1

IOT-AN ARCHITECTURAL OVERVIEW

Unit Structure :

1.0 OBJECTIVES

- To understand the working of an IoT system and components
- To get familiar with the building blocks and the architectural functioning mechanism of IoT
- To acquaint with the design principles and considerations when developing an IoT prototype

1.1 INTRODUCTION

What is Internet of Things? - the idea of connecting any gadget to the Internet and other linked devices (as long as it has an on/off switch). The Internet of Things (IoT) is a vast network of interconnected devices and people, all of which gather and exchange information about their environments and how they are used.

How does it function? Connected to an Internet of Things platform, which combines data from many devices and applies analytics to share the most useful information with applications created to answer particular needs, are gadgets and objects having built-in sensors.

These robust IoT solutions can precisely identify which information is helpful and which may be safely disregarded. This data can be used to identify trends, generate recommendations, and identify potential issues before they arise.

For instance, a company that makes cars might want to know which add-ons, like leather seats or alloy wheels, are the most popular. Technology based on the Internet of Things makes it feasible to:

1] Employ sensors to identify which showroom spaces are the busiest and where clients remain the longest;

2] Analyze the sales data to determine which components are selling the quickest;

3] Automatically match supply and sales data to ensure that in-demand items don't run out of stock.

Making informed decisions about which components to stock up on based on real-time information using the data collected by linked devices helps save time and money.

The ability to improve procedures comes with the insight sophisticated analytics offers. You can automate some jobs thanks to smart devices and systems, especially if they are monotonous, time-consuming, repetitive, or even hazardous.

1.2 BUILDING ARCHITECTURE

The IoT system's fundamental building parts include sensors, processors, gateways, and applications. To create a useful IoT system, each of these nodes must have unique properties.

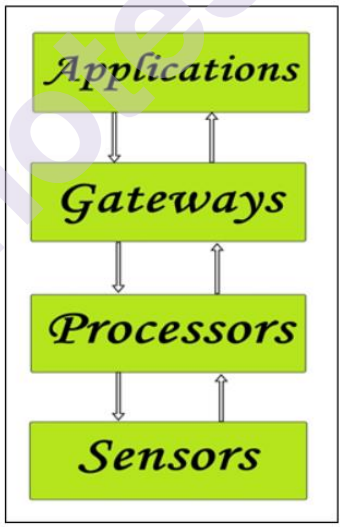


Figure 1: Simplified block diagram of the basic building blocks of the IoT

1] Sensors

- They make up the IoT devices' front end. These are the system's purported "Things." Their primary function is to gather data from their environment (sensors) or to disseminate data to their environment (actuators).
- To be easily recognized across a wide network, these must be uniquely recognizable devices having a unique IP address.

- They must be active, which means they must be able to gather data in real time. Depending on the user's demands, these can either function independently (autonomous in nature) or be modified to function independently (user-controlled).

- Gas sensors, water quality sensors, moisture sensors, and other types of sensors are examples.

2] Processors

- The IoT system's brain is its processor. Their primary duty is to process the information obtained by the sensors and separate the useful information from the vast amounts of raw information gathered. In a single sentence, we may claim that it offers the data intelligence.

- Most processors operate in real-time and are simple for programs to regulate. They are also in charge of encrypting and decrypting data in order to secure the data.

- Because they have processors attached to them, embedded hardware devices, microcontrollers, etc., are the ones that process the data.

3] Gateways

- Gateways are in charge of sending the processed data to the appropriate areas for proper utilization.

- In other words, we can say that a gateway facilitates the communication of data between two points. It gives the data network connectivity. Any IoT system must have network connectivity in order to interact.

- Network gateways include LAN, WAN, PAN, etc.

4] Applications

- Another component of an IoT system is applications. Apps are necessary for the effective use of all obtained data.

- These cloud-based applications are in charge of giving the obtained data an effective meaning. Users control applications, which are used to deliver certain services.

- Applications include things like security systems, industrial control hubs, and apps for home automation.

The far right component in Figure 2 represents the application end of the Internet of Things architecture.

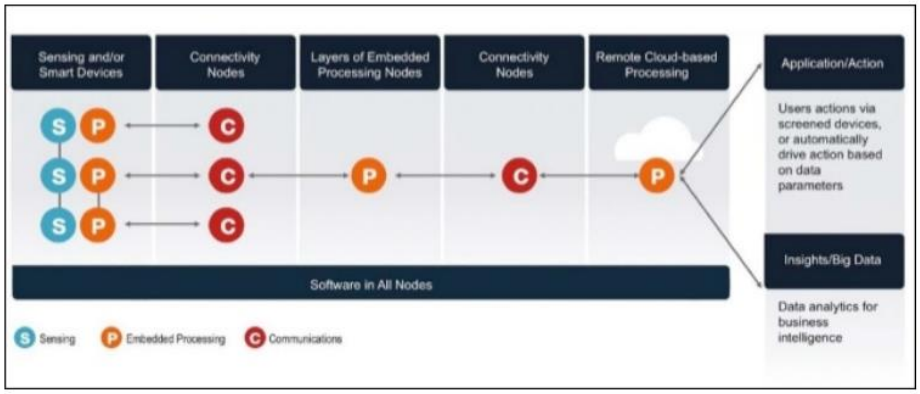


Figure 2: Basic building blocks of IoT

The information obtained by the sensing node (end node) is processed initially, and then via connectivity it reaches the embedded processing nodes which can be any embedded hardware devices where it is processed again. The data is transferred to the application node for proper application of the acquired data as well as for data analysis via big data after passing via the connectivity nodes once more. The remote cloud-based processing can be any software at this point.

1.3 MAIN DESIGN PRINCIPLES AND NEEDED CAPABILITIES

Designing IoT solutions presents whole new design difficulties for designers that are primarily focused on building SW services, screen-based user interfaces, or physical goods. IoT solutions are made up of several components, including physical devices like sensors, actuators, and interactive devices, the network that connects them, the data collected from these devices and analyzed to produce a meaningful experience, and last but not least, the actual physical environment in which the user interacts with the solution. You must do a variety of design tasks, including service and business design as well as industrial product design. The whole user experience (UX) of the IoT system is influenced by all of these aspects, and designing in this environment may seem fairly daunting. Following points represents the design principles and considerations of IoT.

1] Focus on value

User research and service design are more important than ever in the IoT age. Early adopters are eager to test out new technology, while many others are hesitant to do so and cautious when using it because they lack confidence in it. You must go deeply into user demands to identify where a problem actually merits solving and what the solution's true end user value is if you want your IoT solution to be broadly embraced. Also, you need to be aware of any potential obstacles to the adoption of your particular solution as well as new technology in general. You also need to conduct study to choose your feature set. You must carefully consider which features to include and in what order, as things that might be valuable and highly relevant for

tech early adopters may not be appealing to the majority of consumers and vice versa.

2] Take a holistic view

IoT solutions frequently include both physical and digital touchpoints, as well as a variety of devices with various capabilities. The answer might also be offered in conjunction with a variety of other service providers. It is not sufficient to effectively design just one of the touchpoints; rather, you must consider the entire system, the function of each device and service, and the conceptual model of how the user understands and perceives the system. To produce a memorable experience, the entire system must operate without a hitch.

3] Put safety first

IoT solutions are used in the real world, so when something goes wrong, the repercussions could be severe. Building trust should be one of your major design drivers because consumers of IoT solutions may have different comfort levels with new technology. You must take care to ensure that every interaction with the product or service strengthens rather than undermines the trust because it is established gradually and lost easily. What does it actually mean? Understanding potential mistake scenarios connected to the use environment, hardware, software, and network, as well as user interactions, is the first step in trying to prevent them. The user must be properly informed about mistake circumstances and assisted in recovering if they continue to occur. Second, it involves making data security and privacy important design components. Users must have the confidence that their personal information is secure, that their homes, places of employment, and ordinary items cannot be compromised, and that their loved ones are not in danger. Thirdly, quality assurance is essential, and it should concentrate on evaluating the entire system in a real-world setting rather than just the SW.

4] Consider the text

At the nexus of the physical and digital worlds are IoT solutions. Digital interface commands may have real-world consequences, but unlike digital commands, real-world consequences often cannot be reversed. Many unanticipated events can occur in the real world, but users still need to feel secure and in control. Several kinds of criteria for the design are also imposed by the context. Depending on the physical environment, the objective can be to reduce user distraction or, for example, to design equipment that can withstand changing weather conditions. IoT solutions are often multi-user systems in homes, offices, and public spaces, making them less personal than, say, screen-based solutions used in smartphones. This also considers the social context in which the solution is utilized and its design needs.

5] Build a strong band

No matter how carefully you design things and try to establish trust, something unexpected will happen at some point and your solution is going to fail in some way because of the real-world environment of IoT solutions. It is crucial in times like this that you have developed a powerful brand that connects with them on an authentic level. They will be more understanding of system flaws and continue to use your solution if they feel a connection to your brand. Trust should be a crucial component of your brand and one of its basic brand principles. This is something you must keep in mind while you create your brand. This core principle should be mirrored in all other aspects of the brand, such as color scheme, writing style, images, etc.

6] Prototype early and often

Normally, HW and SW have lifespans that are somewhat dissimilar, but since a successful IoT solution requires both HW and SW components, the lifespans should be coordinated. IoT solutions are also difficult to upgrade because once a connected object is installed, it is difficult to replace it with a newer model, especially if the user must pay for the upgrade. Moreover, the connected object's software may be difficult to update for security and privacy concerns. It's essential to get the solution right from the start of implementation due to these factors and to prevent expensive hardware iterations. From a design standpoint, this means that early project stages require quick prototyping and iteration of both the HW and the entire solution. We need new, inventive approaches to fake the solution and prototype it.

