and the additional overhead associated with decoding the MAC header and data payload is avoided. This change makes the control frame exchanges efficient and less power-consuming for the receiving stations.
- **Grouping and sectorization:** Enables an AP to use sector antennas and also group stations (distributing a group ID). In combination with RAW and TWT, this mechanism reduces contention in large cells with many clients by restricting which group, in which sector, can contend during which time window. (Sectors are described in more detail in the following section.)
- **Restricted access window (RAW):** Is a control algorithm that avoids simultaneous transmissions when many devices are present and provides fair access to the wireless network. By providing moreefficient access to the medium, additional power savings for battery-powered devices can be achieved, and collisions are reduced.
- **Target wake time (TWT):** Reduces energy consumption by permitting an accesspoint to define times when a device can access the network. This allows devices to enter a low-power state until their TWT time arrives. It also reduces the probability of collisions in large cells with many clients.
- **Speed frame exchange:** Enables an AP and endpoint to exchange frames during a reserved transmit opportunity (TXOP). This reduces contention on the medium, minimizes the number of frame exchanges to improve channel efficiency, and extends battery life by keeping awake times short.

The 802.11ah MAC layer is focused on power consumption and mechanisms to allow low-power Wi-Fi stations to wake up less often and operate more efficiently. This sort of MAC layer is ideal for IoT devices that often produce short, low-bit-rate transmissions.

Topology

While IEEE 802.11ah is deployed as a star topology, it includes a simple hops relay operation to extend its range. This relay option is not capped, but the IEEE 802.11ah task group worked on the assumption of two hops. It allows one 802.11ah device to act as an intermediary and relay data to another. In some ways, this is similar to a mesh, and it is important to note that the clients and not the access point handle the relay function.

This relay operation can be combined with a higher transmission rate or modulation and coding scheme (MCS). This means that a higher transmit rate is used by relay devices talking directly to the access point. The transmit rate reduces as you move further from the access point via relay clients. This ensures an efficient system that limits transmission speeds at the edge of the relays so that communications close to the AP are not negatively affected.

Sectorization is a technique that involves partitioning the coverage area into several sectors to get reduced contention within a certain sector. This technique is useful forlimiting collisions in cells that have many clients. This technique is also often necessary when the coverage area of 802.11ah access points is large, and interference from neighboring access points is problematic. Sectorization uses an antenna array and beam- forming techniques to partition the cell-coverage area. Figure 2.7 shows an example of 802.11ah sectorization.

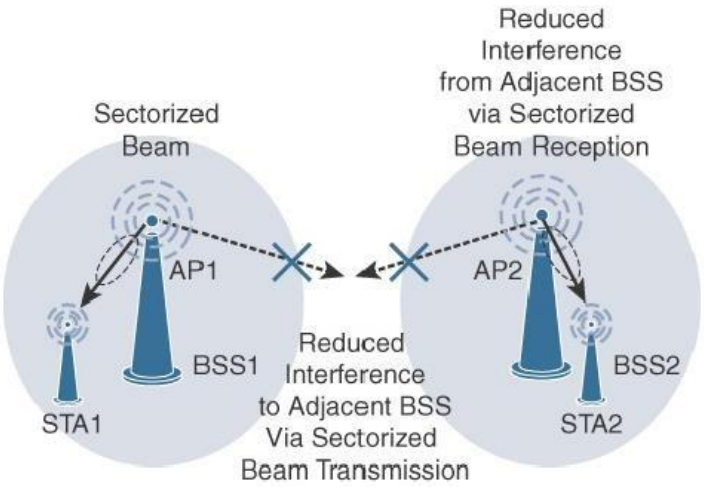


Figure 2.7: IEEE 802.11 ah Sectorization

Security

No additional security has been identified for IEEE 802.11ah compared to other IEEE 802.11 specifications.

Competitive Technologies

Competitive technologies to IEEE 802.11ah are IEEE 802.15.4 and IEEE 802.15.4e.

IEEE 802.11ah Conclusions

The IEEE 802.11ah access technology is an ongoing effort of the IEEE 802.11 working group to define an “industrial Wi-Fi.” Currently, this standard is just at the beginning of its evolution, and it is not clear how the market will react to this new Wi-Fi standard.

This specification offers a longer range than traditional Wi-Fi technologies and provides good support for low-power devices that need to send smaller bursts of data at lower speeds. At the same time, it has the ability to scale to higher speeds as well.

IEEE 802.11ah is quite different in terms of current products and the existing Wi-Fi technologies in the 2.4 GHz and 5 GHz frequency bands. To gain broad adoption and compete against similar technologies in this space, it will need an ecosystem of products and solutions that can be configured and deployed at a low cost.

LoRaWAN

In recent years, a new set of wireless technologies known as Low-Power Wide-Area (LPWA) has received a lot of attention from the industry and press. Particularly well adapted for long-range and battery-powered endpoints, LPWA technologies open new business opportunities to both services providers and enterprises considering IoT solutions.LoRaWAN is an unlicensed-band LPWA technology.

Standardization and Alliances

Initially, LoRa was a physical layer, or Layer 1, modulation that was developed by a French company named Cycleo. Later, Cycleo was acquired by Semtech. Optimized for long-range, two-way communications and low power consumption, the technology evolved from Layer 1 to a broader scope through the creation of the LoRa Alliance.

Semtech LoRa as a Layer 1 PHY modulation technology is available through multiple chipset vendors. To differentiate from the physical layer modulation known as LoRa, the LoRa Alliance uses the term LoRaWAN to refer to its architecture and its specifications that describe end-to-end LoRaWAN communications and protocols.

Figure 2.8 provides a high-level overview of the LoRaWAN layers. In this figure, notice that Semtech is responsible for the PHY layer, while the LoRa Alliance handles the MAC layer and regional frequency bands.

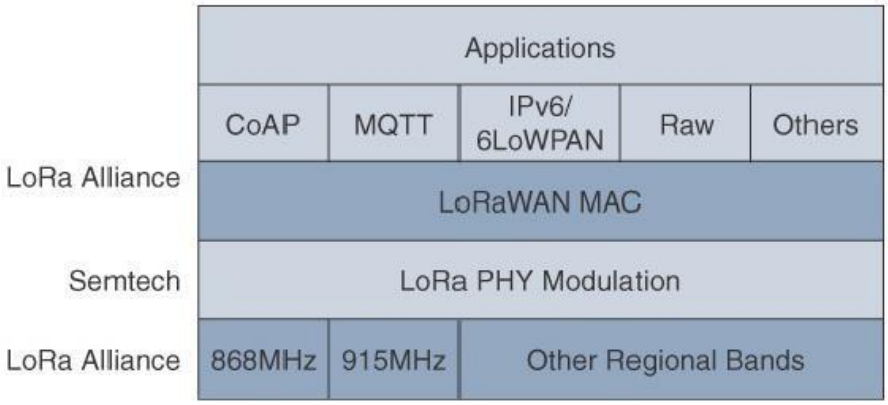


Figure 2.8 LoRa WAN Layers

Physical Layer

Semtech LoRa modulation is based on chirp spread spectrum modulation, which trades a lower data rate for receiver sensitivity to significantly increase the communication distance. In addition, it allows demodulation below the noise floor, offers robustness to noise and interference, and manages a single channel occupation by different spreading factors. This enables LoRa devices to receive on multiple channels in parallel.

LoRaWAN 1.0.2 regional specifications describe the use of the main unlicensed sub-GHz frequency bands of 433 MHz, 779–787 MHz, 863–870 MHz, and 902–928 MHz, as well as regional profiles for a subset of the 902–928 MHz bandwidth. For example, Australia utilizes 915–928 MHz frequency bands, while South Korea uses 920–923 MHz and Japan uses 920–928 MHz

Understanding LoRa gateways is critical to understanding a LoRaWAN system. A LoRagateway is deployed as the center hub of star network architecture. It uses multiple transceivers and channels and can demodulate multiple channels at once or even demodulate multiple signals on the same channel simultaneously. LoRa gateways serve as a transparent bridge relaying data between endpoints, and the endpoints use a single-hop wireless connection to communicate with one or many gateways.

The data rate in LoRaWAN varies depending on the frequency bands and adaptive data rate (ADR). ADR is an algorithm that manages the data rate and radio signal for each endpoint. The ADR algorithm ensures that packets are delivered at the best data ratepossible and that network performance is both optimal and scalable. Endpoints close to the gateways with good

signal values transmit with the highest data rate, which enables a shorter transmission time over the wireless network, and the lowest transmit power. Meanwhile, endpoints at the edge of the link budget communicate at the lowest data rate and highest transmit power.

An important feature of LoRa is its ability to handle various data rates via thespreading factor. Devices with a low spreading factor (SF) achieve less distance in their communications but transmit at faster speeds, resulting in less airtime. A higher SF provides slower transmission rates but achieves a higher reliability at longer distances. Table 2.2 illustrates how LoRaWAN data rates can vary depending on the associated spreading factor for the two main frequency bands, 863–870 MHz and 902–928 MHz.

Table 2.2 LoRa WAN Data Rate example

Configuration	863–870 MHz bps	902–928 MHz bps
LoRa: SF12/125 kHz	250	N/A
LoRa: SF11/125 kHz	440	N/A
LoRa: SF10/125 kHz	980	980
LoRa: SF9/125 kHz	1760	1760
LoRa: SF8/125 kHz	3125	3125
LoRa: SF7/125 kHz	5470	5470
LoRa: SF7/250 kHz	11,000	N/A
FSK: 50 kbps	50,000	N/A
LoRa: SF12/500 kHz	N/A	980
LoRa: SF11/500 kHz	N/A	1760
LoRa: SF10/500 kHz	N/A	3900
LoRa: SF9/500 kHz	N/A	7000
LoRa: SF8/500 kHz	N/A	12,500
LoRa: SF7/500 kHz	N/A	21,900

In Table 2.2, notice the relationship between SF and data rate. For example, at an SF value of 12 for 125 kHz of channel bandwidth, the data rate is 250 bps. However, when the SF is decreased to a value of 7, the data rate increases to 5470 bps. Channel bandwidth values of 125 kHz, 250 kHz, and 500 kHz are also evident in Table 2.2. The effect of increasing the bandwidth is that faster data rates can be achieved for the same spreading factor.

MAC Layer

The MAC layer is defined in the LoRaWAN specification. This layer takes advantage of the LoRa physical layer and classifies LoRaWAN endpoints to optimize their battery life and ensure downstream communications to the LoRaWAN endpoints. The LoRaWAN specification documents three classes of LoRaWAN devices:

- **Class A:** This class is the default implementation. Optimized for battery-powered nodes, it allows bidirectional communications, where a given node is able to receive downstream traffic after transmitting. Two receive windows are available after each transmission.
- **Class B:** This class was designated “experimental” in LoRaWAN 1.0.1 until it can be better defined. A Class B node or endpoint should get additional receive windows compared to Class A, but gateways must be synchronized through a beaconingprocess.
- **Class C:** This class is particularly adapted for powered nodes. This classification enables a node to be continuously listening by keeping it’s receive window open when

not transmitting.

LoRaWAN messages, either uplink or downlink, have a PHY payload composed of a 1-byte MAC header, a variable-byte MAC payload, and a MIC that is 4 bytes in length. The MAC payload size depends on the frequency band and the data rate, ranging from 59 to 230 bytes for the 863–870 MHz band and 19 to 250 bytes for the 902–928 MHz band. Figure 2.9 shows a high-level LoRaWAN MAC frame format.

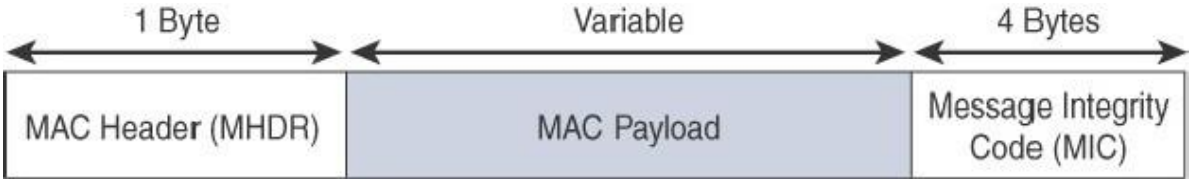


Figure 2.9 High- Level LoRaWAN MAC Frame Format

In version 1.0.x, LoRaWAN utilizes six MAC message types. LoRaWAN devices use join request and join accept messages for over-the-air (OTA) activation and joining the network. The other message types are unconfirmed data up/down and confirmed data up/down. A "confirmed" message is one that must be acknowledged, and "unconfirmed" signifies that the end device does not need to acknowledge.

"up/down" is simply a directional notation identifying whether the message flows in the uplink or downlink path. Uplink messages are sent from endpoints to the network server and are relayed by one or more LoRaWAN gateways. Downlink messages flow from the network server to a single endpoint and are relayed by only a single gateway. Multicast over LoRaWAN is being considered for future versions.

LoRaWAN endpoints are uniquely addressable through a variety of methods, including the following:

- An endpoint can have a global end device ID or DevEUI represented as an IEEE EUI-64 address.
- An endpoint can have a global application ID or AppEUI represented as an IEEE EUI-64 address that uniquely identifies the application provider, such as the owner, ofthe end device.
- In a LoRaWAN network, endpoints are also known by their end device address, known as a DevAddr, a 32-bit address. The 7 most significant bits are the network identifier (NwkID), which identifies the LoRaWAN network. The 25 least significant bits are used as the network address (NwkAddr) to identify the endpoint in the network.

Topology

LoRaWAN topology is often described as a "star of stars" topology. As shown in Figure 2.10, the infrastructure consists of endpoints exchanging packets through gateways acting as bridges, with a central LoRaWAN network server. Gateways connect to the backend network using standard IP connections, and endpoints communicate directly with one or more gateways.

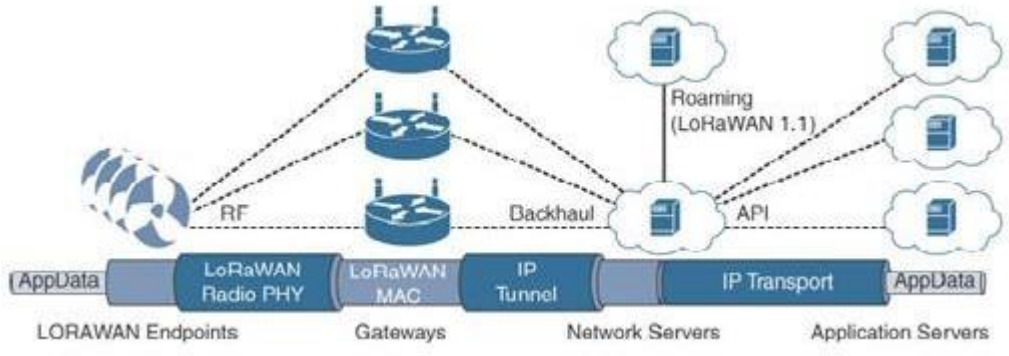


Figure 2.10 LoRa WAN Architecture

In Figure 2.10, LoRaWAN endpoints transport their selected application data over the LoRaWAN MAC layer on top of one of the supported PHY layer frequency bands. The application data is contained in upper protocol layers. These upper layers are not the responsibility of the LoRa Alliance, but best practices may be developed and recommended. These upper layers could just be raw data on top of the LoRaWAN MAC layer, or the data could be stacked in multiple protocols. For example, you could have upper-layer protocols, such as ZigBee Control Layer (ZCL), Constrained Application Protocol (CoAP), or Message Queuing Telemetry Transport (MQTT), with or without an IPv6/6LoWPAN layer.

Figure 2.10 also shows how LoRaWAN gateways act as bridges that relay between endpoints and the network servers. Multiple gateways can receive and transport the same packets. When duplicate packets are received, de-duplication is a function of the network server.

The LoRaWAN network server manages the data rate and radio frequency (RF) of each endpoint through the adaptive data rate (ADR) algorithm. ADR is a key component of the network scalability, performance, and battery life of the endpoints. The LoRaWANnetwork server forwards application data to the application servers, as depicted in Figure 2.10.

In future versions of the LoRaWAN specification, roaming capabilities between LoRaWAN network servers will be added. These capabilities will enable mobile endpoints to connect and roam between different LoRaWAN network infrastructures.

Security

Security in a LoRaWAN deployment applies to different components of the architecture, as detailed in Figure 2.11. LoRaWAN endpoints must implement two layers of security, protecting communications and data privacy across the network.

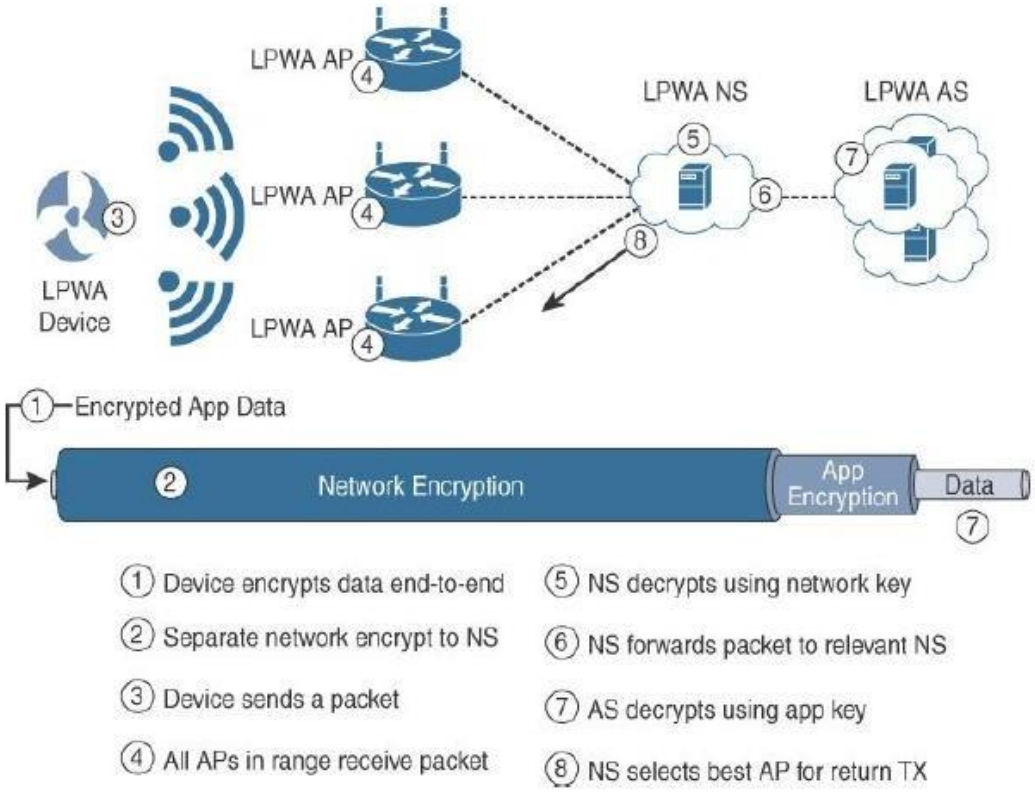


Figure 2.11 LoRaWAN Security

- ✓ The first layer, called "network security" but applied at the MAC layer, guarantees the authentication of the endpoints by the LoRaWAN network server. Also, it protects LoRaWAN packets by performing encryption based on AES. Each endpoint implements a network session key (NwkSKey), used by both itself and the LoRaWAN network server. The NwkSKey ensures data integrity through computing and checking the MIC of every data message as well as encrypting and decrypting MAC- only data message payloads.
- ✓ The second layer is an application session key (AppSKey), which performs encryption and decryption functions between the endpoint and its application server. Furthermore, it computes and checks the application-level MIC, if included. This ensures that the LoRaWAN service provider does not have access to the application payload if it is not allowed that access. Endpoints receive their AES-128 application key (AppKey) from the application owner. This key is most likely derived from an application-specific root key exclusively known to and under the control of the application provider.