7] Use data responsibly

IoT systems can potentially produce enormous amounts of data. The goal is to discover the data points required to make the solution work and be valuable, not to collect as much data as you can. The designer must comprehend the potential of data science and how to interpret the data because the volume of data may be enormous. Data science offers several chances to lower user friction, i.e., to consume less time, energy, and attention, or to experience less stress. It can be used to understand intent from partial or insufficient input, to automate repetitive context-dependent judgements, to filter out noise from relevant signals, and more. Designing successful IoT services requires a thorough understanding of the data that is available and how it can be used to benefit the user.

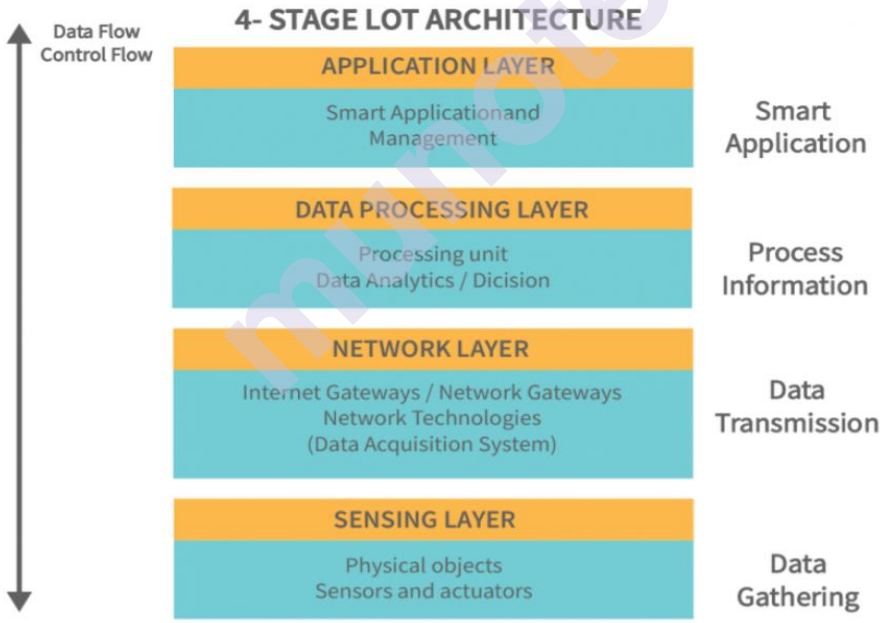
1.4 AN IOT ARCHITECTURE OUTLINE

The complex arrangement of elements that make up IoT networking systems, including sensors, actuators, cloud services, protocols, and layers, is referred to as IoT architecture. It is typically separated into layers that let administrators assess, keep an eye on, and uphold the integrity of the system. Data moves from connected devices to sensors, through a network,

to the cloud for processing, analysis, and storage in a four-step process known as the IoT architecture. The Internet of Things is poised to expand much further with time, offering users fresh and enhanced experiences.

Different layers of IoT architecture

IoT technology has been more well-liked recently and has a wide range of uses. IoT apps function in accordance with how they were created depending on the many application domains. There isn't a set standard defined architecture of work, nevertheless, that is rigidly followed everywhere. Depending on the particular business job at hand, different architectural layers and levels of complexity are used. The most common and standard architecture is a four-layer one.



As you can see from the above image, there are four layers present i.e., the Perception Layer, Network Layer, Processing Layer, and Application Layer.

1] Perception/ Sensing layer

Any IoT system's first layer is made up of "things" or endpoint devices that act as a link between the real world and the digital one. The

physical layer, which contains sensors and actuators capable of gathering, accepting, and processing data across a network, is referred to as perception. Wireless or wired connections can be used to connect sensors and actuators. The components' range and locations are not constrained by the design.

2] Network layer

An overview of the data flow throughout the programme is given by the network layers. Data Acquisition Systems (DAS) and Internet/Network gateways are present in this tier. Data aggregation and conversion tasks are carried out by a DAS (collecting and aggregating data from sensors, then converting analogue data to digital data, etc.). Data gathered by the sensor devices must be transmitted and processed. The network layer performs that function. It enables connections and communication between these gadgets and other servers, smart gadgets, and network gadgets. Also, it manages each device's data transmission.

3] Processing layer

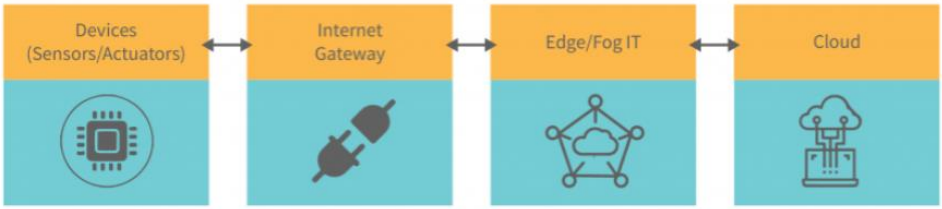
The IoT ecosystem's processing layer functions as its brain. Before being transported to the data center, data is typically evaluated, pre-processed, and stored here. It is then retrieved by software applications that handle the data and prepare future actions. This is where edge analytics or edge IT comes into play.

4] Application layer

The application layer, which provides the user with application-specific services, is where user interaction occurs. A dashboard that displays the status of the devices in a system or a smart home application where users may turn on a coffee maker by touching a button in an app are two examples. The Internet of Things can be used in a variety of applications, including smart homes, smart cities, and smart health.

Stages of IoT solutions architecture

How can organizations take advantage of the IoT layers after learning about them and how can they increase the value of IoT? Although linked devices and protocols are referred to as part of the Internet of Things (IoT), the data produced by these devices is actually siloed, fragmented, and isolated. As a result, these fragmented insights do not alone offer sufficient data to support an IoT strategy that entails a large resource investment. Enterprises must leverage device and system synergies and allow devices to freely interact in order to benefit from IoT. Make sure your infrastructure is compatible with the IoT architecture. The various phases of IoT architecture implementation in businesses are as follows:



- **Connected objects/devices**

  The physical layer within the environment must be built as a first step towards IoT architecture. The Internet of Things would not exist without "smart" or linked objects. On the perception layer, these are frequently wireless sensors or actuators.

  Sensors gather and process environmental data to make it useful for additional research. The change that the sensors register is measured by actuators. Wired or wireless connections can be made between sensors and actuators to accomplish sensing and actuation. Sensors and actuators can be connected using Personal Area Networks (PANs) and Local Area Networks (LANs).

- **Internet gateway**

  After properly completing step one, the next task is to set up an internet gateway. We need a way to convert analogue data that is being collected by the sensors and actuators into digital data so that we can process it. The internet gateway is used to do this activity. Before being transferred to the cloud, raw data from the devices will be received at the internet gateway stage and pre-processed.

  Analog data can be transformed into digital data using data acquisition systems. It establishes connections with the sensors and actuators, collects all the data, and transforms it into digital form so that the internet gateway may send it across the network. It is in charge of conversion and data aggregation. To improve performance and effectiveness, we can also add extra features like analytics and security.

- **Edge IT systems**

  Pre-processing and improved data analytics are part of the third stage of an IoT architecture. Edge IT systems are essential in easing the burden on the main IT infrastructure due to the sizeable volume of data collected by IoT systems and the ensuing bandwidth requirements. Machine learning and visualization techniques are used by edge IT systems to derive insights from gathered data. While visualization tools make the data more comprehensible, machine learning algorithms offer insights into the data.

  The system's speed, as well as the LAN or routers' bandwidth, would suffer if data is sent directly to the server or data center. Analog data is produced very quickly and takes up a lot of storage space. As a

result, it is always advised to convert data to digital format. Only the necessary data is processed and communicated to data centers and servers because the majority of the data acquired by sensors and actuators is not valuable to the enterprise.

- **Data centers and cloud storage**

  The data is delivered to the data centers and servers for final analysis and reporting once it has been appropriately preprocessed, examined, and any gaps have been filled. The management services area includes data centers and cloud services, which often handle data using analytics, device management, and security controls. Data can also be transferred to end-user applications like healthcare, retail, environment, emergency, energy, etc. thanks to the cloud.

The data might be transmitted to data centers or cloud-based servers for final processing after analysis. Hardware expenses can be reduced by using the cloud platform, but data security is still a worry. Physical servers and data centers are safer, but they are also more expensive.

1.5 STANDARDS CONSIDERATIONS

Manufacturers who want to maintain their competitiveness in their sector must connect their devices to the Internet of Things (IoT). IoT capabilities expand the options available to users. Additionally, it enables the manufacturer to maintain contact with their clientele while they explore new product use cases and applications that present them with opportunities for new revenue streams. There are ten considerations to make while creating your first Internet of Things device:

1] Cost

IoT or "smart" products benefit producers and customers equally, although they are more expensive. Consider networking in your next product because both Ethernet and wireless technology have dropped below $10.

2] Network

The network technology you choose for your IoT product has concerns with gateways and routers as well as distance. Ethernet/Wi-Fi is required if you need to connect to the Internet; ZigBee, Z-Wave, and Bluetooth are available if you are self-contained in a room or building. Remember that the FCC must approve all wireless technology.

3] Features

Businesses may now add features to their products that were either impossible or unimaginable without an IoT-connected product. For updates, maintenance, and new revenue opportunities, you can obtain direct access to the consumer with the help of these capabilities.

4] User interface

It matters how a user interacts with a product. On the product, are you intending to use buttons, LEDs, or a display? What web and app interfaces will you offer as well?

5] Power

The choice of a power source should be among the first. All design choices must take power conservation into account if the item will be battery-powered. Many networking technologies won't operate well on batteries. Power selection is also influenced by communication frequency.

6] Size

Size does matter. Think about how the size of the device will be affected by the network. Certain networks' requirements for connectors and antennae will increase the size.

7] Antenna

Whether inside or exterior to the product, an antenna is used by all wireless networks. If the enclosure is plastic, the antenna is increasingly being moved inside. External antennas would be necessary for all metal enclosures.

8] Cloud

Products have a user interface to the product and the data thanks to cloud applications. There are public clouds and private clouds. Most clouds have a common API that you may use to create your application.