For production deployments, it is expected that the LoRaWAN gateways are protected as well, for both the LoRaWAN traffic and the network management and operations over their backhaul link(s). This can be done using traditional VPN and IPSec technologies that demonstrate scaling in traditional IT deployments.

Additional security add-ons are under evaluation by the LoRaWAN Alliance for future revisions of the specification. LoRaWAN endpoints attached to a LoRaWAN network must get registered and authenticated. This can be achieved through one of the two join mechanisms:

- **Activation by personalization (ABP):** Endpoints don't need to run a join procedure as their individual details, including DevAddr and the NwkSKey and AppSKey session keys, are preconfigured and stored in the end device. This same information isregistered in the LoRaWAN network server.
- **Over-the-air activation (OTAA):** Endpoints are allowed to dynamically join a

particular LoRaWAN network after successfully going through a join procedure. The join procedure must be done every time a session context is renewed. During the join process, which involves the sending and receiving of MAC layer join request and join accept messages, the node establishes its credentials with a LoRaWAN network server, exchanging its globally unique DevEUI, AppEUI, and AppKey. The AppKey is then used to derive the session NwkSKey and AppSKey keys.

Competitive Technologies

LPWA solutions and technologies are split between unlicensed and licensed bands. The licensed-band technologies are dedicated to mobile service providers that have acquired spectrum licenses. In addition, several technologies are targeting the unlicensed- band LPWA market to compete against LoRaWAN. The LPWA market is quickly evolving. Table 2.3 evaluates two of the best-established vendors known to provide LPWA options.

Table 2.3 Unlicensed LPWA Technology Comparison

Characteristic	LoRaWAN	Sigfox	Ingenu Onramp
Frequency bands	433 MHz, 868 MHz, 902–928 MHz	433 MHz, 868 MHz, 902–928 MHz	2.4 GHz
Modulation	Chirp spread spectrum	Ultra-narrowband	DSSS
Topology	Star of stars	Star	Star; tree supported with an RPMA extender
Data rate	250 bps–50 kbps (868 MHz) 980 bps–21.9 kbps (915 MHz)	100 bps (868 MHz) 600 bps (915 MHz)	6 kbps
Adaptive data rate	Yes	No	No
Payload	59–230 bytes (868 MHz) 19–250 bytes (915 MHz)	12 bytes	6 bytes–10 KB
Two-way communications	Yes	Partial	Yes
Geolocation	Yes (LoRa GW version 2 reference design)	No	No
Roaming	Yes (LoRaWAN 1.1)	No	Yes
Specifications	LoRA Alliance	Proprietary	Proprietary

Constrained Nodes

In IoT solutions, different classes of devices coexist. Depending on its functions in a network, "thing" architecture may or may not offer similar characteristics compared to a generic PC or server in an IT environment.

Another limit is that this network protocol stack on an IoT node may be required to communicate through an unreliable path. Even if a full IP stack is available on the node, this causes problems such as limited or unpredictable throughput and low convergence when a topology change occurs.

Finally, power consumption is a key characteristic of constrained nodes. Many IoT devices are battery powered, with lifetime battery requirements varying from a few months to 10+ years. This drives the selection of networking technologies since high-speed ones, such as Ethernet, Wi-Fi, and cellular, are not (yet) capable of multi-year battery life. Current capabilities practically allow less than a year for these technologies on battery-powered nodes. Of course, power consumption is much less of a concern on nodes that do not require batteries as an energy source.

The power consumption requirements on battery-powered nodes impact communication intervals. To help extend battery life, one could enable a "low-power" mode instead of one that is "always on." Another option is "always off," which means communications are enabled only when needed to send data.

While it has been largely demonstrated that production IP stacks perform well in constrained nodes. IoT constrained nodes can be classified as follows:

- **Devices that are very constrained in resources, may communicate infrequently to transmit a few bytes, and may have limited security and management capabilities:** This drives the need for the IP adaptation model, where nodes communicate through gateways and proxies.
- **Devices with enough power and capacities to implement a stripped-down IP stack or non- IP stack:** In this case, you may implement either an optimized IP stack and directly communicate with application servers (adoption model) or go for an IP or non-IP stack and communicate through gateways and proxies (adaptation model).
- **Devices that are similar to generic PCs in terms of computing and power resources but have constrained networking capacities, such as bandwidth:** These nodes usually implement a full IP stack (adoption model), but network design and application behaviors must cope with the bandwidth constraints.

The definition of constrained nodes is evolving. The costs of computing power, memory, storage resources, and power consumption are generally decreasing. At the same time, networking technologies continue to improve and offer more bandwidth and reliability. In the future, the push to optimize IP for constrained nodes will lessen as technology improvements and cost decreases address many of these challenges.

Constrained Networks

In the early years of the Internet, network bandwidth capacity was restrained due to technical limitations. Connections often depended on low-speed modems for transferring data. However, these low-speed connections demonstrated that IP could run over low- bandwidth networks.

But today, the evolution of networking has seen the emergence of high-speed infrastructures. However, high-speed connections are not usable by some IoT devices in the last mile. The reasons include the implementation of technologies with low bandwidth, limited distance and bandwidth due to regulated transmit power, and lack of or limited network services.

When link layer characteristics that we take for granted are not present, the networkis constrained. A constrained network can have high latency and a high potential for packet loss. Constrained networks have unique characteristics and requirements. In contrast with typical IP networks, where highly stable and fast links are available, constrained networks are limited by low-power, low bandwidth links (wireless and wired). They operate betweena few kbps and a few hundred kbps and may utilize a star, mesh, or combined network topologies, ensuring proper operations.

With a constrained network, in addition to limited bandwidth, it is not unusual for the packet delivery rate (PDR) to oscillate between low and high percentages. Large bursts of unpredictable errors and even loss of connectivity at times may occur. These behaviours can be observed on both wireless and narrowband power-line communication links, where packet delivery variation may fluctuate greatly during the course of a day.

Unstable link layer environments create other challenges in terms of latency and control plane reactivity. One of the golden rules in a constrained network is to “underreact to failure.” Due to the low bandwidth, a constrained network that overreacts can lead to a network collapse—which makes the existing problem worse.

Control plane traffic must also be kept at a minimum; otherwise, it consumes the bandwidth that is needed by the data traffic. Finally, one has to consider the power consumption in battery-powered nodes.Any failure or verbose control plane protocol may reduce the lifetime of the batteries.

To summarize, constrained nodes and networks pose major challenges for IoT connectivity in the last mile. This in turn has led various standards organizations to work on optimizing protocols for IoT.

IP Versions

For 20+ years, the IETF has been working on transitioning the Internet from IP version 4 to IP version 6. The main driving force has been the lack of address space in IPv4 as the Internet has grown. IPv6 has a much larger range of addresses that should not be exhausted for the foreseeable future. Today, both versions of IP run over the Internet, but most traffic is still IPv4 based.

While it may seem natural to base all IoT deployments on IPv6, you must take into account current infrastructures and their associated lifecycle of solutions, protocols, and products. IPv4 is entrenched in these current infrastructures, and so support for it is required in most cases. Therefore, the Internet of Things has to follow a similar path as the Internet itself and support both IPv4 and IPv6 versions concurrently.

Techniques such as tunnelling and translation need to be employed in IoT solutions to ensure interoperability between IPv4 and IPv6. A variety of factors dictate whether IPv4, IPv6, or both can be used in an IoT solution. Most often these factors include a legacy protocol or technology that supports only IPv4. Newer technologies and protocols almost always support both IP versions. The following are some of the main factors applicable to IPv4 and IPv6 support in an IoT solution:

- **Application Protocol:** IoT devices implementing Ethernet or Wi-Fi interfaces can communicate over both IPv4 and IPv6, but the application protocol may dictate the choice of the IP version. For example, SCADA protocols such as DNP3/IP (IEEE 1815), Modbus TCP, or the IEC 60870-5-104 standards are specified only for IPv4. So, there are no known production implementations by vendors of these protocols over IPv6 today. For IoT devices with application protocols defined by the IETF, such as HTTP/HTTPS, CoAP, MQTT, and XMPP, both IP versions are supported. The selection of the IP version is only dependent on the implementation.
- **Cellular Provider and Technology:** IoT devices with cellular modems are dependent on the generation of the cellular technology as well as the data services offered by the provider. For the first three generations of data services—GPRS, Edge, and 3G— IPv4 is the base protocol version. Consequently, if IPv6 is used with these generations, it must be tunneled over IPv4. On 4G/LTE networks, data services can use IPv4 or IPv6 as a base protocol, depending on the provider.

- **Serial Communications:** Many legacy devices in certain industries, such as manufacturing and utilities, communicate through serial lines. Data is transferred using either proprietary or standards based protocols, such as DNP3, Modbus, or IEC 60870-5-101. In the past, communicating this serial data over any sort of distance could be handled by an analog modem connection. However, as service provider support for analog line services has declined, the solution for communicating with these legacy devices has been to use local connections. To make this work, you connect the serial port of the legacy device to a nearby serial port on a piece of communications equipment, typically a router. This local router then forwards the serial traffic over IP to the central server for processing. Encapsulation of serial protocols over IP leverages mechanisms such as raw socket TCP or UDP. While raw socket sessions can run over both IPv4 and IPv6, current implementations are mostlyavailable for IPv4 only.
- **IPv6 Adaptation Layer:** IPv6-only adaptation layers for some physical and data link layers for recently standardized IoT protocols support only IPv6. While the most common physical and data link layers (Ethernet, Wi-Fi, and so on) stipulate adaptation layers for both versions, newer technologies, such as IEEE 802.15.4 (Wireless Personal Area Network), IEEE 1901.2, and ITU G.9903 (Narrowband Power Line Communications) only have an IPv6 adaptation layer specified. This means that any device implementing a technology that requires an IPv6 adaptation layer must communicate over an IPv6-only sub network. This is reinforced by the IETF routing protocol for LLNs, RPL, which is IPv6 only.

6LoWPAN

While the Internet Protocol is key for a successful Internet of Things, constrained nodes and constrained networks mandate optimization at various layers and on multiple protocols of the IP architecture. Some optimizations are already available from the market or under development by the IETF. Figure 2.12 highlights the TCP/IP layers where optimization is applied.

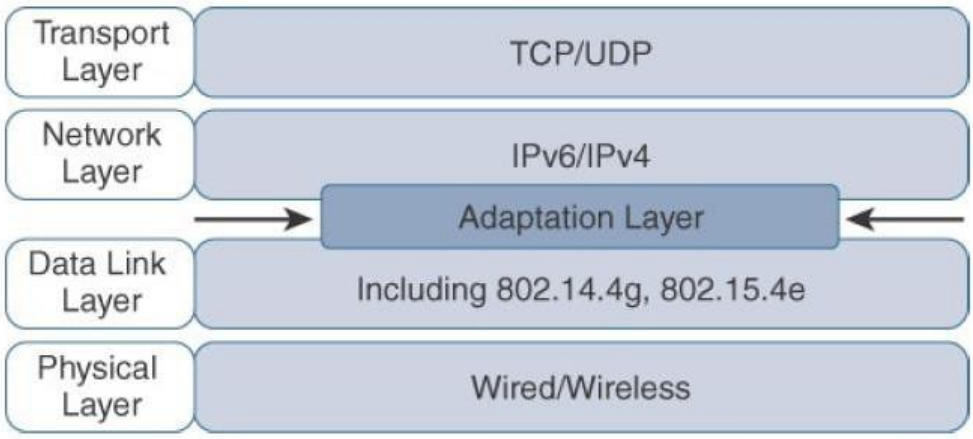


Figure 2.12: Optimizing IP for IoT Using an Adaptation Layer

In the IP architecture, the transport of IP packets over any given Layer 1 (PHY) and Layer 2 (MAC) protocol must be defined and documented. The model for packaging IP into lower-layer protocols is often referred to as an *adaptation layer*.

Unless the technology is proprietary, IP adaptation layers are typically defined by an IETF working group and released as a Request for Comments (RFC). An RFC is a publication from the IETF that officially documents Internet standards, specifications, protocols, procedures, and events. For example, RFC 864 describes how an IPv4 packet gets encapsulated over an Ethernet frame, and RFC 2464 describes how the same function is performed for an IPv6 packet.

IoT-related protocols follow a similar process. The main difference is that an adaptation layer designed for IoT may include some optimizations to deal with constrained nodes and networks. The main examples of adaptation layers optimized for constrained nodes or "things" are the ones under the 6LoWPAN working group and its successor, the 6Lo working group.

The initial focus of the 6LoWPAN working group was to optimize the transmission of IPv6 packets over constrained networks such as IEEE 802.15.4. Figure 2.13 shows an example of an IoT protocol stack using the 6LoWPAN adaptation layer beside the well- known IP protocol stack for reference.

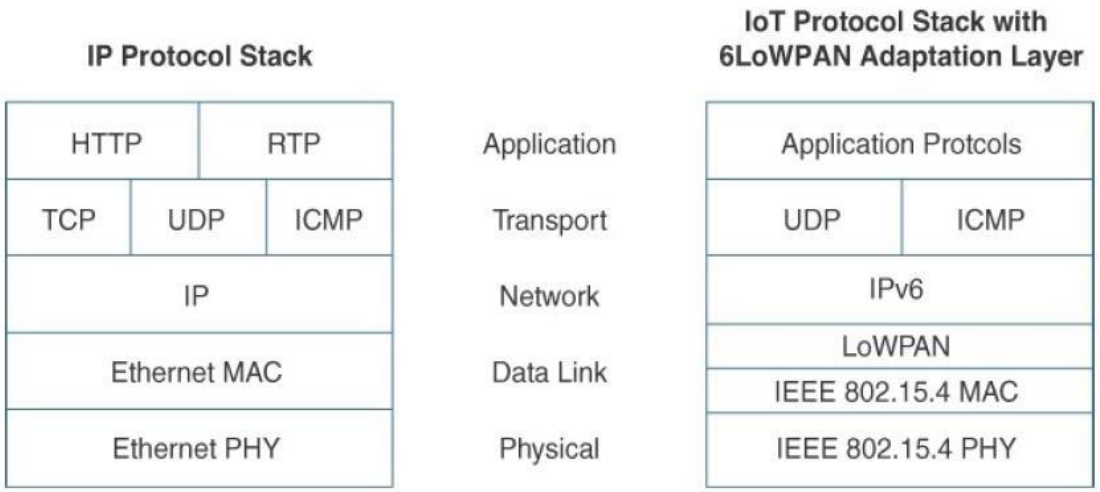


Figure 2.13: Comparison of an IoT Protocol Stack Utilizing 6LoWPAN and an IP Protocol Stack

The 6LoWPAN working group published several RFCs, but RFC 4994 is foundational because it defines frame headers for the capabilities of header compression, fragmentation,and mesh addressing. These headers can be stacked in the adaptation layer to keep these concepts separate while enforcing a structured method for expressing each capability. Depending on the implementation, all, none, or any combination of these capabilities and their corresponding headers can be enabled. Figure 2.14 shows some examples of typical 6LoWPAN header stacks.

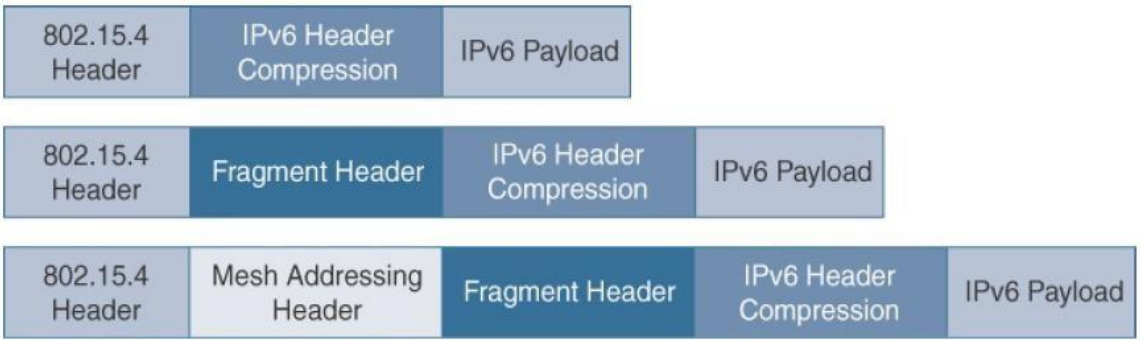


Figure 2.14 6LoWPAN Header Stack

Header Compression

IPv6 header compression for 6LoWPAN was defined initially in RFC 4944 and subsequently updated by RFC 6282. This capability shrinks the size of IPv6's 40-byte headers and User Datagram Protocol's (UDP's) 8-byte headers down as low as 6 bytes combined in some cases. Note that header compression for 6LoWPAN is only defined for an IPv6 header and not IPv4.

The 6LoWPAN protocol does not support IPv4, and, in fact, there is no standardized IPv4 adaptation layer for IEEE 802.15.4. 6LoWPAN header compression is stateless, and conceptually it is not too complicated. However, a number of factors affect the amount of compression, such as implementation of RFC 4944 versus RFC 6922, whether UDP is included, and various IPv6 addressing scenarios.

At a high level, 6LoWPAN works by taking advantage of shared information known by all nodes from their participation in the local network. In addition, it omits some standard header fields by assuming commonly used values. Figure 2.15 highlights an example that shows the amount of reduction that is possible with 6LoWPAN header compression.

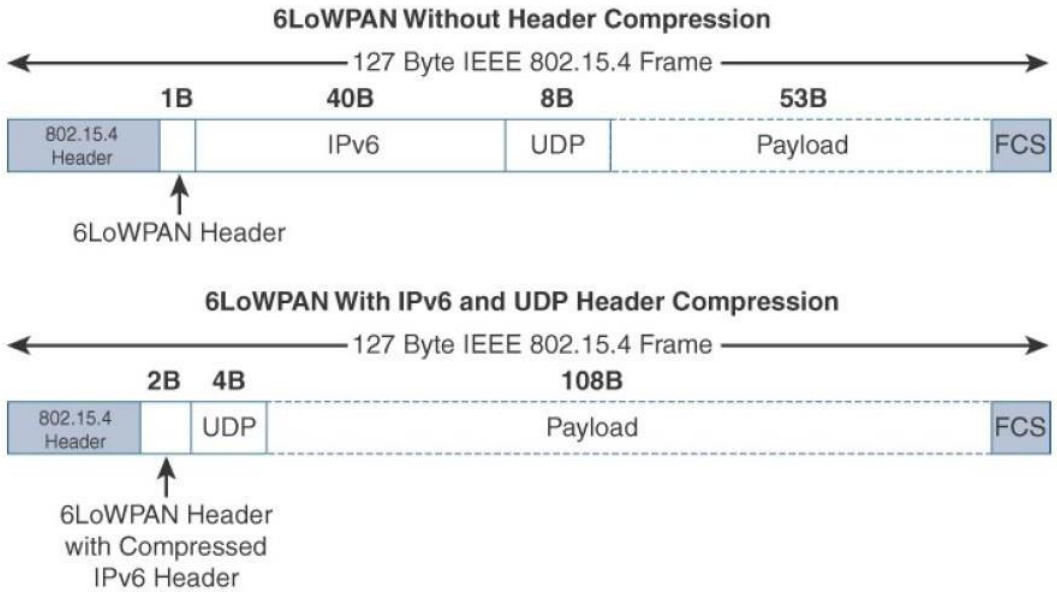


Figure 2.15 6LoWPAN Header Compression

At the top of Figure 2.15, you see a 6LoWPAN frame without any header compression enabled: The full 40- byte IPv6 header and 8-byte UDP header are visible. The 6LoWPAN header is only a single byte in this case. Notice that uncompressed IPv6 and UDP headers leave only 53 bytes of data payload out of the 127- byte maximum frame size in thecase of IEEE 802.15.4.