9] Interoperability

Is communication between your product and those of other vendors required? If so, you must use a common set of protocols, like Apple's HomeKit, to interact with other devices.

10] Security

You must incorporate as many layers of security as you can because security is starting to become a serious concern. The bare minimum is SSL and a password.

SUMMARY

Rapid technological development in the modern era has connected people and things worldwide. IoT solutions have ingrained themselves into our daily lives in recent years. For instance, you can instantly get a response or results by simply speaking or tapping the screen of your smartphone. Regardless of how IoT architecture varies from project to project, managing

vast amounts of data will always be a crucial component of every IoT project.

Enterprises can automate business operations by employing technology like cloud platforms, embedded devices with sensors and actuators, and internet-based communication. The insights drawn from IoT data sets will become a useful source of information for businesses thanks to big data analytics. We may anticipate the deployment of IoT systems in an increasing number of consumer, commercial, industrial, and infrastructural applications in the near future. In terms of technology and device connectivity, the upcoming years will see the emergence of a whole new ecosystem.

LIST OF REFERENCES

1] From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence, Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle,1st Edition, Academic Press, 2014.

2] Learning Internet of Things, Peter Waher, PACKT publishing, BIRMINGHAM – MUMBAI,2015.

3] Building the Internet of Things with IPv6 and MIPv6: The Evolving World of M2M. Communications, Daniel Minoli, Wiley Publications,2013.

4] Internet of Things (A Hands-onApproach), Vijay Madisetti and ArshdeepBahga,1st Edition, VPT, 2014.

5] http://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot/index.html.

UNIT END EXERCISES

1] Define IoT and its functioning.

2] Write a note on building blocks of IoT architecture.

3] What are the main design principles and needed capabilities of an IoT system?

4] Describe an IoT architecture

5] What are the standards considerations of an IoT system?

❄❄❄❄❄❄❄

2

IOT ARCHITECTURE-STATE OF THE ART

Unit Structure :

2.0 OBJECTIVES

- To understand the state of art of an IoT system
- To get familiar with the IoT reference model architectural layers
- To acquaint with the different views associated with an IoT system

2.1 INTRODUCTION: STATE OF THE ART

The Internet of Things (IoT) can be viewed as a dynamic, worldwide networked infrastructure that controls autonomous items in a highly intelligent manner. As a result, new apps and services that can enhance human lives can be developed by connecting IoT devices that share information. At the beginning, Kevin Ashton, the founder of the MIT Auto Identification Center, originally suggested the idea of the IoT in 1999. The Internet of Things has the potential to revolutionize the world, much like the Internet did, according to Ashton. possibly even more so Eventually, the International Telecommunication Union (ITU) formally introduced the Internet of Things in 2005. There are numerous definitions of the IoT offered by numerous organizations and researchers.

However, the ITU's 2012 definition is the one that is most frequently used. According to what was said, there would be "a global infrastructure for the information society, enabling improved services by connecting (physical and virtual) things based on, existing and developing, interoperable information and communication technologies." Moreover, Guillemin and Friess in have offered one of the most straightforward formulations that accurately sum up the Internet of Things. According to the statement, "The Internet of Things enables people and things to be connected Whenever, Anywhere, with Anything and Anybody, ideally via Any Path/Network and Any Service." Many academics have offered many definitions of the Internet of Things (IoT) system from various angles, but the most crucial

point on which most experts have agreed is that the IoT was developed to build a better world for all people.

Figure 1. The IoT can connect anything in anywhere using any path

2.2 REFERENCE MODEL AND ARCHITECTURE

In October 2014, the IoT reference model was announced by the IoT World Forum (IWF) architecture committee. The industry may speed IoT deployments with the support of this model, which serves as a common framework. This reference model aims to promote and consolidate IoT deployment model development and collaboration. The IoT's seven-level architecture model is depicted in Figure 2. This reference model has seven layers, with each layer offering more details to help build a standard nomenclature. The initial step in enabling suppliers to develop IoT solutions that are compatible and interoperable is provided by this document, which also defines where particular types of processing are optimized across various layers of the system. In addition, this model made the IoT as a real and approachable system, instead of simply conceptual.

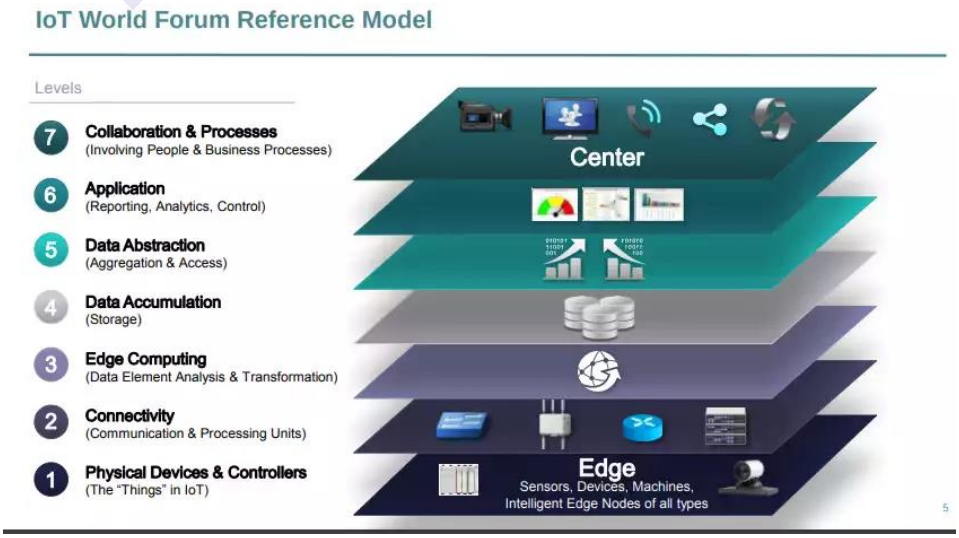


Figure 2: The IoT World Forum Reference Model

The physical layer of the Open System Interconnection (OSI) model of the network architecture is analogous to Layer 1. It is made up of mechanical components and object-controlling controllers. These things stand in for the Internet of Things and encompass a variety of information-sending and -receiving gadgets. For instance, sensors that gather various types of data about the surroundings.

Layer 2 deals with connectivity and communication. This layer consists of the hardware components used to build local and wide-area networks and enable Internet connectivity, such as routers, switches, gateways, and firewalls. Moreover, this layer facilitates communication between devices and with application platforms like PCs, remote controls, and cellphones.

The edge computing layer's function is to transform network data flows into data that can be stored and processed at a higher level. Processing components at this layer may handle large amounts of data and carry out data transformation activities, which leads to the storage of much smaller amounts of data.

The data buildup occurs at layer 4. This layer is responsible for storing data from various IoT devices. The edge computing layer, which takes in enormous amounts of data and stores it in storage so that higher levels can access it, filters and processes this data. The edge computing layer may be supplying data for storage in a variety of forms and from heterogeneous processors. Although the data abstraction layer gathers and organizes stored data so that programs can access it in a more manageable and effective manner.

Information interpretation takes place at layer 6, which is the application layer. Numerous applications that use IoT input data or manage IoT devices fall under this layer. The collaboration and processes layer identifies people who can interact and work together to improve the utility of the IoT system. Several applications are used on this layer to share data and manage information via the Internet. Table 1 lists the layers of the IoT architecture along with what each layer does.

Table 1. A summary of IoT architecture layers with its functions

IoT Layer	Function
Collaboration and processes layer	This layer allows different IoT applications to communicate and collaborate with each other to make the IoT data more useful
Application layer	This layer includes different IoT applications such healthcare, smart home, and others
Data abstraction layer	This layer aggregates and formats the IoT stored data to make it accessible to different IoT applications
Data accumulation layer	This layer is used to store data coming from different IoT devices
Edge computing layer	This layer converts network data into information suitable for storage and higher-level processing
Connectivity layer	This layer allows different IoT devices to communicate with one another using different internetworking devices
Physical layer	This layer consists of physical devices and controllers that control IoT objects/devices

2.3 FUNCTIONAL VIEW

In order to address the concerns of concrete IoT architecture and stakeholders, the reference architecture is presented as a set of architectural views mainly the functional view; information view; deployment and operational view.

The functional view provides the description of what the system does, and its main functions.

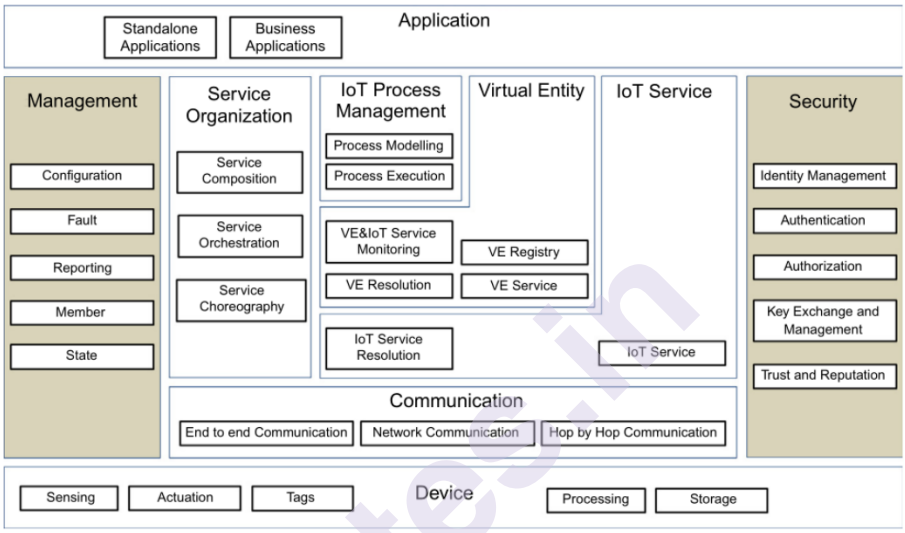


Figure 3: The IoT functional view

Device and Application functional group

- Device FG contains the Sensing, Actuation, Tag, Processing, Storage FCs, or simply components.
- These components represent the resources of the device attached to the Physical Entities of interest. The Application FG contains either standalone applications (e.g. for iOS, Android, Windows phone), or Business Applications that connect the IoT system to an Enterprise system

Communication functional group

- The Communication FG contains the End-to-End Communication, Network Communication, and Hop by-Hop communication components
- The Hop-by-Hop Communication is applicable in the case that devices are equipped with mesh radio networking technologies such as IEEE 802.15.4 for which messages have to traverse the mesh from node to-node (hop-by-hop) until they reach a gateway node which forwards the message (if needed) further to the Internet

Network FC

- The Network FC is responsible for message routing & forwarding and the necessary translations of various identifiers and addresses.
- The translations can be (a) between network layer identifiers to MAC and/or physical network identifiers, (b) between high-level human readable host/node identifiers to network layer addresses (e.g. Fully Qualified Domain Names (FQDN) to IP addresses, a function implemented by a Domain Name System (DNS) server), (c) translation between node/service identifiers and network locators in case the higher layers above the networking layer use node or service identifiers that are decoupled from the node addresses in the network (e.g. Host Identity Protocol)

End to End Communication

- The End-to-End Communication FC is responsible for end-to-end transport of application layer messages through diverse network and MAC/PHY layers.
- In turn, this means that it may be responsible for end to-end retransmissions of missing frames depending on the configuration of the FC.
- For example, if the End-to-End Communication FC is mapped in an actual system to a component implementing the Transmission Control Protocol (TCP) protocol, reliable transfer of frames dictates the retransmission of missing frames

IoT Service functional group- The IoT Service FC

- IoT Service functional group-The IoT Service FG consists of two FCs: The IoT Service FC and the IoT Service Resolution FC
- The IoT Service FC is a collection of service implementations, which interface the related and associated Resources.
- For a Sensor type of a Resource, the IoT Service FC includes Services that receive requests from a User and returns the Sensor Resource value in synchronous or asynchronous (e.g. subscription/notification) fashion.

IoT Service functional group

- The IoT Service Resolution FC-The IoT Service Resolution FC contains the necessary functions to realize a directory of IoT Services that allows dynamic management of IoT Service descriptions and discovery/lookup/resolution of IoT Services by other Active Digital Artifacts.
- Dynamic management includes methods such as creation/update/deletion (CUD) of Service description, and can be invoked by both the IoT Services themselves, or functions from the Management FG.

- The discovery/lookup and resolution functions allow other Services or Active Digital Artifacts to locate IoT Services by providing different types of information to the IoT Service Resolution FC.

Virtual Entity functional group

- The Virtual Entity FG contains functions that support the interactions between Users and Physical Things through Virtual Entity services.
- An example of such an interaction is the query to an IoT system of the form, "What is the temperature in the conference room Titan?"
- The Virtual Entity is the conference room "Titan," and the conference room attribute of interest is "temperature."
- The Virtual Entity Service FC enables the interaction between Users and Virtual Entities by means of reading and writing the Virtual Entity attributes (simple or complex), which can be read or written. The Virtual Entity Registry FC maintains the Virtual Entities of interest for the specific IoT system and their associations. The component offers services such as creating/reading/updating/deleting Virtual Entity descriptions and associations.

Virtual Entity resolution functional group

- The Virtual Entity Resolution FC maintains the associations between Virtual Entities and IoT Services, and offers services such as creating/reading/updating/deleting associations as well as lookup and discovery of associations.
- The Virtual Entity and IoT Service Monitoring FC includes: (a) functionality to assert static Virtual Entity - IoT Service associations, (b) functionality to discover new associations based on existing associations or Virtual Entity attributes such as location or proximity, and (c) continuous monitoring of the dynamic associations between Virtual Entities and IoT Services and updates of their status in case existing associations are not valid any more.

IoT process management functional group

- The IoT Process Management FG aims at supporting the integration of business processes with IoT-related services.
- It consists of two FCs: i] The Process Modeling FC provides that right tools for modeling a business process that utilizes IoT-related services.

ii] The Process Execution FC contains the execution environment of the process models created by the Process Modelling FC and executes the created processes by utilizing the Service Organization FG in order to resolve high-level application requirements to specific IoT services.

Service Organization functional group

- The Service Organization FG acts as a coordinator between different Services offered by the system. It consists of the following FCs: The Service Composition FC manages the descriptions and execution environment of complex services consisting of simpler dependent services. An example of a complex composed service is a service offering the average of the values coming from a number of simple Sensor Services.

- The Service Orchestration FC resolves the requests coming from IoT Process Execution FC or User into the concrete IoT services that fulfil the requirements.

- The Service Choreography FC is a broker for facilitating communication among Services using the Publish/Subscribe pattern.

Security functional group

- The Security FG contains the necessary functions for ensuring the security and privacy of an IoT system.

- It consists of the following FCs:

- The Identity Management FC manages the different identities of the involved Services or Users in an IoT system in order to achieve anonymity.

- The Authentication FC verifies the identity of a User and creates an assertion upon successful verification.

- It also verifies the validity of a given assertion.

- The Authorization FC manages and enforces access control policies. It provides services to manage policies (CUD), as well as taking decisions and enforcing them regarding access rights of restricted resources. The term "resource" here is used as a representation of any item in an IoT system that needs a restricted access.

- Such an item can be a database entry (Passive Digital Artifact), a Service interface, a Virtual Entity attribute (simple or complex), a Resource/Service/Virtual Entity description, etc.

- The Key Exchange & Management is used for setting up the necessary security keys between two communicating entities in an IoT system. This involves a secure key distribution function between communicating entities.

- The Trust & Reputation FC manages reputation scores of different interacting entities in an IoT system and calculates the service trust levels.

Management functional group

- The Management FG contains system-wide management functions that may use individual FC management interfaces. It is not responsible for the management of each component, rather for the management of the system as a whole. It consists of the following FCs:

- The Configuration FC maintains the configuration of the FCs and the Devices in an IoT system (a subset of the ones included in the Functional View).

- The component collects the current configuration of all the FCs and devices, stores it in a historical database, and compares current and historical configurations.

- The component can also set the system-wide configuration (e.g. upon initialization), which in turn translates to configuration changes to individual FCs and devices.

- The Fault FC detects, logs, isolates, and corrects system-wide faults if possible. This means that individual component fault reporting triggers fault diagnosis and fault recovery procedures in the Fault FC.

- The Member FC manages membership information about the relevant entities in an IoT system. Example relevant entities are the FGs, FCs, Services, Resources, Devices, Users, and Applications. Membership information is typically stored in a database along with other useful information such as capabilities, ownership, and access rules & rights, which are used by the Identity Management and Authorization FCs.

- The State FC is similar to the Configuration FC, and collects and logs state information from the current FCs, which can be used for fault diagnosis, performance analysis and prediction, as well as billing purposes. This component can also set the state of the other FCs based on system-wise state information.

- The Reporting FC is responsible for producing compressed reports about the system state based on input from FCs.

2.4 INFORMATION VIEW

Information view provides description of the data and information that the system handles.

- The information view consists of: (a) the description of the information handled in the IoT System, and (b) the way this information is handled in the system; in other words, the information lifecycle and flow (how information is created, processed, and deleted), and the information handling components.

- The pieces of information handled by an IoT system it can be

- Virtual Entity context information, i.e. the attributes (simple or complex) as represented by parts of the IoT Information model.
- IoT Service output itself is another important part of information generated by an IoT system. For example, this is the information generated by interrogating a Sensor or a Tag Service
- Virtual Entity descriptions in general, which contain not only the attributes coming from IoT Devices (e.g. ownership information).
- Associations between Virtual Entities and related IoT Services

2.5 DEPLOYMENT AND OPERATIONAL VIEW

- Deployment and Operational View provides description of the main real world components of the system such as devices, network routers, servers, etc.
- Devices that form networks in the M2M Area Network domain must be selected, or designed, with certain functionality in mind.
- At a minimum, they must have an energy source (e.g. batteries, increasingly EH), computational capability (e.g. an MCU), appropriate communications interface (e.g. a Radio Frequency Integrated Circuit (RFIC) and front end RF circuitry), memory (program and data), and sensing (and/or actuation) capability.
- These must be integrated in such a way that the functional requirements of the desired application can be satisfied

SUMMARY

It is difficult to predict the many applications of IoT
once it has reached the stage of ubiquitous expansion.

- "The Technical Foundations of IoT", Adryan, Obermaier, and Fremantle

The Internet of Things will succeed when it blends into the background and we stop noticing how new it is to have something connected to the Internet that interacts with other systems, services, and gadgets. While some goods are beginning to accomplish this, others still have a way to go. The development of serverless computing platforms as well as the concepts and design patterns of reactive systems will be crucial to the success of the IoT.

The Internet of Things (IoT) is thought to be the next step in the development of the Internet. To promote information sharing, it has the ability to connect and communicate with practically all real-world objects over the Internet. The Internet of Things (IoT) can gather, analyze, and deploy a significant quantity of data with the use of sensors. This data will then be transformed into knowledge and information that can be utilized to develop new applications and services that can enhance our quality of life. The IoT system has been reviewed in this essay. The IoT's cutting-edge

layered architecture is described. In addition, IoT essential features and different communication technologies are presented.

LIST OF REFERENCES

1] From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence, Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle,1st Edition, Academic Press, 2014.

2] Learning Internet of Things, Peter Waher, PACKT publishing, BIRMINGHAM – MUMBAI,2015.

3] Building the Internet of Things with IPv6 and MIPv6: The Evolving World of M2M. Communications, Daniel Minoli, Wiley Publications,2013.

4] Internet of Things (A Hands-onApproach), Vijay Madisetti and ArshdeepBahga,1st Edition, VPT, 2014.

5] http://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot/index.html.

UNIT END EXERCISES

1] Discuss the state of the art of an IoT architecture.

2] Explain the reference Model and architecture.