The bottom half of Figure 2.15 shows a frame where header compression has been enabled for a best-case scenario. The 6LoWPAN header increases to 2 bytes to accommodate the compressed IPv6 header, and UDP has been reduced in half, to 4 bytes from 8. Most importantly, the header compression has allowed the payload to more than double, from 53 bytes to 108 bytes, which is obviously much more efficient. Note that the 2-byte header compression applies to intra-cell communications, while communications external to the cell may require some field of the header to not be compressed.

Fragmentation

The maximum transmission unit (MTU) for an IPv6 network must be at least 1280 bytes. The term *MTU* defines the size of the largest protocol data unit that can be passed. For IEEE 802.15.4, 127 bytes is the MTU. This is a problem because IPv6, with a much larger MTU, is carried inside the 802.15.4 frame with a much smaller one. To remedy this situation, large IPv6 packets must be fragmented across multiple 802.15.4 frames at Layer 2.

The fragment header utilized by 6LoWPAN is composed of three primary fields: Datagram Size, Datagram Tag, and Datagram Offset. The 1-byte Datagram Size field specifies the total size of the unfragmented payload. Datagram Tag identifies the set of fragments for a payload. Finally, the Datagram Offset field delineates how far into a payload a particular fragment occurs. Figure 2.16 provides an overview of a 6LoWPAN fragmentation header.

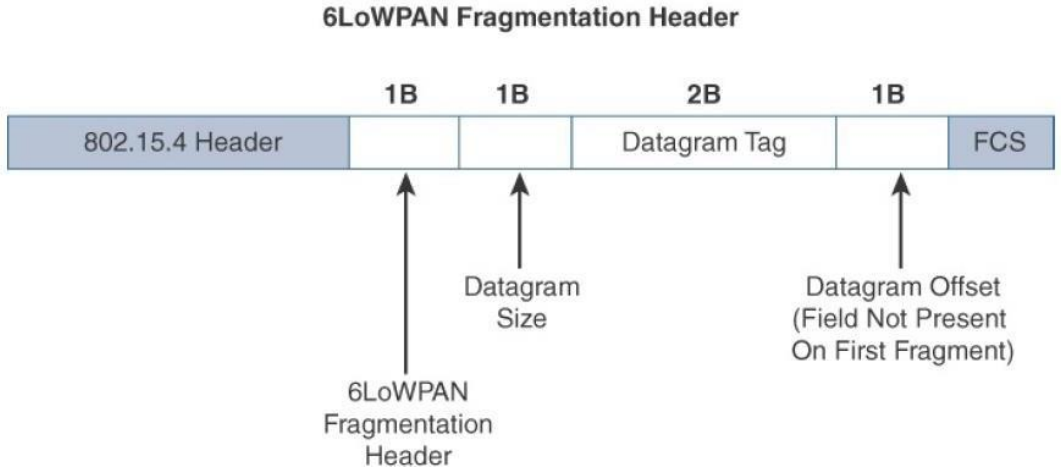


Figure 2.16 6LoWPAN Fragmentation Header

In Figure 2.16, the 6LoWPAN fragmentation header field itself uses a unique bit value to identify that the subsequent fields behind it are fragment fields as opposed to another capability, such as header compression. Also, in the first fragment, the Datagram Offset field is not present because it would simply be set to 0. This results in the first fragmentation header for an IPv6 payload being only 4 bytes long. The remainder of the fragments have a 5-byte header field so that the appropriate offset can be specified.

Mesh Addressing

The purpose of the 6LoWPAN mesh addressing function is to forward packets over multiple hops. Three fields are defined for this header: Hop Limit, Source Address, and Destination Address. Analogous to the IPv6 hop limit field, the hop limit for mesh addressing also provides an upper limit on how many times the frame can be forwarded. Each hop decrements this value by 1 as it is forwarded. Once the value hits 0, it is dropped and no longer forwarded.

The Source Address and Destination Address fields for mesh addressing are IEEE 802.15.4 addresses indicating the endpoints of an IP hop. Figure 2.17 details the 6LoWPAN mesh addressing header fields.

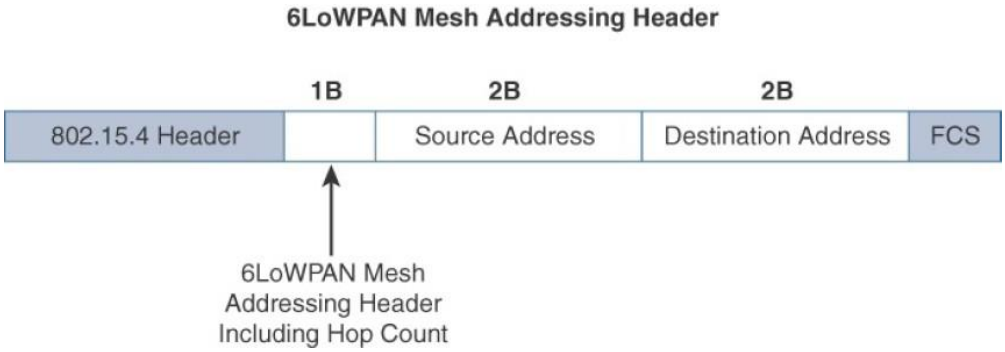


Figure 2.17: 6LoWPAN Mesh Addressing Header

Note that the mesh addressing header is used in a single IP subnet and is a Layer 2 type of routing known as mesh-under. RFC 4944 only provisions the function in this case as the definition of Layer 2 mesh routing specifications was outside the scope of the 6LoWPAN working group, and the IETF doesn't define "Layer 2 routing." An implementation performing Layer 3 IP routing does not need to implement a mesh addressing header unless required by a given technology profile.

Application Transport methodsSCADA

In the world of networking technologies and protocols, IoT is relatively new. Combined with the fact that IP is the de facto standard for computer networking in general, older protocols that connected sensors and actuators have evolved and adapted themselvesto utilize IP.

A prime example of this evolution is supervisory control and data acquisition (SCADA). Designed decades ago, SCADA is an automation control system that was initially implemented without IP over serial links, before being adapted to Ethernet and IPv4.

A Little Background on SCADA

For many years, vertical industries have developed communication protocols that fit their specific requirements. Many of them were defined and implemented when the most common networking technologies were serial link-based, such as RS-232 and RS-485. This led to SCADA networking protocols, which were well structured, compared to the other protocols, running directly over serial physical and data link layers.

At a high level, SCADA systems collect sensor data and telemetry from remotedevices, while also providing the ability to control them. Used in today's networks, SCADA systems allow global, real-time, data-driven decisions to be made about how to improve business processes.

SCADA networks can be found across various industries, but you find SCADA mainly concentrated in the utilities and manufacturing/industrial verticals. Within these specific industries, SCADA commonly uses certain protocols for communications between devices and applications. For example, Modbus and its variants are industrial protocols used to

monitor and program remote devices via a master/slave relationship. Modbus is also found in building management, transportation, and energy applications. The DNP3 and International Electrotechnical Commission (IEC) 60870-5-101 protocols are found mainly in the utilities industry, along with DLMS/COSEM and ANSI C12 for advanced meter reading (AMR).

As mentioned previously, these protocols go back decades and are serial based. So, transporting them over current IoT and traditional networks requires that certain accommodations be made from both protocol and implementation perspectives. These accommodations and other adjustments form various SCADA transport methods.

Adapting SCADA for IP

In the 1990s, the rapid adoption of Ethernet networks in the industrial world drove the evolution of SCADA application layer protocols. For example, the IEC adopted the Open System Interconnection (OSI) layer model to define its protocol framework. Other protocol user groups also slightly modified their protocols to run over an IP infrastructure. Benefits of this move to Ethernet and IP include the ability to leverage existing equipment and standards while integrating seamlessly the SCADA sub networks to the corporate WAN infrastructures.

To further facilitate the support of legacy industrial protocols over IP networks, protocol specifications were updated and published, documenting the use of IP for each protocol. This included assigning TCP/UDP port numbers to the protocols, such as the following:

- DNP3 (adopted by IEEE 1815-2012) specifies the use of TCP or UDP on port 20000 fortransporting
- DNP3 messages over IP.
- The Modbus messaging service utilizes TCP port 502.
- IEC 60870-5-104 is the evolution of IEC 60870-5-101 serial for running over Ethernet and IPv4 using port 2404.
- DLMS User Association specified a communication profile based on TCP/IP in the DLMS/COSEM Green Book (Edition 5 or higher), or in the IEC 62056-53 and IEC 62056-47 standards, allowing data exchange via IP and port 4059.

Like many of the other SCADA protocols, DNP3 is based on a master/slave relationship. The term *master* in this case refers to what is typically a powerful computer located in the control center of a utility, and a *slave* is a remote device with computing resources found in a location such as a substation. DNP3 refers to slaves specifically as *outstations*. Outstations monitor and collect data from devices that indicate their state, suchas whether a circuit breaker is on or off, and take measurements, including voltage, current, temperature, and so on. This data is then transmitted to the master when it is requested, or events and alarms can be sent in an asynchronous manner. The master also issues control commands, such as to start a motor or reset a circuit breaker, and logs the incoming data.

The IEEE 1815-2012 specification describes how the DNP3 protocol implementation must be adapted to run either over TCP (recommended) or UDP. This specification defines connection management between the DNP3 protocol and the IP layers, as shown in Figure

2.18. Connection management links the DNP3 layers with the IP layers in addition to the configuration parameters and methods necessary for implementing the network connection. The IP layers appear transparent to the DNP3 layers as each piece of theprotocol stack in one station logically communicates with the respective part in the other. This means that the DNP3 endpoints or devices are not aware of the underlying IP transport that is occurring.

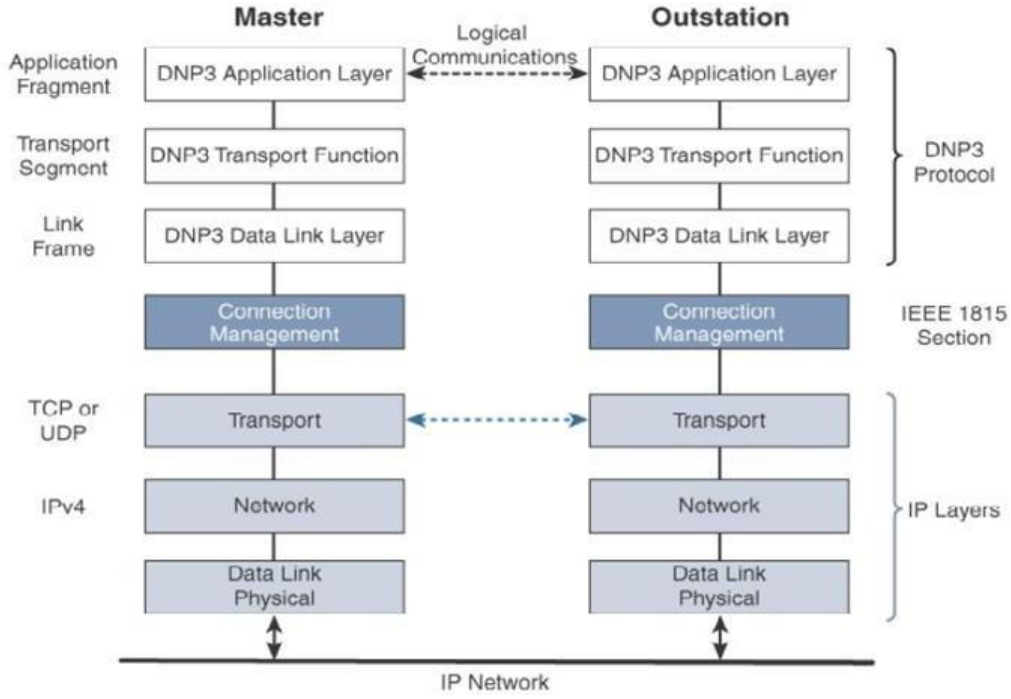


Figure 2.18: Protocol Stack for Transporting Serial DNP3 SCADA over IP

In Figure 2.18, the master side initiates connections by performing a TCP active open. The outstation listens for a connection request by performing a TCP passive open. *Dual endpoint* is defined as a process that can both listen for connection requests and perform an active open on the channel if required. Master stations may parse multiple DNP3 data link layer frames from a single UDP datagram, while DNP3 data link layer frames cannot span multiple UDP data grams. Single or multiple connections to the master may get established while a TCP keepalive timer monitors the status of the connection. Keepalive messages are implemented as DNP3 data link layer status requests. If a response is not received to a keepalive message, the connection is deemed broken, and the appropriate action is taken.

DATA ANALYTICS AND SUPPORTING SERVICES

Traditional data management systems are simply unprepared for the demands of what has come to be known as "big data." As discussed throughout this book, the real value of IoT is not just in connecting things but rather in the data produced by those things, the new services you can enable via those connected things, and the business insights that the data can reveal. However, to be useful, the data needs to be handled in a way that is organized and controlled. Thus, a new approach to dataanalytics is needed for the Internet of Things.

In the world of IoT, the creation of massive amounts of data from sensors is common and one of the biggest challenges—not only from a transport perspective but also from a data management standpoint. A great example of the deluge of data that can be generated by IoT is found in the commercial aviation industry and the sensors that are deployed throughout an aircraft. Modern jet engines are fitted with thousands of sensors that generate a whopping 10GB of data per second.

For example, modern jet engines, similar to the one shown in Figure 1, may be equipped with around 5000 sensors. Therefore, a twin engine commercial aircraft with these engines operating on average 8 hours a day will generate over 500 TB of data daily, and this is just the data from the engines! Aircraft today have thousands of other sensors connected to the airframe and other systems. In fact, a single wing of a modernjumbo jet is equipped with 10,000 sensors.

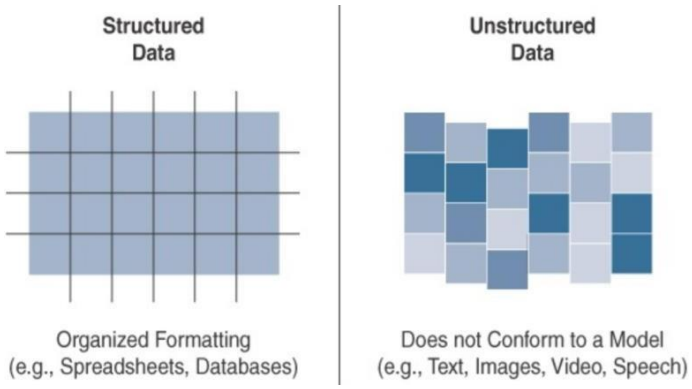
The potential for a petabyte (PB) of data per day per commercial airplane is not farfetched—and this is just for one airplane. Across the world, there are approximately 100,000 commercial flights per day. The amount of IoT data coming just from the commercial airline business is overwhelming. This example is but one of many that highlight the big data problem that is being exacerbated by IoT. Analyzing this amount of data in the most efficient manner possible falls under the umbrella of data analytics. Data analytics must be able to offer actionable insights and knowledge from data, no matter the amount or style, in a timely manner, or the full benefits of IoT cannot be realized.

Before diving deeper into data analytics, it is important to define a few key concepts related to data. For one thing, not all data is the same; it can be categorized

and thus analyzed in different ways. Depending on how data is categorized, various data analytics tools and processing methods can be applied. Two important categorizations from an IoT perspective are whether the data is structured or unstructured and whether it is in motion or at rest.

Structured vs unstructured data

Structured data and unstructured data are important classifications as they typically require different toolsets from a data analytics perspective. Figure 2 provides a high- level comparison of structured data and unstructured data.



Structured data means that the data follows a model or schema that defines how the data is represented or organized, meaning it fits well with a traditional relational database management system (RDBMS). Inmany cases you will find structured data in a simple tabular form—for example, a spreadsheet where data occupies a specific cell and can be explicitly defined and referenced. Structured data can be found in most computing systems and includes everything from banking transaction and invoices to computer log files and router configurations. IoT sensor data often uses structured values, such as temperature, pressure, humidity, and so on, which are all sent in a known format. Structured data is easily formatted, stored, queried, and processed; for these reasons, it has been the core type of data used for making business decisions. Because of the highly organizational format of structured data, a wide array of data analytics tools are readily available for processing this type of data. From custom scriptsto commercial software like Microsoft Excel and Tableau, most people are familiar and comfortable with working with structured data. Unstructured data lacks a logical schema for understanding and decoding the data through traditional programming means. Examples of this data type include text, speech, images, and video. As a general rule, any data that does not fit neatly into a predefined data model is classified as unstructured data. According to some estimates, around 80% of a business's data is unstructured. Because of this fact, data analytics methods that can be applied to unstructured data, such as cognitive computing and machine learning, are deservedly garnering a lot of attention. With machine learning applications, such as natural language processing (NLP), you can decode speech. With image/facial recognition applications, you can extract critical information from still images and video. The

handling of unstructured IoT data employing machine learning techniques is covered in more depth later.

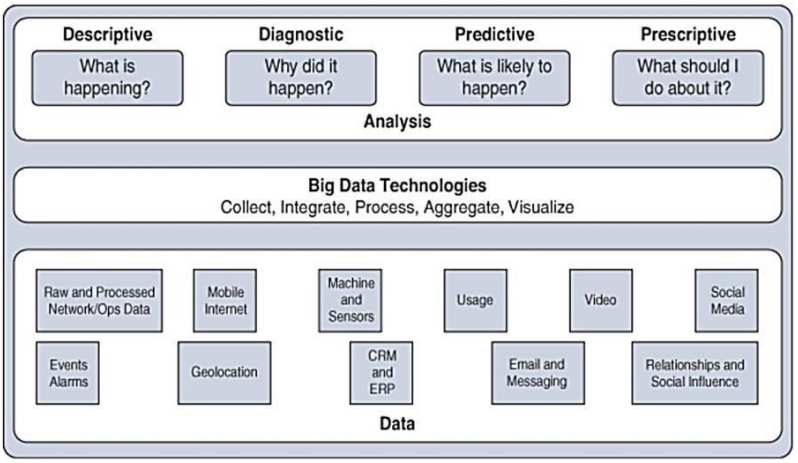
Smart objects in IoT networks generate both structured and unstructured data. Structured data is more easily managed and processed due to its well-defined organization. On the other hand, unstructured data can be harder to deal with and typically requires very different analytics tools for processing the data. Being familiar with both of these data classifications is important because knowing which data classification you are working with makes integrating with the appropriate data analytics solution much easier.

Data in motion versus data at rest

As in most networks, data in IoT networks is either in transit ("data in motion") or being held or stored ("data at rest"). Examples of data in motion include traditional client/server exchanges, such as web browsing and file transfers, and email. Data saved to a hard drive, storage array, or USB drive is data at rest. From an IoT perspective, the data from smart objects is considered data in motion as it passes through the network en route to its final destination. This is often processed at the edge, using fog computing. When data is processed at the edge, it may be filtered and deleted or forwarded on for further processing and possible storage at a fog node or in the data center. Data does not come to rest at the edge.