3] Discuss the functional view of an IoT system.

4] Explain the Information view of an IoT system.

5] Write a note on Deployment and Operational view of an IoT system.

❄❄❄❄❄❄❄

3

IOT DATA LINK LAYER AND NETWORK LAYER PROTOCOLS AND NETWORK LAYER

Unit Structure :

3.0 OBJECTIVES

After going through this unit, you will be able to:

- Design & develop IoT Devices

- Understand the importance of networking in IOT filed.
- be aware of the evolving world of M2M Communications
- IoT analytics
- interconnection and integration of the physical world
- get knowledge of the architecture of IoT.

3.1 INTRODUCTION

IoT is based on networking of things where smart devices communicate with each other by sending and receiving data. So, for that several network protocols (Communication protocols) are used to connect the IoT enabled devices and to establish the communication.

The data link layer is the protocol layer in a program that handles the moving of data into and out of a physical link in a network. The data link layer ensures an initial connection has been set up, divides output data into data frames and handles the acknowledgements from a receiver that the data arrived successfully. It also ensures incoming data has been received successfully by analyzing bit patterns at special places in the frames.

An important aspect of the Internet of Things is that devices are networked in some way, and often connected to the Internet. Networking enables devices to communicate with other IoT devices and larger cloud-based servers. IoT devices can often be thought of as small parts of a much larger collective system which includes large servers based in the cloud. This module will introduce the basics of networking and the Internet protocol in particular. Eventually, most IoT devices are connected to the Internet, so understanding the protocols associated with the Internet is important to the design of IoT devices

3.2.1 PHY/MAC Layer

Most network protocols use the concept of layers to separate different components and functions into independent modules that developers can assemble in different ways.

The characteristics required by applications, such as coverage area, scalability, transmission data rate, and applicability, refer to the Physical and Medium Access Control (MAC) layer designs of protocols.

The PHY layer defines the physical and electrical characteristics of the network. It is responsible for managing the hardware that modulates and demodulates the RF bits.

The perception layer is the physical layer, which has sensors for sensing and gathering information about the environment. It senses some physical parameters or identifies other smart objects in the environment.

Internet of Things (IoT) enables physical objects to sense, communicate, and perform certain actions on demand, which can facilitate a multitude of

applications, such as smart home, smart city, and intelligent transportation system

For achieving high throughput in an energy-efficient manner, it is crucial to design an efficient medium access control (MAC) protocol because the MAC layer is responsible for coordinating access among the IoT devices in the shared wireless medium

The MAC layer is responsible for sending and receiving RF frames. As part of each packet, there is a MAC layer data header that has addressing information as well as packet options. This layer implements packet acknowledgments (ACKs), packet tracking to eliminate duplicates, and so forth.

- When a device is transmitting, it cannot receive packets.
- When a device is not sleeping, it is either receiving or transmitting.
- There are no beacons or master/slave requirements in the design of the MAC/PHY.

3.2.2 3GPP MTC

3GPP (3rd generation partnership project) technologies and machine type communications (MTC) refer to small amounts of data that are communicated between machines (devices to back-end services and vice versa) without the need for any human intervention. In the 3rd Generation Partnership Project (3GPP), MTC is used to refer to all M2M communication (Jain et al. 2012). Thus, they are interchangeable terms.

Machine Type Communications — MTC (or Machine to Machine (M2M) communications) are about enabling direct communications among electronic devices, dubbed MTC devices, and/or enabling communications from MTC devices to a central MTC server or a set of MTC servers. Communications can use both wireless and fixed networks.

MTC will enable an endless number of applications in a wide plethora of domains, impacting different environments and markets. It will connect a potential number of MTC devices to the Internet and other networks, forming the Internet of Things. Several forecasts state a significant market growth over the next few years for both the MTC device and the MTC connectivity segments.

3.2.3 IEEE 802.11

- IEEE 802.11ah is a light (low-energy) version of the original IEEE 802.11 wireless medium access standard. It has been designed with less overhead to meet IoT requirements. IEEE 802.11 standards (also known as Wi-Fi) are the most commonly used wireless standards. They have been widely used and adopted for all digital devices including laptops, mobiles, tablets, and digital TVs. However, the original WiFi standards are not suitable for IoT applications due to their frame overhead and power consumption. Hence, IEEE 802.11 working group initiated 802.11ah task group to develop a standard that supports low overhead, power friendly communication suitable for sensors.

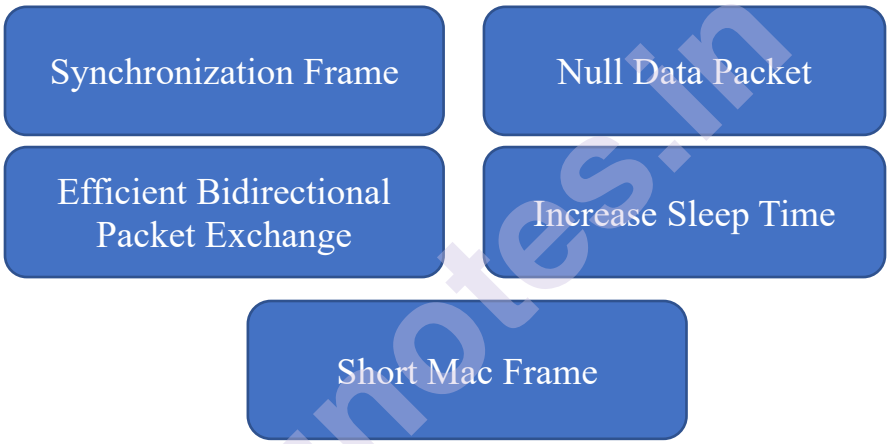
- EEE 802.11ah task group is working on a new amendment of the IEEE 802.11 standard, suitable for high density WLAN networks in the sub 1 GHz band. It is expected to be the prevalent standard in many Internet of Things (IoT) and Machine to Machine (M2M) applications where it will support long-range and energy-efficient communication in dense network environments.

- Therefore, significant changes in the legacy 802.11 standards have been proposed to improve the network performance in high contention scenarios, most important of which is the Restricted Access Window (RAW) mechanism described in the amendment. In this paper we analyze the performance of the RAW mechanism in the Non-Cross Slot Boundary case under various possible holding schemes.

- There are proposed new holding schemes as well as a new grouping scheme for RAW mechanism based on backoff states of the stations. The proposed schemes are there to improve the saturation throughput and energy efficiency of the network through extensive simulations. These schemes can therefore be adapted in practical deployment

scenarios of the IEEE 802.11ah use cases to improve the overall network performance. Overall, these advanced features make 802.11ah standard a true IoT-enabling technology towards seamless integration of massive number of connected devices in the future.

- **MAC layer features**



  - **Synchronization Frame**: A station is not allowed to transmit unless it has valid medium information that allows it to capture the medium and stop packet exchange by others. It can know such information if it receives the duration field packet correctly. If it does not receive it correctly, then it should wait for a duration called Probe Delay. Probe Delay can be configured by the access points in 802.11ah and announced by transmitting a synchronization frame at the beginning of the time slot.
  - **Efficient Bidirectional Packet Exchange**: This feature allows the sensor device to save more power by allowing both uplink and downlink communication between the access point and the sensor and allowing it to go to sleep as soon as it finishes the communication.
  - **Short Mac Frame**: The normal IEEE 802.11 frame is about 30 bytes, which is too large for IoT applications. IEEE 802.11ah mitigates this problem by defining a short MAC frame with about 12 bytes.

- **Null Data Packet**: In IEEE 802.11 the control frames, such as Acknowledgment (ACK) frames, are about 14 bytes and have no data, which adds a lot of overhead. IEEE 802.11ah mitigates this problem by replacing the ACK frame with a preamble, a tiny signal.
- **Increase Sleep Time**: 802.11ah is designed for low-power sensors and, hence, it allows a long sleep period of time and waking up infrequently to exchange data only.

3.2.4 IEEE 802.15

- IEEE 802.15.4 is the most commonly used IoT standard for MAC. It defines a frame format, headers including source and destination addresses, and how nodes can communicate with each other. The frame formats used in traditional networks are not suitable for low power multi-hop networking in IoT due to their overhead.
- **IEEE802.15.4e was created to extend IEEE802.15.4 and support low power communication.**
- **MAC features**
- **Slot frame Structure**: frame structure is designed for scheduling and telling each node what to do. A node can sleep, send, or receive information. In the sleep mode, the node turns off its radio to save power and stores all messages that it needs to send at the next transmission opportunity. When transmitting, it sends its data and waits for an acknowledgment. When receiving, the node turns on its radio before the scheduled receiving time, receives the data, sends an acknowledgement, turn off its radio, delivers the data to the upper layers and goes back to sleep.
- **Scheduling:** The standard does not define how the scheduling is done but it needs to be built carefully such that it handles mobility scenarios. It can be centralized by a manager node which is responsible for building the schedule, informing others about the schedule and other nodes will just follow the schedule.
- **Synchronization:** Two approaches can be used: acknowledgment-based or frame- based synchronization. In acknowledgement-based mode, the nodes are already in communication and they send acknowledgment for reliability guarantees, thus can be used to maintain connectivity as well. In frame-based mode, nodes are not communicating and hence, they send an empty frame at pre-specified intervals (about 30 second typically).
- **Channel Hopping**: IEEE802.15.4e introduces channel hopping for time slotted access to the wireless medium. Channel hopping requires changing the frequency channel using a pre- determined random sequence.