When data arrives at the data center, it is possible to process it in real-time, just like at the edge, while it is still in motion. Tools with this sort of capability, such as Spark, Storm, and Flink, are relatively nascent compared to the tools for analyzing stored data. Later sections of this chapter provide more information on these real-time streaming analysis tools that are part of the Hadoop ecosystem. Data at rest in IoT networks canbe typically found in IoT brokers or in some sort of storage array at the data center. Myriad tools, especially tools for structured data in relational databases, are available from a data analytics perspective. The best known of these tools is Hadoop. Hadoop notonly helps with data processing but also data storage. It is discussed in more detail later.

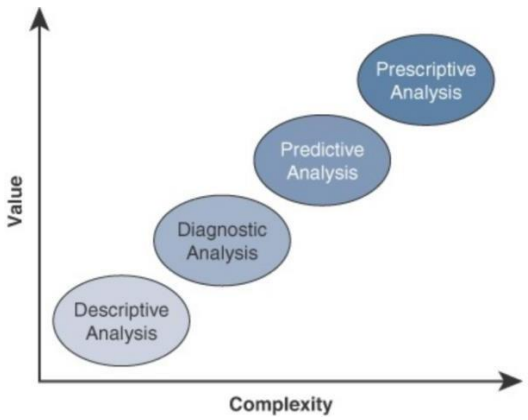
Descriptive: Descriptive data analysis tells you what is happening, either now or in the past. For example, a thermometer in a truck engine reports temperature values every second. From a descriptive analysis perspective, you can pull this data at any moment to gain insight into the current operating 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 and recommends solutions for upcoming problems. A prescriptive analysis of the temperature data froma truck engine might calculate various alternatives to cost-effectively maintain ourtruck. 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.



IoT Data Analytics Challenges

As IoT has grown and evolved, it has become clear that traditional data analytics 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 moredemanding. 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 designed correctly 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 a 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 high volume of smart object data that frequently changes, IoT also brings challenges with the live streaming 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 of a very high volume, and it is valuable only if it is possible to analyze 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 kindof immediate response. To have a chance of affecting the outcome of this problem, you naturally must be able to filter and analyze the data while it is occurring, as close to the edge as possible. The market for analyzing 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. (Edge streaming analytics is discussed in depth later in this chapter.) 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. Network analytics, including both Flexible NetFlow and IPFIX, is covered in more detail.

Data acquiring

Having learnt about devices, devices-network data, messages and packet communication to the Internet, let us understand the functions required for applications, services and business processes at application-support and application layers. These functions are data acquiring, data storage, data transactions, analytics, results visualisations, IoT applications integration, services, processes, intelligence, knowledge discovery and knowledge management.

Let us first discuss the following terms and their meanings used in IoT application layers.

Application refers to application software or a collection of software components. An application enables a user to perform a group of coordinated activities, functions and tasks. Streetlights control and monitoring is an example of an application. Software for tracking and inventory control are other examples of applications. Tracking applications use tags and locations data of the RFIDs.

An application enables a user to withdraw cash using an Automatic Teller Machine (ATM). An umbrella sending warning messages for weather (Example 1.1), a waste container management, health monitoring, traffic lights control, synchronisation and monitoring are other examples of IoT applications.

Service denotes a mechanism, which enables the provisioning of access to one or more capabilities. An interface for the service provides the access to capabilities. The access to each capability is consistent with constraints and policies, which a service-description specifies. Examples of service capabilities are automotive maintenance service capabilities or service capabilities for the Automatic Chocolate Vending Machines (ACVMs) for timely filling of chocolates into the machines.

Service consists of a set of related software components and their functionalities. The set is reused for one or more purposes. Usage of the set is consistent with the controls, constraints and policies which are specified in the service description for each service. A service also associates a Service Level Agreement (SLA).

A service consists of a collection of self-contained, distinct and reusable components. It provides logically grouped and encapsulated functionalities. Traffic lights synchronising service, automobile maintenance service, devices location, detection and tracking service, home security-breach detection and management service, waste containers substitution service, and health-alerts service are the examples of IoT services.

Service Oriented Architecture (SOA) is a software architecture model, which consists of services, messages, operations and processes. SOA components are distributed over a network or the Internet in a high-level business entity. New business applications and applications integration architecture in an enterprise can be developed using an SOA.

Message means a communicating entity or object.

Operation means action or set of actions. For example, actions during a bank transaction.

Transaction (trans + action) refers to two inter-related sets of operations or actions or instructions. For example, a transaction may be access to sales data to select and get the annual sales in a specific year in return. One operation is access to sales data and other is annual sales in return. Another example of a transaction is a query transaction with a Database Management System (DBMS).

Query is a command for getting select values from a database which in return transfers the answer to the query after its processing. A query example is command to ACVMs database for providing sales data of ACVMs on Sundays near city gardens in a specific festival period in a year. Another example is query to service center database forproviding the list of automobile components needing replacement that have completedexpected service-life in a specific vehicle.

Query Processing is a group of structured activities undertaken to get the results from a data store as per the query

Key Value Pair (KVP) refers to a set of two linked entities, one is the key, which is a unique identifier for a linked entity and the other is the value, which is either the entity that is identified or a pointer to the location of that entity. A KVP example is birthday- date pair. KVP is birthday: July 17, 2000. Birthday is the key for a table and date July 17, 2000 is the value. KVP applications create the look-up tables, hash tables and the network or device configuration files.

Hash Table (also called hash map) refers to a data structure which maps the KVPs and is used to implement an associative array (for example array of KVPs). A hash table may use an index (key) which is computed using a hash function and key maps to the value. Index is used to get or point to the desired value.

Bigtable maps two arbitrary string values into an associated arbitrary byte array. One is used as row key and the other as column key. Time stamp associates in three- dimensional mapping. Mapping is unlike a relational database but can be considered as a sparse, distributed multi-dimensional sorted map. The table can scale up to 100s to 1000s of distributed computing nodes with ease of adding more nodes.

Business Transaction (BT) in database theory, refers to a (business) process that requests information from or that changes the data in a database.One operation in a BT

is a command ‘connect’ that connects a DBMS and database, which in turn also connects with the DBMS. Similarly, BTs are processes using commands ‘insert’, ‘delete’, ‘append’, and ‘modify’.

Process means a composition of a group of structured activities or tasks that lead to a particular goal (or that interact to achieve a result). For example, streetlights control process of the purchase process for an airline ticket. A process specifies activities with relevance rules based on data in the process.

Process Matrix is a multi-element entity, each element of which relates a set of data or inputs to an activity (or subset of activities).

Business Process (BP) is an activity or series of activities or a collection of inter-related structured activities, tasks or processes. A BP serves a particular goal or specific result or service or product. The BP is a representation or process matrix or flowchart of a sequence of activities with interleaving decision points; interleaving means putting in between. Decision point means an instance in a series of activities when decisions are taken for further activities.

A web definition states, “a BP is a specific event in a chain of structured business activities that typically change the state of data and/or a product and generate some type of output”. Examples of BPs include finding the annual sales growth and managing the supplies. Another definition2 of BP is that “business process is an activity or set of activities that will accomplish a specific organizational goal”. One more definition3 of BPstates, “BP is a series of logically related activities or tasks (such as planning, production, or sales) performed together to produce a defined set of results.”

Business Intelligence (BI) is a process which enables a business service to extract new facts and knowledge, and then undertake better decisions. These new facts and knowledge follow from earlier results of data processing, aggregation and analysis of these results.

Data Acquiring

1. Data Generation

Data generates at devices that later on, transfers to the Internet through a gateway. Data generates as follows:

Passive devices data: Data generate at the device or system, following the result of interactions. A passive device does not have its own power source. An external source helps such a device to generate and send data. Examples are an RFID or an ATM debit card. The device may or may not have an associated microcontroller, memory and transceiver. A contactless card is an example of the former and a label or barcode is the example of the latter.

Active devices data: Data generates at the device or system or following the result of interactions. An active device has its own power source. Examples are active RFID, streetlight sensor or wireless sensor node. An active device also has an associated microcontroller, memory and transceiver.

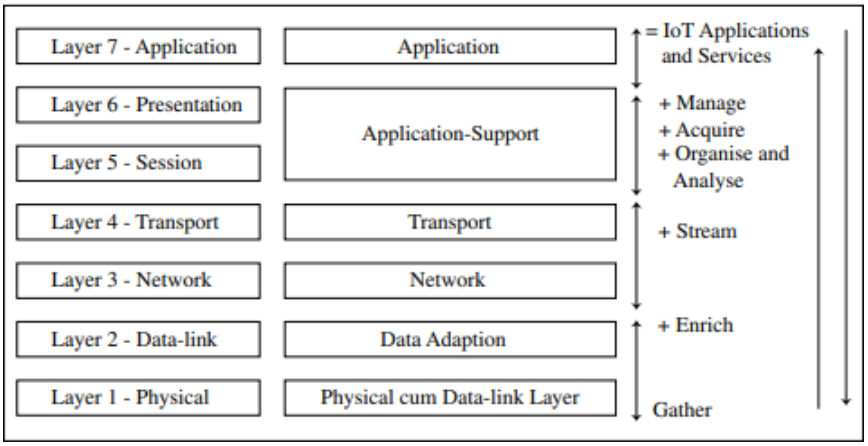
Event data: A device can generate data on an event only once. For example, on detection of the traffic or on dark ambient conditions, which signals the event. The event on darkness communicates a need for lighting up a group of streetlights (Example 1.2). A system consisting of security cameras can generate data on an event of security breach or on detection of an intrusion. A waste container with associate circuit can generate data in the event of getting it filled up 90% or above. The components and devices in an automobile generate data of their performance and functioning. Forexample, on wearing out of a brake lining, a play in steering wheel and reduced air- conditioning is felt. The data communicates to the Internet. The communication takes place as and when the automobile reaches near a Wi-Fi access point.

Device real-time data: An ATM generates data and communicates it to the server instantaneously through the Internet. This initiates and enables Online Transactions Processing (OLTP) in real time.

Event-driven device data: A device data can generate on an event only once. Examples are: (i) a device receive command from Controller or Monitor, and then performsaction(s) using an actuator. When the action completes, then the device sends an acknowledgement; (ii) when an application seeks the status of a device, then the devicecommunicates the status.

2. Data acquisition

Data acquisition means acquiring data from IoT or M2M devices. The data communicates after the interactions with a data acquisition system (application). The application interacts and communicates with a number of devices for acquiring the needed data. The devices send data on demand or at programmed intervals. Data of devices communicate using the network, transport and security layers.



An application can configure the devices for the data when devices have configuration capability. For example, the system can configure devices to send data at definedperiodic intervals. Each device configuration controls the frequency of data generation. For example, system can configure an umbrella device to acquire weather data from the Internet weather service, once each working day in a week. An ACVM can be configured to communicate the sales data of machine and other information, every hour. The ACVM system can be configured to communicate instantaneously in event of fault or in case requirement of a specific chocolate flavour needs the Fill service.

Application can configure sending of data after filtering or enriching at the gateway at the data-adaptation layer. The gateway in-between application and the devices can provision for one or more of the following functions—transcoding, data management and device management. Data management may be provisioning of the privacy and security, and data integration, compaction and fusion

Device-management software provisions for device ID or address, activation, configuring (managing device parameters and settings), registering, deregistering, attaching, and detaching

3. Data validation

Data acquired from the devices does not mean that data are correct, meaningful or consistent. Data consistency means within expected range data or as per pattern or data not corrupted during transmission. Therefore, data needs validation checks. Data validation software do the validation checks on the acquired data. Validation software applies logic, rules and semantic annotations. The applications or services depend on valid data. Then only the analytics, predictions, prescriptions, diagnosis and decisions can be acceptable.

Large magnitude of data is acquired from a large number of devices, especially, from machines in industrial plants or embedded components data from large number of automobiles or health devices in ICUs or wireless sensor networks, and so on. Validation software, therefore, consumes significant resources. An appropriate strategy needs to be adopted. For example, the adopted strategy may be filtering out the invaliddata at the gateway or at device itself or controlling the frequency of acquiring or cyclically scheduling the set of devices in industrial systems. Data enriches, aggregates, fuses or compacts at the adaptation layer.

4. Data Categorization for Storage

Services, business processes and business intelligence use data. Valid, useful and relevant data can be categorized into three categories for storage—data alone, data as well as results of processing, only the results of data analytics are stored. Following are three cases for storage:

a. Data which needs to be repeatedly processed, referenced or audited in future, and therefore, data alone needs to be stored.

b. Data which needs processing only once, and the results are used at a later time using the analytics, and both the data and results of processing and analytics are stored. Advantages of this case are quick visualization and reports generation without reprocessing. Also the data is available for reference or auditing in future.

c. Online, real-time or streaming data need to be processed and the results of this processing and analysis need storage.

Data from large number of devices and sources categorizes into a fourth category called Big data. Data is stored in databases at a server or in a data warehouse or on a Cloud as Big data.

5. Assembly Software for the Events

A device can generate events. For example, a sensor can generate an event when temperature reaches a preset value or falls below a threshold. A pressure sensor in a boiler generates an event when pressure exceeds a critical value which warrants attention. Each event can be assigned an ID. A logic value sets or resets for an event state. Logic 1 refers to an event generated but not yet acted upon. Logic 0 refers to an event generated and acted upon or not yet generated. A software component in applications can assemble the events (logic value, event ID and device ID) and can also add Date time stamp. Events from IoTs and logic-flows assemble using software.

6. Data store

A data store is a data repository of a set of objects which integrate into the store. Features of data store are:

Objects in a data-store are modeled using Classes which are defined by the database schemas

A data store is a general concept. It includes data repositories such as database, relational database, flat file, spreadsheet, mail server, web server, directory services and VMware

A data store may be distributed over multiple nodes. Apache Cassandra is an example of distributed data store.

A data store may consist of multiple schemas or may consist of data in only onescheme. Example of only one scheme data store is a relational database.

Repository in English means a group, which can be related upon to look for required things, for special information or knowledge. For example, a repository of paintings of artists. A database is a repository of data which can be relied upon for reporting,

analytics, process, knowledge discovery and intelligence. A flat file is another repository.

7. Data Centre Management

A data centre is a facility which has multiple banks of computers, servers, large memory systems, high speed network and Internet connectivity. The centre provides data security and protection using advanced tools, full data backups along with data recovery, redundant data communication connections and full system power as well as electricity supply backups.

Large industrial units, banks, railways, airlines and units for whom data are the critical components use the services of data centres. Data centres also possess a dust free, heating, ventilation and air conditioning (HVAC), cooling, humidification and dehumidification equipment, pressurisation system with a physically highly secure environment. The manager of data centre is responsible for all technical and IT issues, operations of computers and servers, data entries, data security, data quality control, network quality control and the management of the services and applications used for data processing.

8. Server Management

Server management means managing services, setup and maintenance of systems of all types associated with the server. A server needs to serve around the clock. Server management includes managing the following:

● Short reaction times when the system or network is down

● High security standards by routinely performing system maintenance and updation ● Periodic system updates for state-of-the art setups

● Optimised performance

● Monitoring of all critical services, with SMS and email notifications

● Security of systems and protection

● Maintaining confidentiality and privacy of data

● High degree of security and integrity and effective protection of data, files anddatabases at the organisation

● Protection of customer data or enterprise internal documents by attackers which includes spam mails, unauthorised use of the access to the server, viruses, malwares and worms

● Strict documentation and audit of all activities.

9. Spatial Storage

Consider goods with RFID tags. When goods move from one place to another, the IDs of goods as well as locations are needed in tracking or inventory control applications. Spatial storage is storage as spatial database which is optimised to store and later on receives queries from the applications. Suppose a digital map is required for parking slots in a city. Spatial data refers to data which represents objects defined in a geometric space. Points, lines and polygons are common geometric objects which can be represented in spatial databases. Spatial database can also represent database for 3D objects, topological coverage, linear networks, triangular irregular networks and other complex structures. Additional functionality in spatial databases enables efficient processing. Internet communication by RFIDs, ATMs, vehicles, ambulances, traffic lights,streetlights, waste containers are examples of where spatial database are used.

Spatial database functions optimally for spatial queries. A spatial database can perform typical SQL queries, such as select statements and performs a wide variety of spatial operations. Spatial database has the following features:

● Can perform geometry constructors. For example, creating new geometries

● Can define a shape using the vertices (points or nodes)

● Can perform observer functions using queries which replies specific spatial information such as location of the centre of a geometric object

● Can perform spatial measurements which mean computing distance between geometries, lengths of lines, areas of polygons and other parameters

● Can change the existing features to new ones using spatial functions and can predicate spatial relationships between geometries using true or false type queries.

Computing Using a Cloud Platform for IoT/M2M Applications/Services

A few conventional methods for data collection and storage are as follows:

● Saving devices' data at a local server for the device nodes

● Communicating and saving the devices' data in the files locally on removable media, such as micro SD cards and computer hard disks

● Communicating and saving the data and results of computations in a dedicated data store or coordinating node locally

● Communicating and saving data at a local node, which is a part of a distributed DBMS

● Communicating and saving at a remote node in the distributed DBMS

● Communicating on the Internet and saving at a data store in a web or enterprise

server

● Communicating on the Internet and saving at data centre for an enterprise

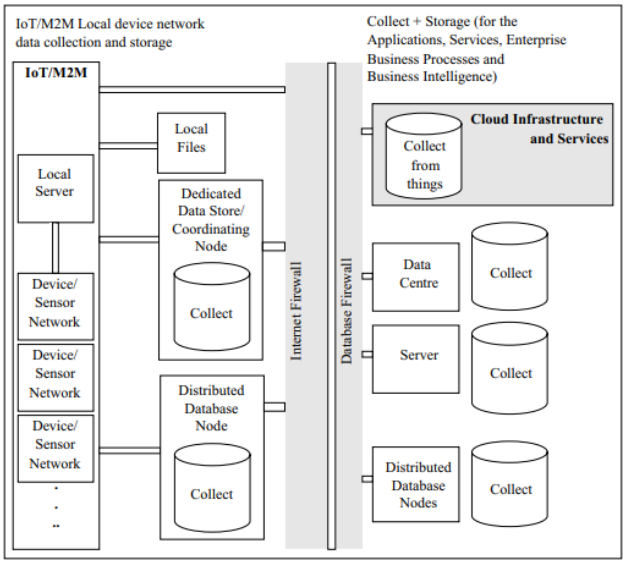
Cloud is a new generation method for data collection, storage and computing. cloud computing paradigm for data collection, storage, computing and services. describes cloud-computing service models in a software architectural concept, 'everything as a service'.Describes IoT specific cloud based services, Xively, Nimbits. describes platforms such as AWS IoT, Cisco IoT, IOx and Fog, IBM IoT Foundation, TCS Connected Universe (TCS CUP).

Different methods of data collection, storage and computing are shown in Figure 6.1. The figure shows (i) Devices or sensor networks data collection at the device web server, (ii) Local files, (iii) Dedicated data store at coordinating node, (iii) Local node in a distributed DBMS, (iv) Internet-connected data centre, (v) Internet-connected server,
(vi) Internet-connected distributed DBMS nodes, and (vii) Cloud infrastructure and services.

Cloud computing paradigm is a great evolution in Information and Communications Technology (ICT). The new paradigm uses XAAS at the Internet connected clouds for collection, storage and computing.

Following are the key terms and their meanings, which need to be understood before learning about the cloud computing platform.