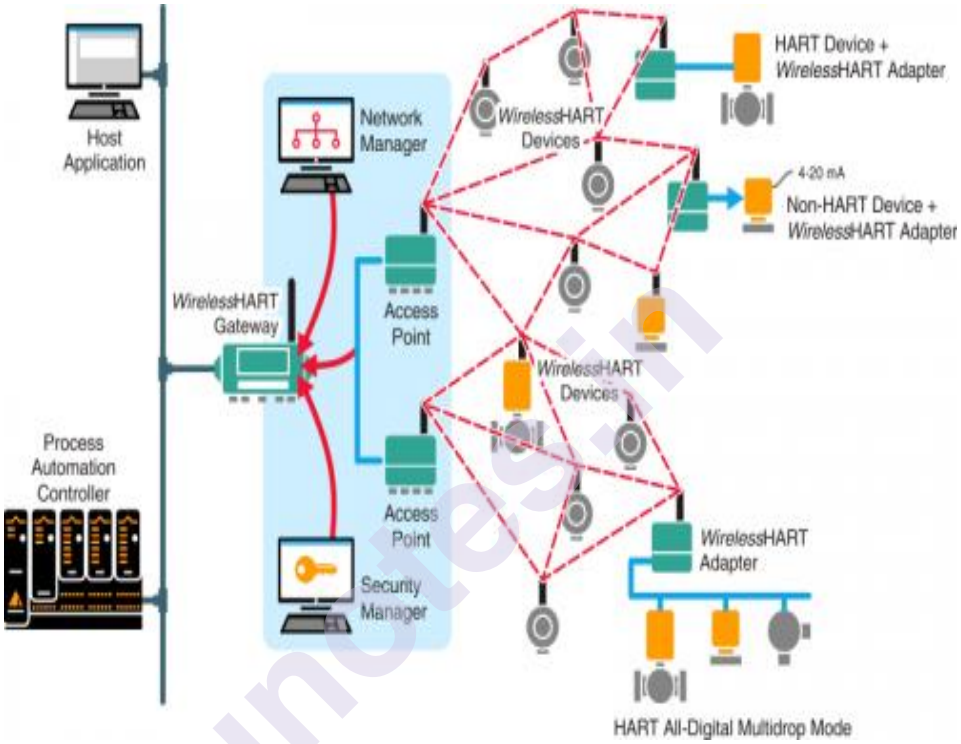
- **Network formation**: Network formation includes advertisement and joining components. A new device should listen for advertisement commands and upon receiving at least one such command, it can send a join request to the advertising device. In a centralized system, the join request is routed to the manger node and processed there while in distributed systems, they are processed locally. Once a device joins the network and it is fully functional, the formation is disabled and will be activated again if it receives another join request.

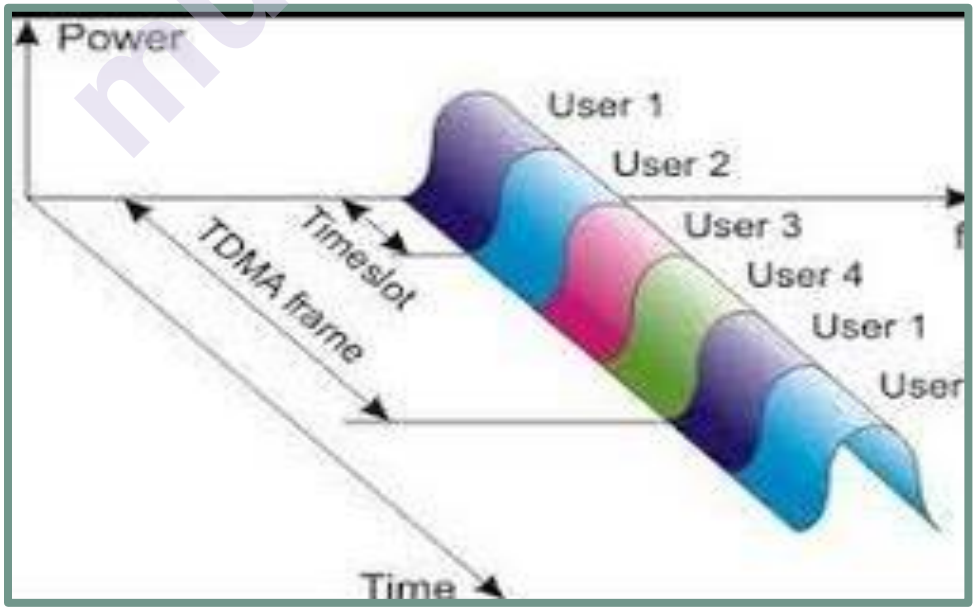
3.2.5 Wireless HART

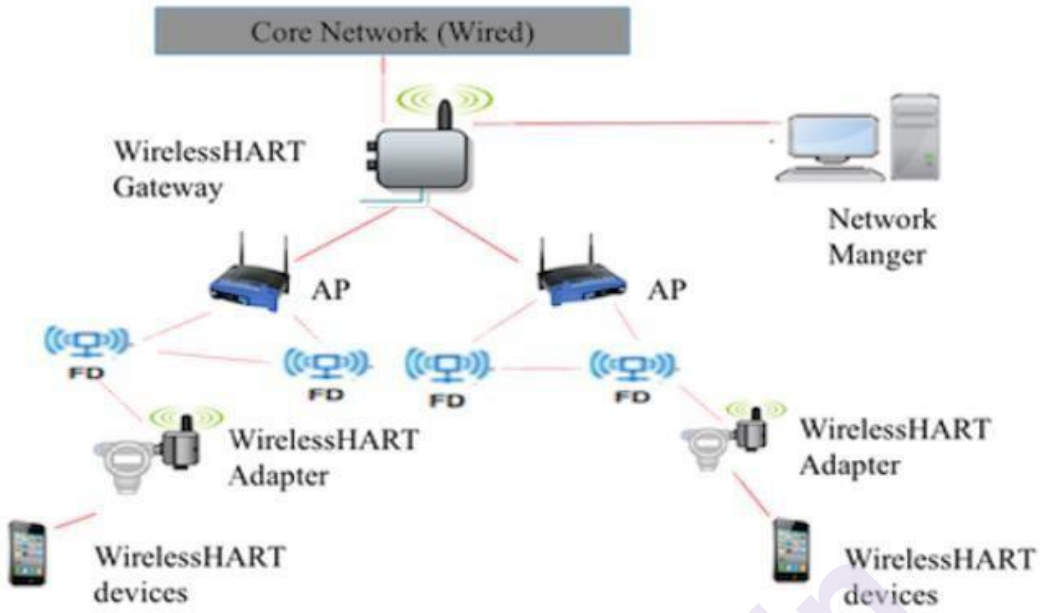
- Wireless HART is a datalink protocol that operates on the top of IEEE 802.15.4 PHY and adopts Time Division Multiple Access (TDMA) in its MAC. It is a secure and reliable MAC protocol that uses advanced encryption to encrypt the messages and calculate the integrity in order to offer reliability.
- Wireless HART is a wireless sensor networking technology based on the Highway Addressable Remote Transducer Protocol (HART). Wireless HART was defined for the requirements of process field device networks.
- Developed as a multi-vendor, interoperable wireless standard, Wireless HART was defined for the requirements of process field device networks. The standard was initiated in early 2004 and developed by 37 HART Communications Foundation (HCF) companies that - amongst others - included ABB, Emerson, Endress+Hauser, Pepperl+Fuchs, Siemens which form WiTECK an open, non-profit membership organization whose mission is to provide a reliable, cost-effective, high-quality portfolio of core enabling system software for industrial wireless sensing applications, under a company- and platform-neutral umbrella.More than 30 million devices worldwide use the HART communication protocol.
- Wireless HART is a radio communications protocol that adds wireless capabilities to the HART protocol while maintaining compatibility with existing HART devices, commands, and tools.
- Wireless HART utilizes a time synchronized, self-organizing, and self-healing mesh architecture.
- Wireless HART supports operation in the 2.4 GHz ISM (license free) band using IEEE 802.15.4 standard radios.
- In a Wireless HART network, every participating device simultaneously works as a signal source and a data repeater.
- Through the routing of the individual signals across the entire network, a wide network structure becomes possible.
- Data repeaters (routers) can be strategically placed to expand the network reach throughout the facility or field application.

There are three main elements in any Wireless HART network:

- Wireless field devices connect to plant equipment.
- Gateways enable communication between these devices and hosted applications.
- A network manager, typically integrated into the Gateway, is responsible for configuring the network and managing communication and message routing.







- consists of a network manager, a security manager, a gateway to connect the wireless network to the wired networks, wireless devices as field devices, access points, routers and adapters. The standard offers end-to-end, per-hop or peer-to- peer security mechanisms. End to end security mechanisms enforce security from sources to destinations while per-hop mechanisms secure it to next hop only.

Advantages of Wireless HART

- widely used,
- cost-effective,
- scalable,
- platform-neutral,
- secure,
- backwards compatible, and
- it provides a robust wireless protocol for the full range of process measurement, control, and asset management applications.

3.2.6 Z-Wave

- Z-Wave is a low-power MAC protocol designed for home automation and has been used for IoT communication, especially for smart home and small commercial domains. It covers about 30- meter point-to-point communication and is suitable for small messages in IoT applications, like light control, energy control, wearable healthcare control and others. It uses CSMA/CA for collision detection and ACK messages for reliable transmission. It follows a master/slave architecture in which the master controls the slaves, send them commands, and handling scheduling of the whole network.

- Z-Wave technology is the new wireless communication protocol for home automation devices. Used in a variety of smart home applications like lighting, security, entertainment and others, it's one of the upcoming new communication standards in the world of Internet-of-Things (IoT). It's currently being used by more than 300 plus manufacturers and thousands of devices, making it one of the most popular communication protocols after ZigBee.

- Unlike the **ZigBee** protocol that works at 2.4GHz wireless frequency, Z-Wave operates at in 900 MHz frequency bands. Due to band constraints in different geographies, Z-Wave is a region-specific protocol, which means it has different legally permissible frequencies in different regions.

- However, since it operates in the low-frequency region, Z-Wave offers superior performance. The longer wavelength and lower frequency of the Z-Wave allow it to easily penetrate objects and walls, thereby establishing a more reliable and faster communication topology between the connected Z-Wave devices.