Resource refers to one that can be read (used), written (created of changed) or executed (processed). A path specification is also a resource. The resource is atomic (not-further divisible) information, which is usable during computations. A resource may have multiple instances or just a single instance. The data point, pointer, data, object, data store or method can also be a resource.



Devices or sensors network data collection at a device local-server, local files, dedicated data store, at a coordinating node, a local node of a distributed DBMS, Internet-connected server of data centre, server or distributed database nodes or a cloud infrastructure

System resource refers to an operating system (OS), memory, network, server, software or application. Environment refers to an environment for programming, program execution or both. For example, cloud9 online provides an open programming environment for BeagleBoneboard for the development of IoT devices; Windows environment for programming and execution of applications; Google App Engine environment for creation and execution of web applications in Python or Java. Platform denotes the basic hardware, operating system and network, and is used for software applications or services over which programs can be run or developed.

A platform may provide a browser and APIs which can be used as a base on which other applications can be run or developed.

Edge computing is a type of computing that pushes the frontier of computing applications, data and services away from centralised nodes to IoT data generating nodes, that means at logical extremes of the network.2 IoT device nodes are pushed by events, triggers, alerts, messages and data is collected for enrichment, storage and computations from the remote centralised database nodes. Pushing the computations from centralised nodes enables the usage of resources at device nodes, which could be a requirement in case of low power lossy networks. The processing can also be classified as edge computing at local cloud, grid or mesh computing. The nodes may be mobile or of a wireless sensor network or cooperative distributed in peer-to-peer and ad-hoc networks.

Distributed computing refers to computing and usage of resources which are distributed at multiple computing environments over the Internet. The resources are logically-related, which means communicating among themselves using message passing and transparency concepts, and are cooperating with each other, movable without affecting the computations and can be considered as one computing system (location independent).

Service is a software which provides the capabilities and logically grouped and encapsulated functionalities. A service is called by an application for utilising the capabilities. A service has a description and discovery methods, such as advertisement for direct use or through a service broker. The service binds to Service Level Agreement (SLA) between service (provider end point) and application (end point). One service can also use another service.

Web Service, according to the W3C definition, is an application identified by a URI, described and discovered using the XML based Web-Service Description Language (WSDL). A web serviceinteracts with other services and applications using XML messages and exchanges the objects using Internet protocols.

Service-oriented architecture consists of components which are implemented as independent

services which can be dynamically bonded and orchestrated, and which possess loosely coupled configurations, while the communication between them uses messages. Orchestrating means a process which predefines an order of calling the services (in sequences and in parallel) and the data and message exchanges.

Web computing refers to computing using resources at computing environment of web server(s) or web services over the Internet.

Grid computing refers to computing using the pooled interconnected grid of computing resources and environments in place of web servers.

Utility computing refers to computing using focus on service levels with optimum amount of resources allotted when required and takes the help of pooled resources and environmentsfor hosting applications. The applications utilise the services.

Cloud computing refers to computing using a collection of services available over the Internet that deliver computational functionality on the infrastructure of a service provider for connected systems and enables distributed grid and utility computing.

Key Performance Indicator (KPI) refers to a set of values, usually consisting of one or more raw monitored values including minimum, average and maximum values specifying the scale. A service is expected to be fast, reliable and secure. The KPIs monitor the fulfillment of these objectives. For example, a set of values can relate to Quality of Service (QoS) characteristics, such as bandwidth availability, data backup capability, peak and average workload handling capacity, ability to handle defined volume of demand at different times of the day, and the ability to deliver defined total volume of service. A cloud service should be able to fulfill the defined minimum, average and maximum KPI values agreed in the SLA

Localisationmeans cloud computing content usage is monitored by determining localisation of the QoS level and KPIs.

Seamless cloud computing means during computing the content usages and computations continue without any break when the service usage moves to a location with similar QoS level and KPIs. For example, continue using same cloud platform when developer of software shifts.

Elasticity denotes that an application can deploy local as well as remote applications orservices and release them after the application usage. The user incurs the costs as per the usages and KPIs.

Measurability (of a resource or service) is something which can be measured for controlling or monitoring and enables report of the delivery of resource or service.

Homogeneity of different computing nodes in a cluster or clusters refers to integration with the kernel providing the automatic migration of the processes from one to other homogeneous nodes. System software on each computing node should ensure same storage representation and same results of processing

Resilient computing refers to the ability of offering and maintaining the accepted QoS and KPIs in presence of the identified challenges, defined and appropriate resilience metrics, and protecting the service.4 The challenges may be small to big, such as misconfiguration of a computing node in a network to natural disasters. An English Cambridge dictionary gives the meaning of resilience as the power or ability to return to the original form, position, etc., afterbeing bent, compressed or stretched.

Scalability in cloud services refers to the ability using which an application can deploy smaller local resources as well as remotely distributed servers and resources, and then increase or decrease the usage, while incurring the cost as per the usage on increasing scales.

Maintainability in cloud services refers to the storage, applications, computing infrastructure, services, data centres and servers maintenances which are responsibilities of the remotely connected cloud services with no costs to the user.

XAAS is a software architectural concept that enables deployment and development of applications, and offers services using web and SOA. A computing paradigm is to integrate complex applications and services (Section 5.3.5) and use XAAS concept for deploying a cloud platform

Multitenant cloud model refers to accessibility to a cloud platform and computingenvironment by multiple users who pay as per the agreed QoS and KPIs, which are defined at separate SLAs with each user. Resource pooling is done by the users but each uses pays separately. The following subsections describe the cloud computing paradigm and deploymentmodels.

Cloud Computing Paradigm

Cloud computing means a collection of services available over the Internet. Cloud delivers the computational functionality. Cloud computing deploys infrastructure of a cloud-service provider. The infrastructure deploys on a utility or grid computing or webservices environment that includes network, system, grid of computers or servers or data centres. Just as we—users of electricity—do not need to know about the source and underlying infrastructure for electricity supply service, similarly, a user of computing service or application need not know how the infrastructure deploys or the details of the computing environment. Just as the user does not need to know Intel processor inside a computer, similarly, the user uses the data, computing and intelligence in the cloud, as part of the services. Similarly, the services are used as a utility at the cloud.

Cloud Platform Services

Cloud platform offers the following:

- Infrastructure for large data storage of devices, RFIDs, industrial plant machines,

automobiles and device networks

- Computing capabilities, such as analytics, IDE (Integrated Development Environment)
- Collaborative computing and data store sharing

Cloud Platform Usages

Cloud platform usages are for connecting devices, data, APIs, applications and services, persons, enterprises, businesses and XAAS.

The following describes a simple conceptual framework of the Internet Cloud:

Internet Cloud + Clients = User applications and services with 'no boundaries and no walls'

An application or service executes on a platform which includes the operating system (OS), hardware and network. Multiple applications may initially be designed to run on diversified platforms (OSs, hardware and networks). Applications and services need to integrate them on a common platform and running environment. Cloud storage and computing environment offers a virtualized environment, which refers to a running environment made to appear as one to all applications and services, but in fact physically two or more running environments and platforms may be present.

Virtualization

A characteristic of virtualized environment is that it enables applications and services toexecute in an independent execution environment (heterogeneous computing environment). Each one of them stores and executes in isolation on the same platform, though in fact, it mayactually execute or access to a set of data centers or servers or distributed services and computing systems. The applications or services which are hosted remotely and are accessibleusing the Internet can easily be deployed at a user application or service in a virtualized environment, provided the Internet or other communications are present.

Applications need not be aware of the platform, just Internet connectivity to the platform, called cloud platform, is required. The storage is called cloud storage. The computing is called cloud computing. The services are called cloud services in line with the web services whichhost on web servers.

Virtualization of storage means user application or service accesses physical storage using abstract database interface or file system or logical drive or disk drive, though in fact storage may be accessible using multiple interfaces or servers. For example, Apple iCloud offers storage to a user or user group that enables the sharing of albums, music, videos, data store, editing files and collaboration among the user group members.

Network Function Virtualisation (NFV) means a user application or service accesses the resources appearing as just one network, though the network access to the resources may be through multiple resources and networks.

Virtualisation of server means user application accesses not only one server but in fact accesses multiple servers.

Virtualised desktop means the user application can change and deploy multiple desktops, though the access by the user is through their own computer platform (OS) that in fact may bethrough multiple OSs and platforms or remote computers

Cloud Computing Features and Advantages

Essential features of cloud storage and computing are:

● On demand self-service to users for the provision of storage, computing servers, software delivery and server time

● Resource pooling in multi-tenant model

● Broad network accessibility in virtualised environment to heterogeneous users, clients, systems and devices

● Elasticity

● Massive scale availability

● Scalability

● Maintainability

● Homogeneity

● Virtualisation

● Interconnectivity platform with virtualised environment for enterprises and provisioning of in-between Service Level Agreements (SLAs)

● Resilient computing

● Advanced security

● Low cost

Cloud Computing Concerns

Concerns in usage of cloud computing are:

● Requirement of a constant high-speed Internet connection ● Limitations of the services available

● Possible data loss

● Non delivery as per defined SLA specified performance

- Different APIs and protocols used at different clouds
- Security in multi-tenant environment needs high trust and low risks
- Loss of users' control

Cloud Deployment Models

Following are the four cloud deployment models:

Public cloud: This model is provisioned by educational institutions, industries, government institutions or businesses or enterprises and is open for public use.

Private cloud: This model is exclusive for use by institutions, industries, businesses or enterprises and is meant for private use in the organisation by the employees and associated users only.

. Community cloud: This model is exclusive for use by a community formed by institutions, industries, businesses or enterprises, and for use within the community organisation, employees and associated users. The community specifies security and compliance considerations.

Hybrid cloud: A set of two or more distinct clouds (public, private or community) with distinct data stores and applications that bind between them to deploy the proprietary or standard technology.

Cloud platform architecture is a virtualised network architecture consisting of a cluster of connected servers over the data centres and Service Level Agreements (SLAs) between them.A cloud platform controls and manages resources, and dynamically provisions the networks, servers and storage. Cloud platform applications and network services are utility, grid and distributed services. Examples of cloud platforms are Amazon EC2, Microsoft Azure, Google App Engine, Xively, Nimbits, AWS IoT, CISCO IoT, IOx and Fog, IBM IoT Foundation, TCSConnected Universe Platform.

Everything as a service and cloud service models

Cloud connects the devices, data, applications, services, persons and business. Cloud services can be considered as distribution service—a service for linking the resources (computing functions, data store, processing functions, networks, servers and applications) and for provision of coordinating between the resources.

Cloud computing can be considered by a simple equation:

Cloud Computing = SaaS + Paas + IaaS + DaaS

SaaS means Software as a Service. The software is made available to an application or service on demand. SaaS is a service model where the applications or services deploy and host at the cloud, and are made available through the Internet on demand by the service user. The software control, maintenance, updation to new version and infrastructure, platform and resource requirements are the responsibilities of the cloud service provider.

PaaS means Platform as a Service. The platform is made available to a developer of anapplication on demand. PaaS is a service model where the applications and services develop and execute using the platform (for computing, data store and distribution services) which is made available through the Internet on demand for the developer of the applications. The platform, network, resources, maintenance, updation and security as per the developers' requirements are the responsibilities of the cloud service provider.

IaaS means Infrastructure as a Service. The infrastructure (data stores, servers, data centres and network) is made available to a user or developer of application on demand. Developer installs the OS image, data store and application and controls them at the infrastructure. IaaS is a service model where the applications develop or use the infrastructure which is made available through the Internet on demand on rent (pay as per use in multi-tenancy model) bya developer or user. IaaS computing systems, network and security are the responsibilities of the cloud service provider.

DaaS means Data as a Service. Data at a data centre is made available to a user or developerof application on demand. DaaS is a service model where the data store or data warehouse is made available through the Internet on demand on rent (pay as per use in multi tenancy model) to an enterprise. The data centre management, 24×7 power, control, network, maintenance, scale up, data replicating and mirror nodes and systems as well as physical security are the responsibilities of the data centre service provider.

UNIT-4 : CASE STUDIES & CHALLENGES

Definition:

"A case study is a research strategy and an empirical inquiry that investigates a phenomenon within its real-life context. Case studies are based on an in-depth investigation of a single individual, group or event to explore the causes of underlying principles".

One of the most promising IoT use cases is creating smarter, more efficient cities. Public energy grids can be optimized to balance workloads, predict energy surges, and distribute energy more equitably to customers.Traffic lights could be synced using IoT to adapt to traffic conditions in real-time.

IoT is the next step in the evolution of the internet and is being used in about everything you can think of.

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

IoT Applications:Smart Home, Smart Buildings and Infrastructure

IoT home automation is the ability to control domestic appliances by electronically controlled, internet-connected systems. It may include setting complex heating and lighting systems in advance and setting alarms and home security controls, all connected by a central hub and remote-controlled by a mobile app.

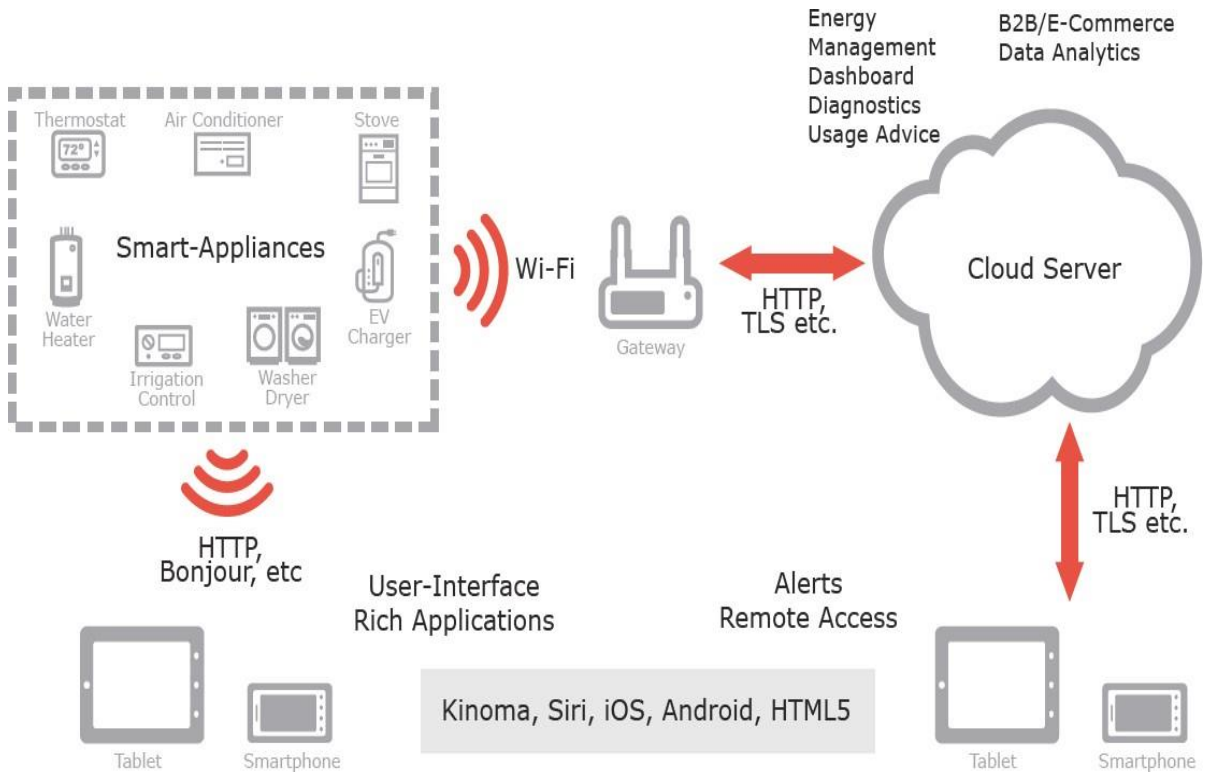


Figure .Smarthomeplatform.

The rise of Wi-Fi's role in home automation has primarily come about due tothe networked nature of deployed electronics where electronic devices (TVsand AV receivers, mobile devices, etc.) have started becoming part of thehome IP network and due the increasingrate of adoption of mobile computingdevices (smartphones, tablets, etc.), see above Figure.

The networking aspectsare bringing online streaming services or network playback, while becoming amean to control of the device functionality over the network. At the same timemobile devices ensure that consumers have access to a portable 'controller' forthe electronics connected to the network. Both types of devices can be used asgateways for IoT applications. In this context many companies are consideringbuilding platforms that

integrate the building automation with entertainment,healthcare monitoring, energy monitoring and wireless sensor monitoring inthe home and building environments.

IoT applications using sensors to collect information about the operating conditions combined with cloud hosted analytics software that analyzes disparatedata points will help facility managers become far more proactive about managingbuildings at peak efficiency.

Issues of building ownership (i.e., building owner, manager, or occupants)challenge integration with questions such as who pays initial system costand who collects the benefits over time. A lack of collaboration between thesubsectors of the building industry slows new technology adoption and canvent new buildings from achieving energy, economic and environmentalperformance targets.

Integration of cyber physical systems both within the building and withexternal entities, such as the electrical grid, will require stakeholder cooperationto achieve true interoperability. As in all sectors, maintaining securitywill be a critical challenge to overcome.

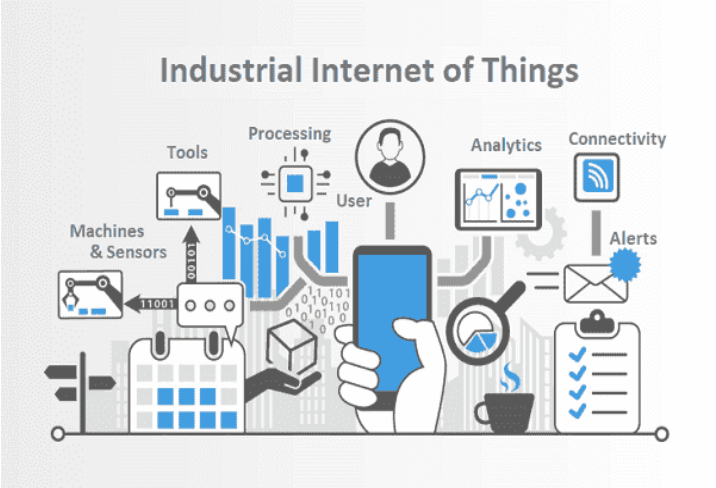
Within this field of research the exploitation of the potential of wirelesssensor networks (WSNs) to facilitate intelligent energy management in buildings,which increases occupant comfort while reducing energy demand, ishighly relevant.

In addition to the obvious economic and environmental gainsfrom the introduction of such intelligent energy management in buildings otherpositive effects will be achieved. Not least of which is the simplification ofbuilding control; as placing monitoring, information feedback equipment andcontrol capabilities in a single location will make a buildings' energy managementsystem easier to handle for the building owners, building managers,maintenance crews and other users of the building. Using the Internet togetherwith energy management systems also offers an opportunity to access abuildings'energy information and control systems from a laptop or a Smartphoneplaced anywhere in the world. This has a huge potential for providing the managers,owners and inhabitants of buildings with energy consumption feedbackand the ability to act on that information.

In the context of the future Internet of Things, Intelligent Building ManagementSystems can be considered part of a much larger information system.This system is used by facilities managers in buildings to manage energy useand energy procurement and to maintain buildings systems. It is based on theinfrastructure of the existing Intranets and the Internet, and therefore utilizes the same standards as other IT devices. Within this context reductionsin thecost and reliability of WSNs are transforming building automation, by makingthemaintenance of energy efficient healthy, productive work spaces inbuildings increasingly cost effective.