- The Z-Wave uses a source-routed mesh network which consist of the one primary node – often referred to as the controller or hub in Home Automation, and many other secondary nodes referred to as sensors. In all the Z-Wave networks, the primary controllers are identified by network IDs and the secondary nodes or Z-Wave devices are identified by Node IDs.

- The primary controller assigns Network IDs to all the devices configured within its established network. Thus, all the devices within a Z-Wave network have the same Network ID because of which they are interoperable. Devices with different Network IDs cannot communicate with each other.

Z-Wave Features

- Z-Wave uses much lower transmission power as compared to Wi-Fi ensuring 3-5 years of battery life to its users.

- Z-Wave operates within the 900 MHz frequency band causing less interference and higher penetration.

- Allows interoperability between version through 6 layers of backward compatibility.

- Z-Wave devices can communicate within the range of 120 feet to 40 metres.

- Z-Wave offers data rates of 100 kbps and higher security through AES128 encryption.

- Works across range of household products like lightings, thermostats, security sensors, locks etc.

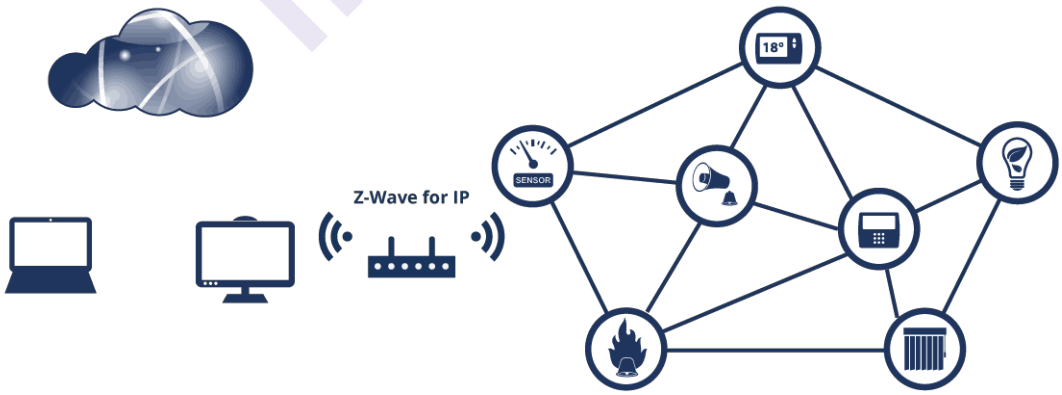
- Working on a different frequency, it will not interfere with Wi-Fi or any other household signals within the home.
- A given Z-Wave network can control up to 232 devices configured within that network.

Applications of Z-Wave Technology

- The widely used application of Z-Wave technology as on date is for Home Automation systems.
- Z-Wave can also be used for efficient energy management systems, security power theft etc.
- The facility of interoperability among devices working on Z-Wave help you to club applications like lighting automation, smart security automation, entertainment automation etc.
- Also used in industrial automation for smooth functioning of interdependent processes.
- Widely used in smart home security systems.

Examples of Z wave devices:

- Garage Door
- Lighting Controls
- On/Off Outlets
- Remote Controls
- Smart Home Security Systems
- Smart Locks
- Smoke Detectors
- Thermostats



3.2.7 Bluetooth Low Energy

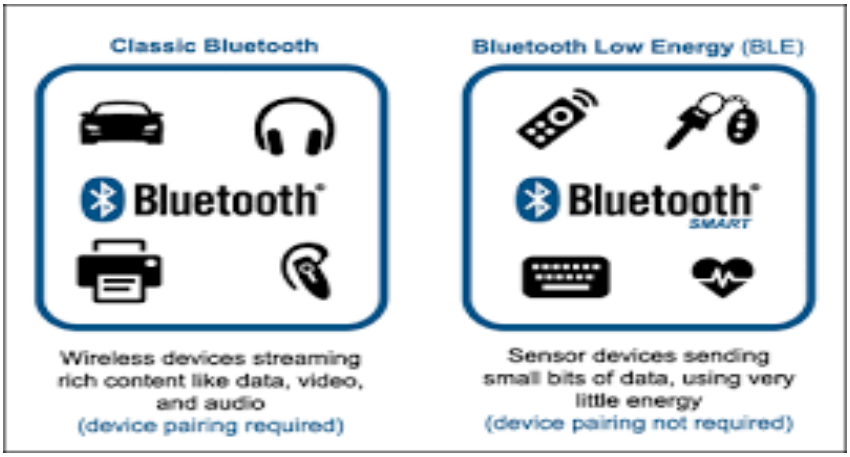
- Bluetooth low energy or Bluetooth smart is a short-range communication protocol with PHY and MAC layer widely used for in-vehicle networking. Its low energy can reach ten times less than the classic Bluetooth while its latency can reach 15 times. Its access control uses a contention- less MAC with low latency and fast transmission. It follows master/slave architecture and offers two types of frames: adverting and data frames. The Advertising frame is used for discovery and is sent by slaves on one or more of dedicated advertisement channels. Master nodes sense advertisement channels to find slaves and connect them. After connection, the master tells the slave its waking cycle and scheduling sequence. Nodes are usually awake only when they are communicating and they go to sleep otherwise to save their power.

- Today, most smartphones and tablets are BLE compatible, which means they can seamlessly communicate with Bluetooth-enabled wireless headphones, digital signage, car stereos, fitness trackers, smartwatches, and hardware devices like beacons.

- **How does BLE technology work?**

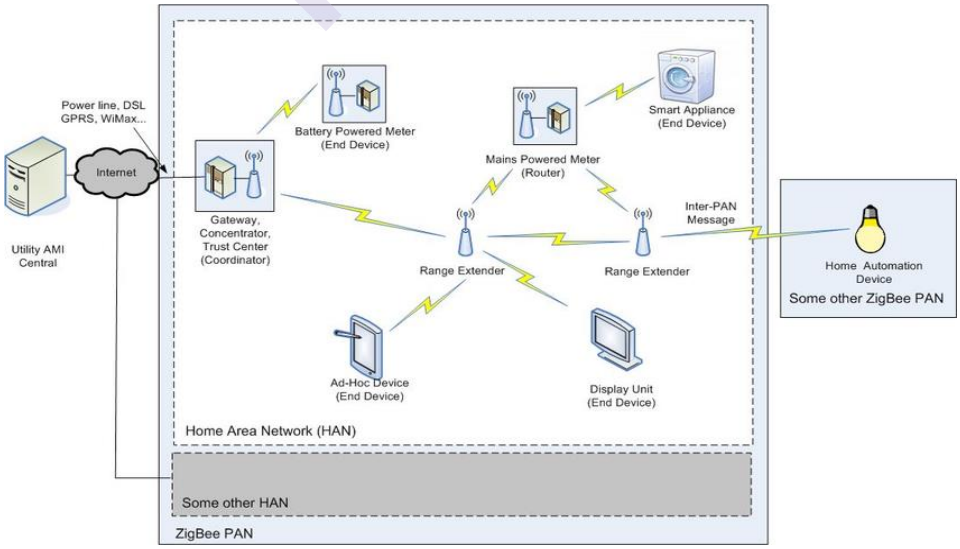
  BLE data transfer is essentially one-way communication. Let's take an example of BLE beacons trying to communicate with a smartphone nearby – a Bluetooth beacon device broadcasts packets of data at regular intervals. These data packets are detected by app/pre-installed services on smartphones nearby. This BLE communication triggers actions such as pushing a message or promoting an app.

  To save energy and provide higher data transfer speed, the entire Bluetooth BLE communication framework consists of 40 frequency channels, separated by 2MHz. 3 of these channels are the primary advertisement channels, while the remaining 37 channels are secondary, known as data channels. The Bluetooth communication starts with the 3 primary advertisement channels and then offloads to the secondary channels.



3.2.8 Zigbee Smart Energy

- ZigBee smart energy is designed for a large range of IoT applications including smart homes, remote controls and healthcare systems. It supports a wide range of network topologies including star, peer-to-peer, or cluster-tree. A coordinator controls the network and is the central node in a star topology, the root in a tree or cluster topology and may be located anywhere in peer-to-peer. ZigBee standard defines two stack profiles: ZigBee and ZigBee Pro. These stack profiles support full mesh networking and work with different applications allowing implementations with low memory and processing power. ZigBee Pro offers more features including security using symmetric-key exchange, scalability using stochastic address assignment, and better performance using efficient many-to-one routing mechanisms.
- ZigBee Smart Energy (SE) is the world's leading **standard for interoperable wireless products** that monitor, control and automate the delivery and use of energy (and other resources, such as water).



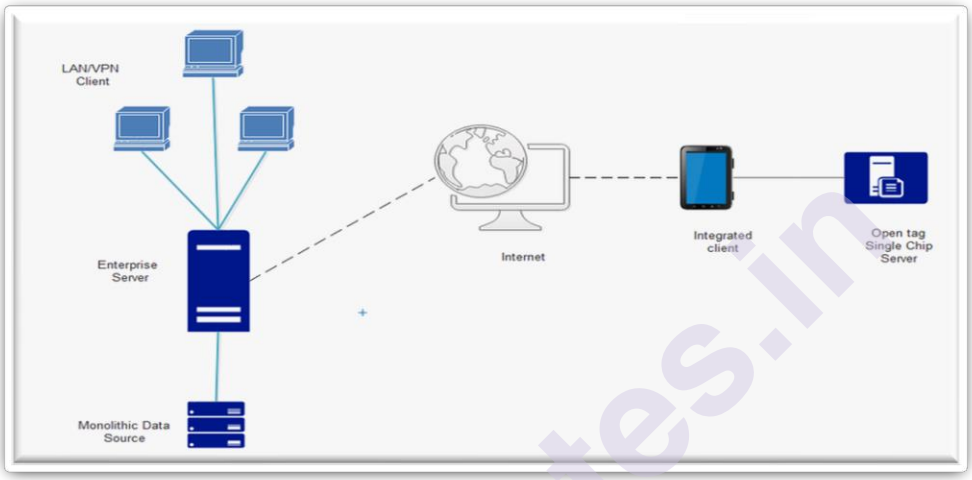
Zigbee is a wireless technology that supports automation—it allows many homes security and smart home devices to interconnect in a single system. It sounds complicated, but it works a lot like your home Wi-Fi network, just for smart devices. Just like you need a wireless router to create a Wi-Fi network, you also need a hub to set up and control a Zigbee network.