IoT Application in industries:

IoT in industry is a rapidly developing area. Numerous IoT research and application projects havebeen done by universities or in joint industry- university consortia in recent years.



Internet of things (IoT) has become part of your daily life. The “things connectedto the internet” ideais continuously evolving in content, areas of applications, visions and technology. New real life and industrial projects havebeen done and joint future oriented industry and government initiatives suchas Industry 4.0 in Germany, have been started [1].Since Industrial production is one of the world’s biggest economic factorsone of the major objectives of these initiatives is to bring the paradigms of theIoT to the factories enabling them to cope with the challenges raised by popularmegatrends.

The foremost megatrends relevant for factories are globalization,progressing technological evolution, the dynamization of product life cycles,the aging work force and the shortage of resources. Central effects are theacceleration of innovation cycles and the increasing customer demand forindividualized mass produces with highest quality expectations.Within the context of industrial production IoT projects and applicationsare developing in manufacturing, supply chain,

supervision and servicing.A major question in all projects is about the value, the benefit such applicationcan bring to the user, to the owner or to society.

The value question is extremely pertinent in the industry: in the manufacturing industry entire factory related processes, but also in industrial applications where it comes to ensure operation of industrial installations and provide supervision, and improved life service. It is the value which such applications bring which will determine their adoption, acceptance and wide use. However, this value is very difficult to quantify and prove, and it depends on multiple aspects which are strongly application area dependent.

IoT applications form the value creation for industry and brings together expert opinions from academia, research and industry. The industrial application of IoT is multi- facetted and each of the subsections in this paper will highlight an aspect related to industrialapplication, discuss or show a case or the evolution and potential of a specific technology from industry application point of view. The paper is having a holistic manner to industrial challenges and requirements. Also it will refer to factory concepts and applications supported by IoT, including processes and flows taking a view on related technologies and their evolution.

IoT applications benefit and value creation in an industrial environment may have its origin in different aspects, depending on the application type. There is no value but "values" each contributing to the total benefit such as:

- Value from visibility identification, location tracking
- Value form IoT-supported safety in hard industrial environments
- Value from right information providing or collecting
- Value form improved industrial operation and flows in industry
- Value from reduced production losses
- Value from reduced energy consumption
- Value from new type of processes made possible by IoT applica-
- tions
- Value form new type of maintenance and lifetime approaches
- Value enabled by smart objects, connected aspects
- Value from sustainability.

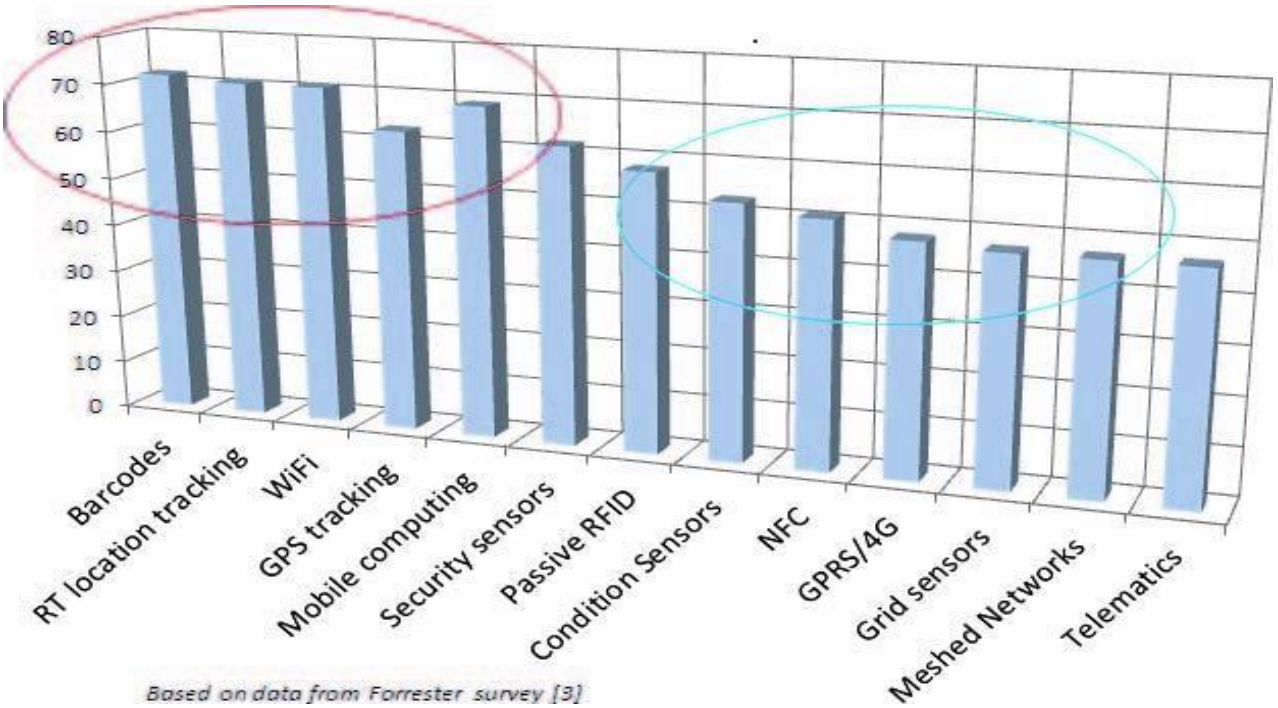


Fig. 5.2 View on very important and important perceived IoT technologies expected to bring value in applications.

The status and estimated potential of IoT applications is presented in Figure 5.3 considering three major areas: supply chain, future industry/future actory and over lifetime applications and activities such as logistics, manufacturing and service/maintenance. A strong potential and additional application is expected in industry operation and industry lifetime applications including lifetime service.

Areas	Supply chain	Industry	Lifetime
Activities	Logistics	Manufacturing	Service
IoT **present** Applications and Value	Many	Some	Few
IoT **additional** Applications **Potential**	Increase	Strong	Strong

Figure.Status and estimated potential of IoT applications.

IoT application requirements and capabilities:

The expectations toward IoT applications in industry are high. The capabilities they have to offer are depending strongly on the industrial area and the concrete application. For example the environment where IoT application may be used may range from clean room condition and normal ambient temperatures to heavy and dirty environment, locations with high temperatures, areas withexplosion risk, areas with metallic surroundings, and corrosive environment on sea or underground.

A list of a set of industry related capabilities and requirements is presented below, without claiming completeness. The list items are related to the IoT hardware, software and to serviceability and management aspects. Comments have been added to all items to make the requirement more specific. The IoT application capabilities for industrial application should meet requirements such as:

- Reliability.
  Reliable IoT devices and systems should allow a continuous operation of industrial processes and perform on-site activities.
- Robustness.
  The IoT application and devices should be robust and adapted to the task and hard working conditions. This should include also the certifications for the specific work environment where they are used.
- Reasonable cost.
  Cost aspects are essential and should be fully justifiable and adapted to the benefit. It is basically about the right balance between cost and benefit rather than low cost. Also the costs are related to a more holistic view and life costs and consider the impact on the whole industrial installation in case of a failed IoT device or application.

- Security and safety.
  Security requirements are related to the cyber security threats and have to be part of the entire security strategy of the company.
  Safety is mainly related to the device construction and the area of use but also to usability such that no safety threats occur due to use of the IoT applications and devices.
- Simple use.
  Simple, intuitive use and (almost) self-explaining are important for the overall IoT application acceptance. The IoT application should ideally be context aware and adapt to the skills of the user and location or environment aspects.
- Optimal and adaptive set of features.
  The IoT application should allow to perform desired task with the sufficient, not-richer-then-necessary, set of features
- Low/No maintenance.
  Maintenance free or reduced maintenance IoT applications and devices over operational life would be ideal. Maintenance over lifetime is an important aspect impacting the life cycle costs of IoT based solutions. It is affected by the sometimes high number of IoT devices in place, the fact that they are typically distributed over large areas, the required skills, tools and time needed for any type of IoT maintenance operation. This is valid for all devices but especially for active IoT devices or active wireless sensing.
- Standardization.
  IoT devices and applications should be using a set of standards to support interoperability of IoT devices, easy exchange and multi-vendor possibilities.
- Integration capabilities.
  Easy integration in the IT and automation and process landscape of the industrial plant are required and may decide if a IoT solution will be used. This is particularly important for brown-field projects but also for green field in the view of future plant extensions.
- Reach sensing and data capabilities.
  IoT applications will relay more and more on complex sensing allowing distributed supervision and data collection and data capabilities. This is a chance in terms of additional data and

real-time information but also a challenge in terms of data and processing.

- Industry grade support and services.
  The IoT applications should be supported over years in operation by a set of rich tools and continuously updated services. Typically industry application requires also a centralized management of devices and systems, managed access rights, this might apply to some of IoT devices too.

Presently there are also numerous challenges to reach all the above.

Challenges faced by IoT industry applications

The challenges for IoT industrial applications can be subject of a more extended treatment, however for the needs of present IoT applications and value creation they have been divided in 4 groups:

- IoT device technical challenges
- Lifetime and energy challenge
- Data and information challenge
- Humans and business

The IoT devices technical challenges are numerous and subject of intense research. Some aspects will be addressed also in the following sections. A set of technical features will be especially needed in industrial applications, depending on application, such as extended capabilities for sensing in terms of sensor types and high sampling rate, communication, wireless data transfer and precise time synchronous collection of data both in single-hop and multi-hop industrial networks. Another aspect is related to the easy deployment, configuration and re-use of non-permanently attached devices, such as the ones used for ad-hoc sensing. One critical and often neglected aspect is the device packaging for the industrial application needs which is essential for reliable operation.

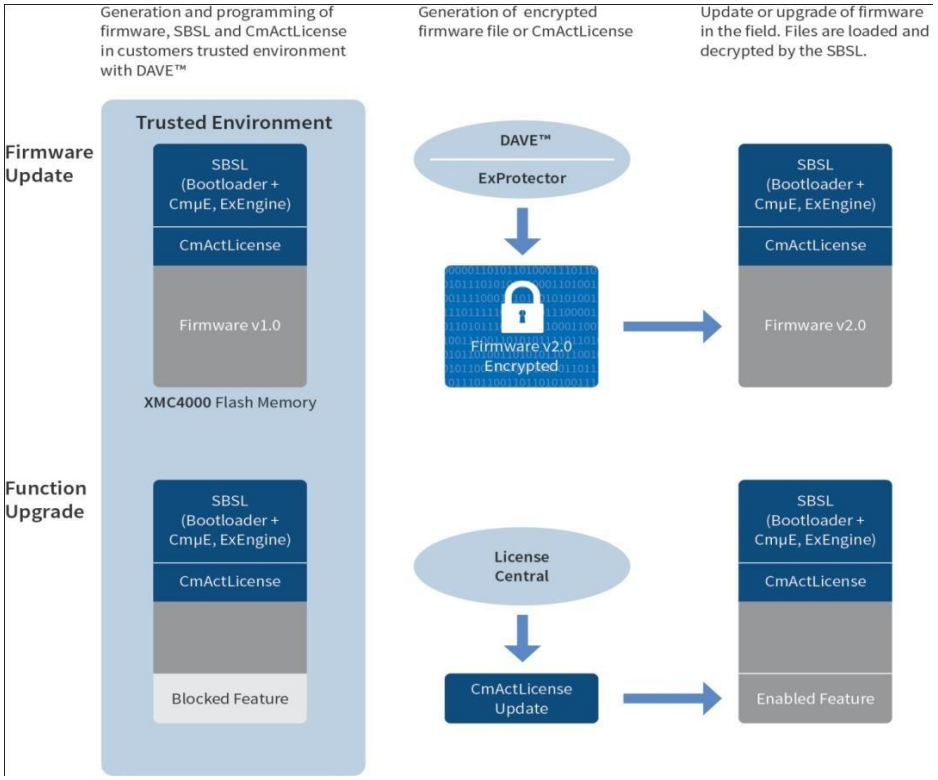
Seurity:

- IoT devices are connected to your desktop or laptop. Lack of security increases the risk of your personal information leaking while the data is collected and transmitted to the IoT device.
- IoT devices are connected with a consumer network. This network is also connected with other systems. So if the IoT device contains any security vulnerabilities, it can be harmful to the consumer's network. This vulnerability can attack other systems and damage them.
- Sometimes unauthorized people might exploit the security vulnerabilities to create risks to physical safety.

Privacy Risks:

- In IoT, devices are interconnected with various hardware and software, so there are obvious chances of sensitive information leaking through unauthorized manipulation.
- All the devices are transmitting the user's personal information such as name, address, dateof birth, health card information, credit card detail and much more without encryption.

Though there are security and privacy concerns with IoT, it adds values to our lives by allowing us to manage our daily routine tasks remotely and automatically, and more importantly, it is a game-changer for industries.



IoT Application of home appliances:

Internet of Things is a technology that can connect to the internet without the influence of people and send information collected to users through this internet network to which they are connected. Devices in this dynamic are very common today. Many homes, companies and even public organizations benefit from this technology. Used in smart home IoT home appliances is also one of them.

A house must have smart devices to be smart. These smart devices are the building blocksof today's technology. So why are these devices and apps smart? First, these devices have their own Internet. With this internet tool, users can receive information from the device. With this internet connection, you can get a lot of information from your smart device. This information which receives from smart devices makes safety for your living area.

Smart devices work with technological devices while making you and your home a more secure space. The biggest hero of these technological devices is microprocessors.microprocessors act as the brain for your smart device. There are sensors that allow your smart devices to be classified according to their characteristics and detect the danger or differences in your home.

There are many sensors classified by type. Motion sensors, light sensors, image detection, and processing sensors are one of them. For example, if the position of your belongings changes without your knowledge, there are motion sensors that can detect this position change. The motion sensor detects the position change and sends you information about this.

Home Appliance in Internet of Things:

Smart home systems are integrated and enable you to play an active role in every part of your homeby surrounding your home. When you're not at home, but your mind stays at home, it's behind you.With smart home systems, you can intervene in your home as if you are at home and perform the necessary controls.In addition to these protection systems, smart home appliances have been making human life easier since the day it was developed.

Smart Washing Machine:

It is very important to save time in daily life. we live in a period where we have to keep up. that's where technology comes in. You can access the developed smart washing machine on your smartphone. you can monitor and control the process at the same time. This smart washing machine can also dry your laundry with the control application.

Smart Refrigerator with Internet of Things:

Internet in this kitchen which makes life easier for you and your family in the kitchen. With this internet connection, you can transmit a lot of information to your shopping list in the weather. You can also view the inside of your refrigerator with its camera technology.

Shortest Way to Dry Hair:

This time it has infrared technology. With this technology, the device is created wirelessly. Wireless shape so you can dry your hair without connecting the machine

Smart Doorbell:

The most important thing in smart home applications is known to be secure and protected home. With this smart doorbell designed for security, you can recognize people who come to your home with high quality. The night also has infrared technology added to the smart bell. This will also send the screen to you when it gets dark.

Smart Camera for Safe Home:

Control of your home is in your hands from every part. This smart camera sends records from every part of your home to your smartphone with the Internet of Things technology. Research on smart camera technology will continue for those who want a safe life.

Industry 4.0 concepts

Industry 4.0 refers to a new phase in the Industrial Revolution that focuses heavily on interconnectivity, automation, machine learning, and real-time data. Industry 4.0, also sometimes referred to as IIoT(Industrial Internet of Things) or smart manufacturing which provides physical production and operations with smart digital technology, machine learning and big data to create a more holistic and better connected ecosystem for companies that focus on manufacturing and supply chain management.

While every company and organization operating today is different, they all face a common challenge—the need for connectedness and access to real-time insights across processes, partners, products, and people. That's where Industry 4.0 comes into play. Industry 4.0 is not just about investing in new technology and tools to improve manufacturing efficiency but it's about revolutionizing the way the entire business operates and grows.

Industry 4.0 refers to the use of automation and data exchange in manufacturing. Accordingto the Boston Consulting Group there are nine principal technologies that make up Industry 4.0: Autonomous Robots, Simulation, Horizontal and Vertical System Integration, the Industrial Internet of Things, Cybersecurity, The Cloud, Additive Manufacturing, Data and Analytics, and Augmented Reality. These technologies are used to create a "smart factory" where machines, systems, and humans communicate with each other in order to coordinate and monitor progress along the assembly line. Networked devices provide sensor data and are digitally controlled. The net effect is the ability to rapidly design, modify, create, and customize things in the real world, while lowering costs and reacting to changes in consumer preferences, demand,the supply chain and technology.

The goal is to enable autonomous decision-making processes, monitor assets and processes in real-time, and enable equally real-time connected value creation networks through early involvement of stakeholders, and vertical and horizontal integration.

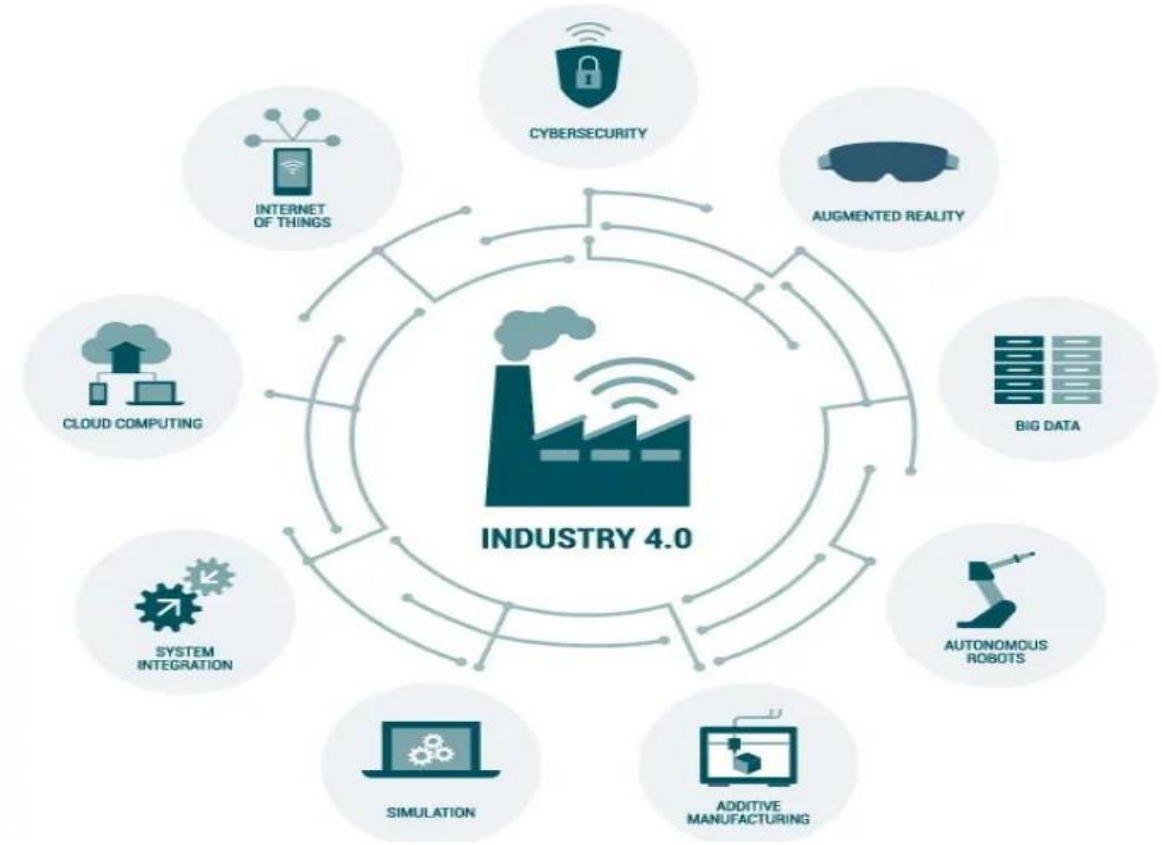
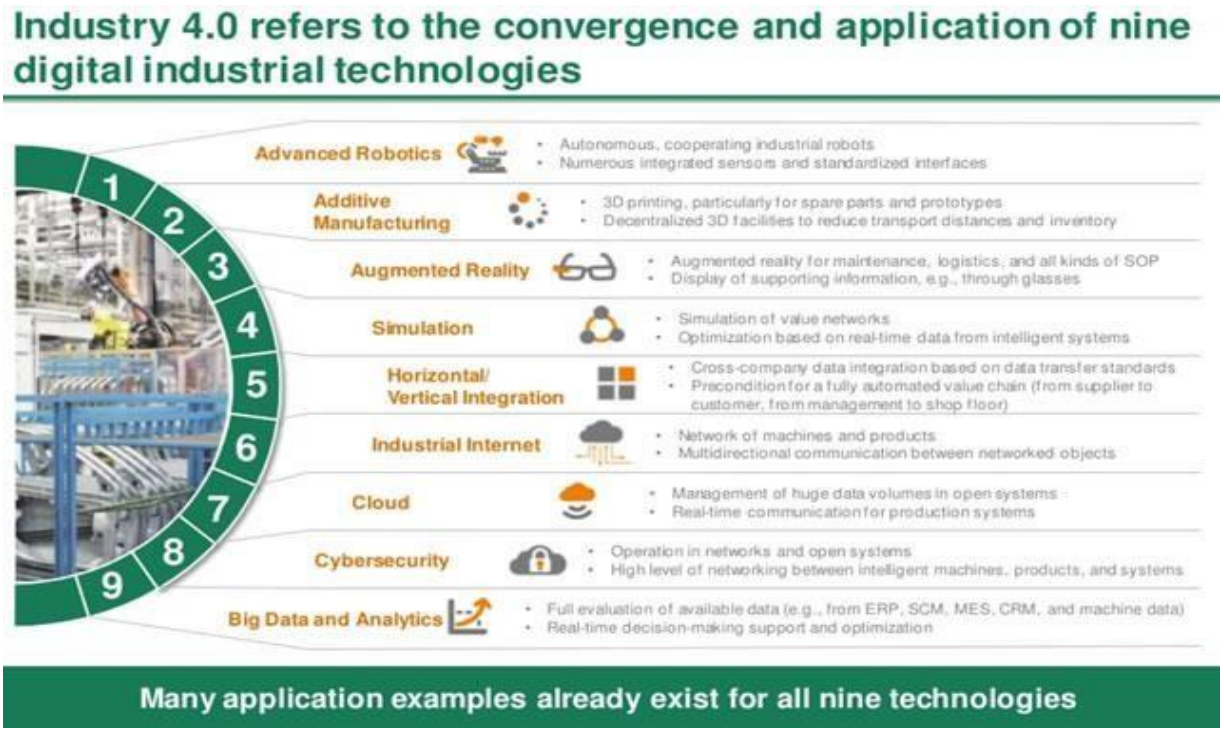


Figure: Nine Technologies of Industry 4.0



Today some companies have invested in a few of these technologies; predominantly the traditional pillars of the third platform such as cloud and Big Data / Analytics and increasingly in the Industrial Internet of Things from an integrated perspective and thus

overlapping with several of these “technologies” or maybe better: sets of technologies and connected benefits.

Evolution of Industry 4.0

There are four distinct industrial revolutions that the world either has experienced or continues to experience today.

1. The First Industrial Revolution

The first industrial revolution happened between the late 1700s and early 1800s. During this period of time, manufacturing evolved from focusing on manual labor performed by people and aided by work animals to a more optimized form of labor performed by people through the use of water and steam-powered engines and other types of machine tools.

2. The Second Industrial Revolution

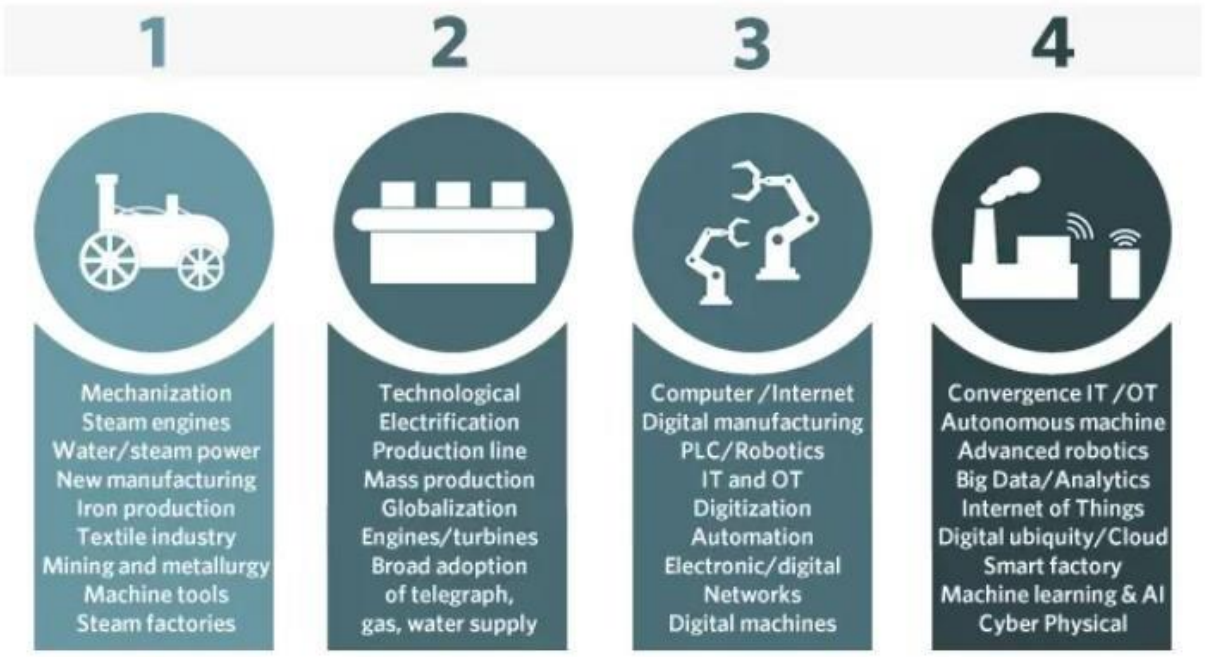
In the early part of the 20th century, the world entered a second industrial revolution with the introduction of steel and use of electricity in factories. The introduction of electricity enabled manufacturers to increase efficiency and helped make factory machinery more mobile. It was during this phase that mass production concepts like the assembly line were introduced as a way to boost productivity.

3. Third Industrial Revolution

Starting in the late 1950s, a third industrial revolution slowly began to emerge, as manufacturers began incorporating more electronic and eventually computer technology into their factories. During this period, manufacturers began experiencing a shift that put less emphasis on analog and mechanical technology and more on digital technology and automation software.

4. Fourth Industrial Revolution[Industry 4.0]

Fourth industrial revolution has emerged known as Industry 4.0. Industry 4.0 takes the emphasis on digital technology from recent decades to a whole new level with the help of interconnectivity through the Internet of Things (IoT), access to real-time data, and the introduction of cyber-physical systems. Industry 4.0 offers a more comprehensive,interlinked and holistic approach to manufacturing. It connects physical with digital, and allows for better collaboration and access across departments, partners, vendors, product, and people. An industry 4.0 empowers business owners to control and understand every aspect of their operation, and allows them to leverage instant data to boost productivity, improve processes, and drive growth.



Industry 4.0 is often used interchangeably with the notion of the fourth industrial revolution. It is characterized among others by

1) even more automation than in the third industrial revolution

2) the bridging of the physical and digital world through cyber-physical systems, enabledby Industrial IoT

3) a shift from a central industrial control system to one where smart products define the production steps

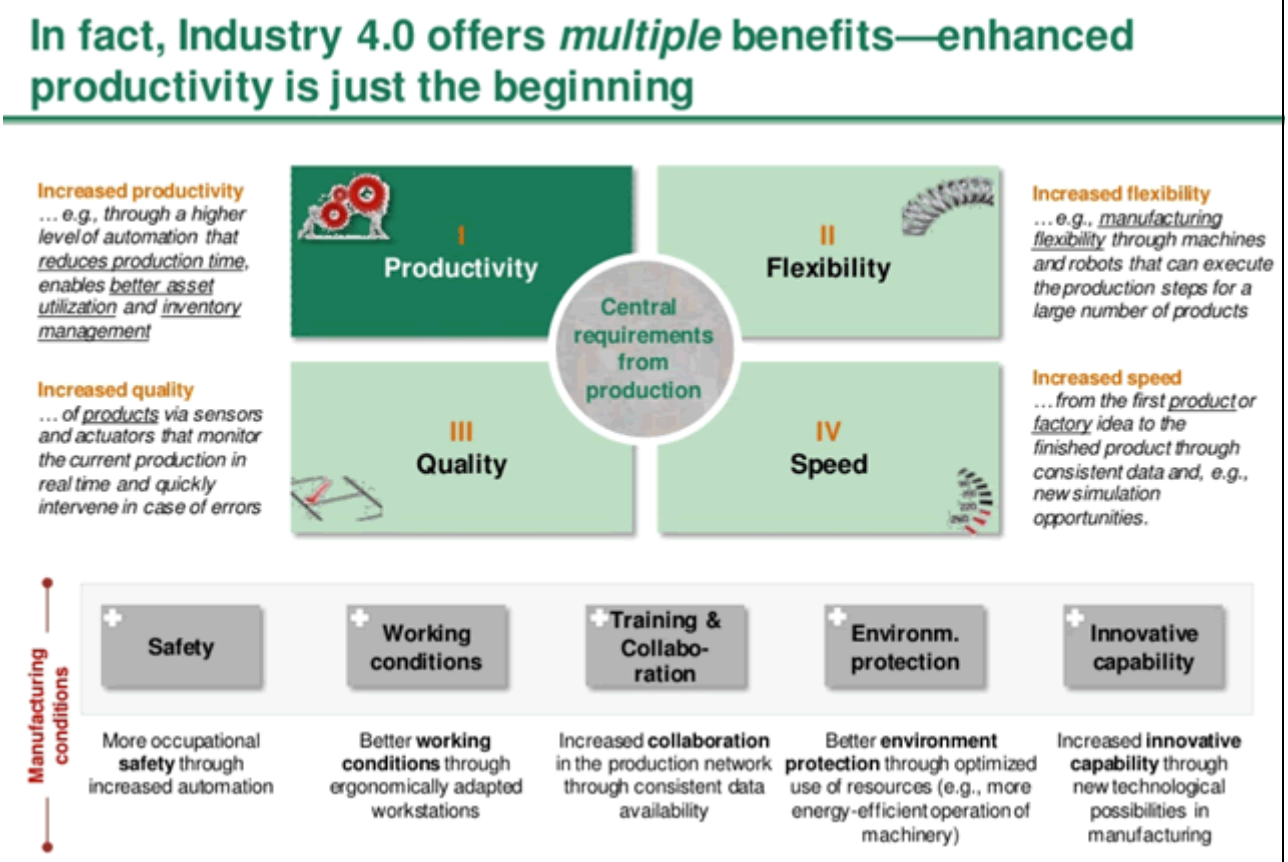
4) closed-loop data models and control systems and

5) personalization/customization of products.

Benefits of Industry 4.0

Industry 4.0 spans the entire product life cycle and supply chain, design, sales, inventory, scheduling, quality, engineering, and customer and field service. Everyone shares informed, up-to-date, relevant views of production and business processes and much richer and more timely analytics.

The essential goal of Industry 4.0 is to make manufacturing and related industries such as logistics faster, more efficient and more customer-centric, while at the same time going beyond automation and optimization and detect new business opportunities and models.



Most of the benefits of Industry 4.0 are obviously similar to the benefits of the digital transformation of manufacturing, the usage of the IoT in manufacturing, operational and business process optimization, information-powered ecosystems of value, digital transformation overall, the Industrial Internet and many other topics on our website. Few ofthe key benefits of Industry 4.0 are.

1. Enhanced productivity through optimization and automation

2. Real-time data for a real-time supply chain in a real-time economy

3. Higher business continuity through advanced maintenance and monitoring possibilities

4. Better quality products: real-time monitoring, IoT-enabled quality improvement andcobots

5. Better working conditions and sustainability

6. Personalization and customization for the ‘new’ consumer

7. Improved agility

8. The development of innovative capabilities and new revenue models

UNIT-5 Developing IoTs

Python

Python is a general-purpose high level programming language and suitable for providing a solid foundation to the reader in the area of cloud computing.

The main characteristics of Python are:

1) Multi-paradigm programminglanguage.
2) Python supports more than one programming paradigms including object- oriented programming and structured programming.
3) InterpretedLanguage.
4) Python is an interpreted language and does not require an explicit compilationstep.
5) The Python interpreter executes the program source code directly, statement by statement, as a processor or scripting engine does.
6) Interactive Language
7) Python provides an interactive mode in which the user can submit commands at the Python prompt and interact with the interpreterdirectly.

Python Benefits

- **Easy-to-learn, read and maintain**
  - Python is a minimalistic language with relatively few keywords, uses English keywords and has fewer syntactical constructions as compared to other languages. Reading Python programs feels like English with pseudo-code like constructs. Python is easy to learn yet an extremely powerful language for a wide range of applications.
- **Object and Procedure Oriented**
  - Python supports both procedure-oriented programming and object-oriented programming. Procedure oriented paradigm allows programs to be written around procedures or functions that allow reuse of code. Procedure oriented paradigm allows programs to be written around objects that include both data and functionality.
- **Extendable**
  - Python is an extendable language and allows integration of low-level modules written in languages such as C/C++. This is useful when you want to speed up a critical portion of a program.
- **Scalable**
  - Due to the minimalistic nature of Python, it provides a manageable structure for large programs.
- **Portable**
  - Since Python is an interpreted language, programmers do not have to worry about compilation, linking and loading of programs. Python programs can be directly executed from source
- **Broad Library Support**
  - Python has a broad library support and works on various platforms such as Windows, Linux, Mac, etc.

Python - Setup

- Windows
  - Python binaries for Windows can be downloaded from http://www.python.org/getit .
  - For the examples and exercise in this book, you would require Python 2.7 which can be directly downloaded from http://www.python.org/ftp/python/2.7.5/python-2.7.5.msi
  - Once the python binary is installed you can run the python shell at the command prompt using

    ```
    > python
    ```

- Linux

```
#Install Dependencies
sudo apt-get install build-essential
sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3-dev tk-dev libgdbm-dev  libc6-dev libbz2-dev

#Download Python
wget http://python.org/ftp/python/2.7.5/Python-2.7.5.tgz
tar -xvf Python-2.7.5.tgz
cd Python-2.7.5

#Install Python
./configure
make
sudo make install
```

Datatypes

Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

There are various data types in Python. Some of the important types are listed below.

Python Numbers

Integers, floating point numbers and complex numbers falls under Python numbers category. They are defined as int, float and complex class in Python. We can use the type() function to know which class a variable or a value belongs to and the isinstance() function to check if an object belongs to a particular class.

Script.py

```
1. a = 5
2. print(a, "is of type", type(a))
3. a = 2.0
4. print(a, "is of type", type(a))
5. a = 1+2j
6. print(a, "is complex number?", isinstance(1+2j,complex))
```

Integers can be of any length, it is only limited by the memory available. A floating point number is accurate up to 15 decimal places. Integer and floating points are separated by decimal points. 1 is integer, 1.0 is floating point number. Complex numbers are written in the form, x + yj, where x is the real part and y is the imaginary part. Here are someexamples.

```
>>> a = 1234567890123456789
>>> a
1234567890123456789
>>> b = 0.1234567890123456789
>>> b
0.12345678901234568
```

```
>>> c = 1+2j

>>> c

(1+2j)
```

Python List

List is an ordered sequence of items. It is one of the most used datatype in Python and is very flexible. All the items in a list do not need to be of the same type. Declaring a list is pretty straight forward. Items separated by commas are enclosed within brackets [].

```
>>> a = [1, 2.2, 'python']
```

We can use the slicing operator [ ] to extract an item or a range of items from a list. Index starts form 0 in Python.

Script.py

```
1. a = [5,10,15,20,25,30,35,40]
2. # a[2] = 15
3. print("a[2] = ", a[2])
4. # a[0:3] = [5, 10, 15]
5. print("a[0:3] = ", a[0:3])
6. # a[5:] = [30, 35, 40]
7. print("a[5:] = ", a[5:])
```

Lists are mutable, meaning; value of elements of a list can be altered.

```
>>> a = [1,2,3]
>>> a[2]=4
>>> a
[1, 2, 4]
```

Python Tuple

Tuple is an ordered sequences of items same as list. The only difference is that tuples are immutable. Tuples once created cannot be modified. Tuples are used to write-protect data and are usually faster than list as it cannot change dynamically. It is defined within parentheses () where items are separated bycommas.

```
>>> t = (5,'program', 1+3j)
```

Script.py

```
t = (5,'program', 1+3j)
# t[1] = 'program'
print("t[1] = ", t[1])
# t[0:3] = (5, 'program', (1+3j))
print("t[0:3] = ", t[0:3])
# Generates error
# Tuples are immutable
t[0] = 10
```

Python Strings

String is sequence of Unicode characters. We can use single quotes or double quotes to represent strings. Multi-line strings can be denoted using triple quotes, ''' or """.

```
>>> s = "This is a string"
>>> s = '''a multiline
```

Like list and tuple, slicing operator [ ] can be used with string. Strings are immutable.
Script.py

```
a ={5,2,3,1,4}
# printing setvariable
print("a = ", a)
# data type of variable a
print(type(a))
```

We can perform set operations like union, intersection on two sets. Set have unique values. They eliminate duplicates. Since, set are unordered collection, indexing has no meaning. Hence the slicing operator [] does not work. It is generally used when we have a huge amount of data. Dictionaries are optimized for retrieving data. We must know the key to retrieve the value. In Python, dictionaries are defined within braces {} with each item being a pair in the form key:value. Key and value can be of anytype.

```
>>> d = {1:'value','key':2}

>>> type(d)

<class 'dict'>
```

We use key to retrieve the respective value. But not the other way around.

Script.py

```
d ={1:'value','key':2}
print(type(d))
print("d[1] = ",d[1]);
print("d['key'] = ", d['key']);
# Generates error
print("d[2] = ",d[2]);
```

Python if...else Statement

Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes. Decision making is required when we want to execute a code only if a certain condition is satisfied.

The if…elif…else statement is used in Python for decision making.

Python if Statement

Syntax

```
if test expression:
   statement(s)
```

Here, the program evaluates the test expression and will execute statement(s) only if the text expression is True.
If the text expression is False, the statement(s) is not executed. In Python, the body of the if statement is indicated by the indentation. Body starts with an indentation and the first unindented line marks the end. Python interprets non-zero values as True. None and 0 are interpreted as False.

Python if Statement Flowchart

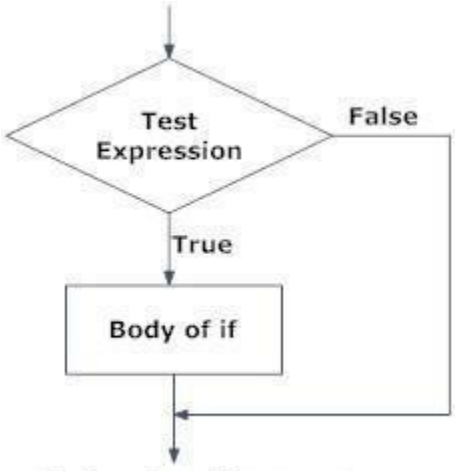


Fig: Operation of if statement

Example: Python if Statement

```
# If the number is positive, we print an appropriate message

num = 3
if num > 0:
   print(num, "is a positive number.")
print("This is always printed.")
num = -1
if num >0:
   print(num, "is a positive number.")
print("This is also always printed.")
```

When you run the program, the output willbe:
3 is a positivenumber
This is alwaysprinted
This is also always printed.

In the above example, num > 0 is the test expression. The body of if is executed only if this evaluates to True.
When variable num is equal to 3, test expression is true and body inside body of if is executed. If variable num is equal to -1, test expression is false and body inside body of if is skipped. The print() statement falls outside of the if block (unindented). Hence, it is executed regardlessof the testexpression.

Python if...else Statement

Syntax

```
if test expression:
   Body of if
else:
   Body of else
```

The if..else statement evaluates test expression and will execute body of if only when test condition is True.
If the condition is False, body of else is executed. Indentation is used to separate the blocks.

Python if..else Flowchart

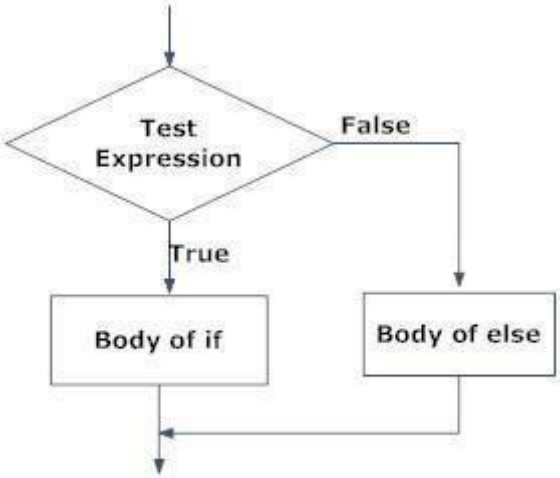


Fig: Operation of if...else statement

Example of if...else

```
# Program checks if the number is positive or negative
# And displays an appropriate message
num = 3
# Try these two variations as well.
# num = -5
# num = 0
if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

In the above example, when num is equal to 3, the test expression is true and body of if is executed and body of else is skipped.

If num is equal to -5, the test expression is false and body of else is executed and body of if is skipped.

If num is equal to 0, the test expression is true and body of if is executed and body of else is skipped.

Python if...elif...else Statement

Syntax

```
if test expression:
   Body of if
elif test expression:
   Body of elif
else:
   Body of else
```

The elif is short for else if. It allows us to check for multiple expressions. If the condition for if is False, it checks the condition of the next elif block and so on. If all the conditions are False, body of else is executed. Only one block among the several if...elif...else blocks is executed according to the condition. The if block can have only one else block. But it can have multiple elifblocks.

Flowchart of if...elif...else

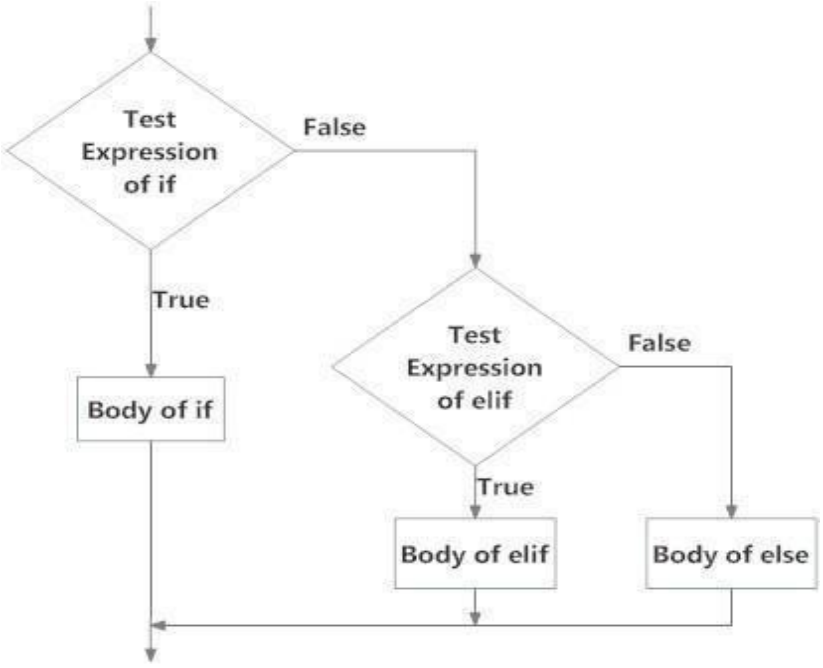


Fig: Operation of if...elif...else statement

Example of if...elif...else

```
# In this program,
# we check if the number is positive or
# negative or zero and
# display an appropriate message
num = 3.4
```

```
# Try these two variations as well:
# num = 0
# num = -4.5
if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

When variable num is positive, Positive number is printed.

If num is equal to 0, Zero is printed.
If num is negative, Negative number is printed

Python Nested if statements

We can have a if...elif...else statement inside another if...elif...else statement. This is called nesting in computer programming. Any number of these statements can be nested inside one another. Indentation is the only way to figure out the level of nesting. This can get confusing, so must be avoided if we can.

Python Nested if Example

```
# In this program, we input a number
# check if the number is positive or
# negative or zero anddisplay
# an appropriate message
# This time we use nested if

num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

Output 1

Enter a number: 5
Positive number
Output 2
Enter a number: -1

Negative number
Output 3
Enter a number: 0
Zero

Python for Loop

The for loop in Python is used to iterate over a sequence (list, tuple, string) or other iterable objects. Iterating over a sequence is called traversal.

Syntax of for Loop

```
for val in sequence:
        Body of for
```

Here, val is the variable that takes the value of the item inside the sequence on each iteration. Loop continues until we reach the last item in the sequence. The body of for loop is separated from the rest of the code using indentation.

Flowchart of for Loop

Syntax

```
# Program to find the sum of all numbers stored in a list
# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]
# variable to store the sum
sum = 0

# iterate over the list
for val in numbers:
        sum =   sum+val
# Output: The sum is 48
print("The sum is", sum)
```

when you run the program, the output will be:
The sum is 48

The range() function

We can generate a sequence of numbers using range() function. range(10) will generate numbers from 0 to 9 (10 numbers). We can also define the start, stop and step size as range(start,stop,step size). step size defaults to 1 if not provided. This function does not store all the values in emory, it would be inefficient. So it remembers the start, stop, step size and generates the next number on thego.

To force this function to output all the items, we can use the function list().

The following example will clarify this.

```
# Output: range(0, 10)
print(range(10))
# Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(list(range(10)))
```

```
# Output: [2, 3, 4, 5, 6, 7]
print(list(range(2, 8)))
# Output: [2, 5, 8, 11, 14, 17]
print(list(range(2, 20, 3)))
```

We can use the range() function in for loops to iterate through a sequence of numbers. It can be combined with the len() function to iterate though a sequence using indexing. Here is anexample.

```
# Program to iterate through a list using indexing
genre = ['pop', 'rock', 'jazz']
# iterate over the list using index
for i in range(len(genre)):
        print("I like", genre[i])
```

When you run the program, the output will be:

```
I likepopI
likerock I
likejazz
```

What is while loop in Python?

The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true. We generally use this loop when we don't know beforehand, the number of times to iterate.

Syntax of while Loop in Python

```
while
   test_expression:
   Body of while
```

In while loop, test expression is checked first. The body of the loop is entered only if the test_expression evaluates to True. After one iteration, the test expression is checked again. This process continues until the test_expression evaluates to False. In Python, the body of the while loop is determined through indentation. Body starts with indentation and the first unindented line marks the end. Python interprets any non-zero value as True. None and 0 are interpreted asFalse.

Flowchart of while Loop

```
# Program to add
natural # numbers
upto
# sum = 1+2+3+...+n
# To take input from the
user, # n = int(input("Enter
n: "))
n = 10
# initialize sum and
counter sum = 0
i = 1
while i <= n:
```

```
    sum = sum +
    i
    i=i+1     #
updatecounter # print
thesum
print("The sum is", sum)
```

When you run the program, the output will be:

```
Enter n: 10
The sum is
55
```

In the above program, the test expression will be True as long as our counter variable i is less than or equal to n (10 in ourprogram).
We need to increase the value of counter variable in the body of the loop. This is very important (and mostly forgotten). Failing to do so will result in an infinite loop (never ending loop).
Finally the result is displayed.

Python Modules

A file containing a set of functions you want to include in the application is called Module.

Create a Module

To create a module just save the code you want in a file with the file extension .py:

Example

Save this code in a file named mymodule.py

```
def greeting(name):
  print("Hello, " + name)
```

Use a Module

Now we can use the module we just created, by using the import statement:

Example

Import the module named mymodule, and call the greeting function:

```
import mymodule
mymodule.greeting("Jonathan")
```

Note: When using a function from a module, use the syntax: module_name.function_name.
Variables in Module

The module can contain functions, as already described, but also variables of all types(arrays, dictionaries, objects etc):

Example

Save this code in the file mymodule.py

```
person1 = {"name": "John","age": 36,"country": "Norway"}
```

Example

Import the module named mymodule, and access the person1 dictionary:

```
import mymodule
a = mymodule.person1["age"]
print(a)
```

Naming a Module

You can name the module file whatever you like, but it must have the file extension .py

Re-naming a Module

You can create an alias when you import a module, by using the as keyword:

Example

Create an alias for mymodule called mx:

```
import mymodule as mx
a = mx.person1["age"]
print(a)
```

Built-in Modules

There are several built-in modules in Python, which you can import whenever you like.

Example

Import and use the platform module:

```
import platform
x = platform.system()
print(x)
```

Using the dir() Function

There is a built-in function to list all the function names (or variable names) in a module. The dir() function:

Example

List all the defined names belonging to the platform module:

```
import platform
```

```
x = dir(platform)
 print(x)
```

Note: The dir() function can be used on all modules, also the ones you create yourself.

Import from Module

You can choose to import only parts from a module, by using the from keyword.

Example

The module named mymodule has one function and one dictionary:

```
def greeting(name):
print("Hello, " + name)
person1 = {"name": "John", "age": 36, "country": "Norway"}
```

Example

Import only the person1 dictionary from the module:

```
from mymodule import person1
print (person1["age"])
```

Note: When importing using the from keyword, do not use the module name when referring to elements in the module. Example: person1["age"], not mymodule.person1["age"].

Packages

We don't usually store all of our files in our computer in the same location. We use a well-organized hierarchy of directories for easier access. Similar files are kept in the same directory, for example, we may keep all the songs in the "music" directory. Analogous to this, Python has packages for directories and modules for files. As our application program grows larger in size with a lot of modules, we place similar modules in one package and different modules in different packages. This makes a project (program) easy to manage and conceptuallyclear.

Similar, as a directory can contain sub-directories and files, a Python package can have sub-packages and modules. A directory must contain a file namedinit.py in order for Python to consider it as a package. This file can be left empty but we generally place the initialization code for that package in this file. Here is an example. Suppose we are developing a game, one possible organization of packages and modules could be as shown in the figure below.

Package Module Structure in Python Programming
Importing module from a package

We can import modules from packages using the dot (.) operator. For example, if want to import the start module in the above example, it is done as follows.

```
import Game.Level.start
```

Now if this module contains a function named select_difficulty(), we must use the full name to reference it.

```
Game.Level.start.select_difficulty(2)
```

If this construct seems lengthy, we can import the module without the package prefix as follows.
```
from Game.Level import start
```

We can now call the function simply as follows.

```
start.select_difficulty(2)
```

Yet another way of importing just the required function (or class or variable) form a module within a package would be as follows.

```
from Game.Level.start import select_difficulty
```

Now we can directly call this function.

```
select_difficulty(2)
```

Although easier, this method is not recommended. Using the full namespace avoids confusion and prevents two same identifier names from colliding. While importing packages, Python looks in the list of directories defined in sys.path, similar as for module search path.

Files

File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g. hard disk). Since, random access memory (RAM) is volatile which loses its data when computer is turned off, we use files for future use of the data. When we want to read from or write to a file we need to open it first. When we are done, it needs to be closed, so that resources that are tied with the file are freed. Hence, in Python, a file operation takes place in the followingorder.

1. Open afile
2. Read or write (perform operation)
3. Close thefile

How to open a file?

Python has a built-in function open() to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.

```
>>> f=open("test.txt")      # open file in currentdirectory
>>> f = open("C:/Python33/README.txt") # specifying full path
```

We can specify the mode while opening a file. In mode, we specify whether we want to read 'r', write 'w' or append 'a' to the file. We also specify if we want to open the file in text mode or binary mode. The default is reading in text mode. In this mode, we get strings when reading fromthe file. On the other hand, binary mode returns bytes and this is the mode to be used when dealing with non-text files like image or exe files.

Python File Modes

Mode	Description
'r'	Open a file for reading. (default)
'w'	Open a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
'x'	Open a file for exclusive creation. If the file already exists, the operation fails.
'a'	Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.
't'	Open in text mode. (default)
'b'	Open in binary mode.
'+'	Open a file for updating (reading and writing)

```
f=open("test.txt")        # equivalent to 'r' or 'rt'
f = open("test.txt",'w') # write in textmode
f = open("img.bmp",'r+b') # read and write in binary mode
```

Unlike other languages, the character 'a' does not imply the number 97 until it is encoded using ASCII (or other equivalent encodings). Moreover, the default encoding is platform dependent. In windows, it is 'cp1252' but 'utf-8' in Linux. So, we must not also rely on the default encoding or else our code will behave differently in different platforms. Hence, when working with files in text mode, it is highly recommended to specify the encoding type.

```
f = open("test.txt",mode = 'r',encoding = 'utf-8')
```

How to close a file Using Python?

When we are done with operations to the file, we need to properly close the file. Closing a file will free up the resources that were tied with the file and is done using Python close() method. Python has a garbage collector to clean up unreferenced objects but, we must not rely on it to close the file.

```
f = open("test.txt",encoding = 'utf-8')
```

```
# perform file operations
f.close()
```

This method is not entirely safe. If an exception occurs when we are performing some operation with the file, the code exits without closing the file.

A safer way is to use a try...finally block.

```
try:
   f = open("test.txt",encoding = 'utf-8')
   # perform file operations
finally:
   f.close()
```

This way, we are guaranteed that the file is properly closed even if an exception is raised, causing program flow to stop. The best way to do this is using the with statement. This ensures that the file is closed when the block inside with is exited. We don't need to explicitly call the close() method. It is doneinternally.

```
with open("test.txt",encoding = 'utf-8') as f:
   # perform file operations
```

How to write to File Using Python?

In order to write into a file in Python, we need to open it in write 'w', append 'a' or exclusive creation 'x' mode. We need to be careful with the 'w' mode as it will overwrite into the file if it already exists. All previous data are erased. Writing a string or sequence of bytes (for binary files) is done using write() method. This method returns the number of characters written to the file.

```
with open("test.txt",'w',encoding = 'utf-8') as f:
   f.write("my first file\n")
   f.write("This file\n\n")
   f.write("contains three lines\n")
```

This program will create a new file named 'test.txt' if it does not exist. If it does exist, it is overwritten. We must include the newline characters ourselves to distinguish different lines.

How to read files in Python?

To read a file in Python, we must open the file in reading mode. There are various methods available for this purpose. We can use the read(size) method to read in size number of data. If size parameter is not specified, it reads and returns up to the end of the file.

```
>>> f = open("test.txt",'r',encoding = 'utf-8')
>>> f.read(4) # read the first 4 data
```

'This'

>>>f.read(4) # read the next 4 data
' is'

>>>f.read() # read in the rest till end of file
'my first file\nThis file\ncontains threelines\n'

>>> f.read() # further reading returns empty sting
''

We can see that, the read() method returns newline as '\n'. Once the end of file is reached, we get empty string on further reading. We can change our current file cursor (position) using the seek() method. Similarly, the tell() method returns our current position (in number of bytes).

>>>f.tell() # get the current file position
56

>>> f.seek(0) # bring file cursor to initial position
0

>>> print(f.read()) # read the entire file
This is my first file
This file
contains three lines

We can read a file line-by-line using a for loop. This is both efficient and fast.

>>> for line in f:
... print(line, end = '')
...
This is my first file
This file
contains three lines
The lines in file itself has a newline character '\n'.

Moreover, the print() end parameter to avoid two newlines when printing. Alternately, we can use readline() method to read individual lines of a file. This method reads a file till the newline, including the newlinecharacter.

>>> f.readline()
'This is my first file\n'

>>> f.readline()
'This file\n'

```
>>> f.readline()
'contains three lines\n'

>>> f.readline()
''
```

Lastly, the readlines() method returns a list of remaining lines of the entire file. All these reading method return empty values when end of file (EOF) is reached.

```
>>> f.readlines()

['This is my first file\n', 'This file\n', 'contains three lines\n']
```

Python File Methods

There are various methods available with the file object. Some of them have been used in above examples. Here is the complete list of methods in text mode with a brief description.

Python File Methods

Method	**Description**
close()	Close an open file. It has no effect if the file is already closed.
detach()	Separate the underlying binary buffer from the TextIOBase and return it.
fileno()	Return an integer number (file descriptor) of the file.
flush()	Flush the write buffer of the file stream.
isatty()	Return True if the file stream is interactive.
read(n)	Read at most n characters form the file. Reads till end of file if it is negative or None.
readable()	Returns True if the file stream can be read from.
readline(n=-1)	Read and return one line from the file. Reads in at most n bytes if specified.
readlines(n=-1)	Read and return a list of lines from the file. Reads in at most n bytes/characters if specified.
seek(offset,from=SE EK_SET)	Change the file position to offset bytes, in reference to from (start, current, end).
seekable()	Returns True if the file stream supports random access.
tell()	Returns the current file location.
truncate(size=None)	Resize the file stream to size bytes. If size is not specified, resize to current location.
writable()	Returns True if the file stream can be written to.
write(s)	Write string s to the file and return the number of characters written.
writelines(lines)	Write a list of lines to the file.

Method Description
close() Close an open file. It has no effect if the file is already closed.
detach() Separate the underlying binary buffer from the TextIOBase and returnit.
fileno()Return an integer number (file descriptor) of thefile.
flush() Flush the write buffer of the file stream.
isatty() Return True if the file stream is interactive.
read(n) Read at most n characters form the file.

Reads till end of file if it is negative or None.

readable() Returns True if the file stream can be read from.
readline(n=-1) Read and return one line from the file. Reads in at most n bytes if specified.
readlines(n=-1) Read and return a list of lines from the file. Reads in at most n bytes/characters if specified.
seek(offset,from=SEEK_SET) Change the file position to offset bytes, in reference to from (start, current,end).
seekable() Returns True if the file stream supports random access.
tell() Returns the current filelocation.
truncate(size=None) Resize the file stream to size bytes. If size is not specified, resize to currentlocation.
writable() Returns True if the file stream can be written to.
write(s) Write string s to the file and return the number of characters written.
writelines(lines) Write a list of lines to the file.