The hub coordinates the system, and each device also acts as a node to create a mesh network of devices.

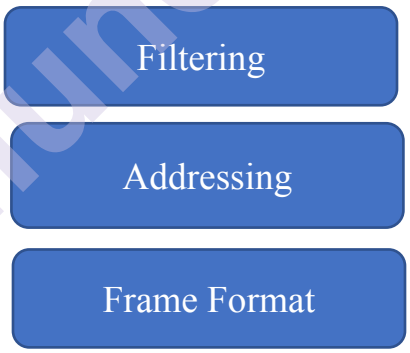
3.2.9 DASH7

- DASH7 is an "instant-on," long-range, low power wireless communications standard for applications requiring modest bandwidth like text messages, sensor readings, or location-based advertising coordinates.
- It is an open-source RFID-standard for wireless sensor networking, which operates in the 433 MHz unlicensed ISM band/SRD band. DASH7 provides multi-year battery life, range of up to 2 km, indoor location with 1 meter accuracy, low latency for connecting with moving things, a very small open-source protocol stack, AES 128-bit shared key encryption support, and data transfer of up to 200 kbit/s.
- DASH7 Alliance Protocol originates from the ISO/IEC 18000-7 standard describing a 433 MHz ISM band air interface for active RFID. This standard was mainly used for military logistics.
- The DASH7 Alliance re-purposed the original 18000-7 technology in 2011 and made it evolve toward a wireless sensor network technology for commercial applications. The DASH7 Alliance Protocol covers all sub-GHz ISM bands, making it available globally. The name of the new protocol was derived from the section seven denoted as -7 of the original standard documents.
- The current version of the DASH7 Alliance protocol is no longer compliant with the ISO/IEC 18000-7 standard.

- DASH7 networks serves applications in which low power usage is essential and data transmission is typically much slower and/or sporadic, like basic telemetry.
- DASH7 is a wireless communication protocol for active RFID that operates in globally available Industrial Scientific Medical (ISM) band and is suitable for IoT requirements. It is mainly designed for scalable, long range outdoor coverage with higher data rate compared to traditional ZigBee. It is a low-cost solution that supports encryption and IPv6 addressing. It supports a master/slave architecture and is designed for burst, lightweight, asynchronous and transitive traffic.



- **Its MAC layer features**



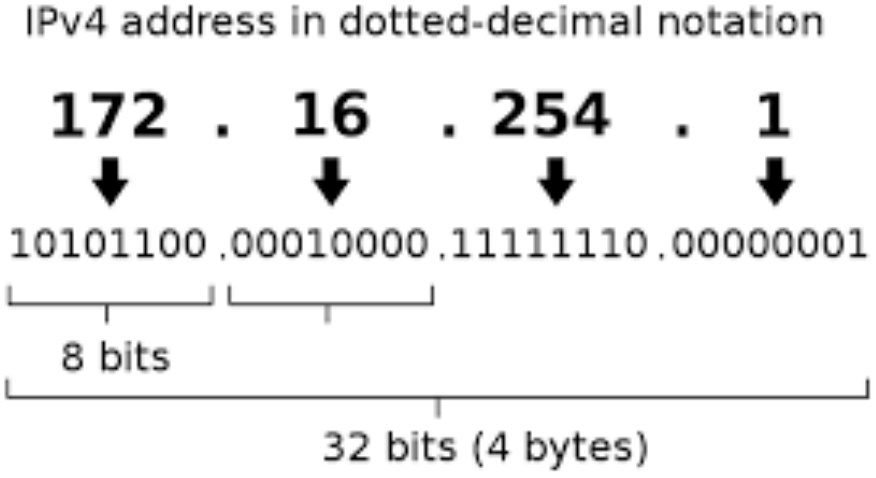
- **Filtering**: Incoming frames are filtered using three processes; cyclic redundancy check (CRC) validation, a 4-bit subnet mask, and link quality assessment. Only the frames that pass all three checks are processed further.
- **Addressing**: DASH7 uses two types of addresses: the unique identifier which is the EUI-64 ID and dynamic network identifier which is 16-bit address specified by the network administrator.
- **Frame format**: The MAC frame has a variable length of maximum 255 bytes including addressing, subnets, estimated power of the transmission and some other optional fields.

3.3 NETWORK LAYER

3.3.1 IPv4

- Internet Protocol Version 4 (IPv4) is the fourth revision of the Internet Protocol and a widely used protocol in data communication over different kinds of networks. IPv4 is a connectionless protocol used in packet-switched layer networks, such as Ethernet. It provides the logical connection between network devices by providing identification for each device. There are many ways to configure IPv4 with all kinds of devices – including manual and automatic configurations – depending on the network type.
- It is one of the core protocols of standards-based internetworking methods in the Internet, and was the first version deployed for production in the ARPANET in 1983. IPv4 is a connectionless protocol for use on packet-switched networks.
- It operates on a best effort delivery model, in that it does not guarantee delivery, nor does it assure proper sequencing or avoidance of duplicate delivery. These aspects, including data integrity, are addressed by an upper layer transport protocol, such as the Transmission Control Protocol (TCP).
- IPv4 is based on the best-effort model. This model guarantees neither delivery nor avoidance of duplicate delivery; these aspects are handled by the upper layer transport.
- IPv4 is defined and specified in IETF publication RFC 791. It is used in the packet- switched link layer in the OSI model.
- IPv4 uses 32-bit addresses for Ethernet communication in five classes: A, B, C, D and E. Classes A, B and C have a different bit length for addressing the network host. Class D addresses are reserved for multicasting, while class E addresses are reserved for future use.

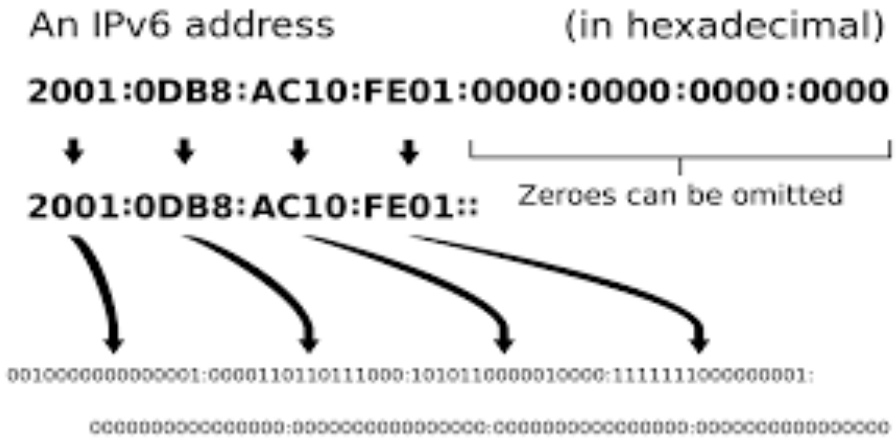
Eg.



3.3.2 IPv6

- An Ipv6 address uses 128 bits as opposed to 32 bits in IPv4.
- IPv6 provides for end devices to have multiple addresses and an even more distributed routing mechanism than the IPv4 Internet. This allows different stakeholders to assign IoT end-device addresses that are consistent with their own application and network practices.
- IPv6 provides strong features and solutions to support mobility of end-nodes, as well as mobility of the routing nodes of the network.
- IPv6 provides an address self-configuration mechanism (Stateless mechanism). The nodes can define their addresses in very autonomous manner.
- IPv6 Gateways can be fully Internet compliant. In other words, it is possible to build a proprietary network of smart things or to interconnect one's own smart things with the rest of the World via a gateway that is fully compliant with IP requirements towards the Internet.
- Thus, multiple stakeholders can deploy their own applications, sharing a common sensor/actuation infrastructure, without impacting the technical operation or governance of the Internet.
- IPv6 addresses are written using hexadecimal, as opposed to dotted decimal in IPv4.
- Because a hexadecimal number uses 4 bits this means that an IPv6 address consists of 32 hexadecimal numbers.
- These numbers are grouped in 4's giving 8 groups or blocks. The groups are written with a: (colon) as a separator.
- group1: group2: group3: group4: group5: group6: group7: group8.

Eg



3.3.3 6LoWPAN

- IPv6 over Low power Wireless Personal Area Network (6LoWPAN) is the first and most commonly used standard in this category. It efficiently encapsulates IPv6 long headers in IEEE802.15.4 small packets, which cannot exceed 128 bytes. The specification supports different length addresses, low bandwidth, different topologies including star or mesh, power consumption, low cost, scalable networks, mobility, unreliability and long sleep time. The standard provides header compression to reduce transmission overhead, fragmentation to meet the 128-byte maximum frame length in IEEE802.15.4, and support of multi-hop delivery.

- A key IP (Internet Protocol)-based technology is 6LowPAN (IPv6 Low-power wireless Personal Area Network). Rather than being an IoT application protocols technology like Bluetooth or ZigBee, 6LowPAN is a network protocol that defines encapsulation and header compression mechanisms. The standard has the freedom of frequency band and physical layer and can also be used across multiple communications platforms, including Ethernet, Wi-Fi, 802.15.4 and sub-1GHz ISM.

- A key attribute is the IPv6 (Internet Protocol version 6) stack, which has been a very important introduction in recent years to enable the IoT. IPv6 is the successor to IPv4 and offers approximately 5 x 10^{28} addresses for every person in the world, enabling any embedded object or device in the world to have its own unique IP address and connect to the Internet.

- Especially designed for home or building automation, for example, IPv6 provides a basic transport mechanism to produce complex control systems and to communicate with devices in a cost-effective manner via a low-power wireless network.

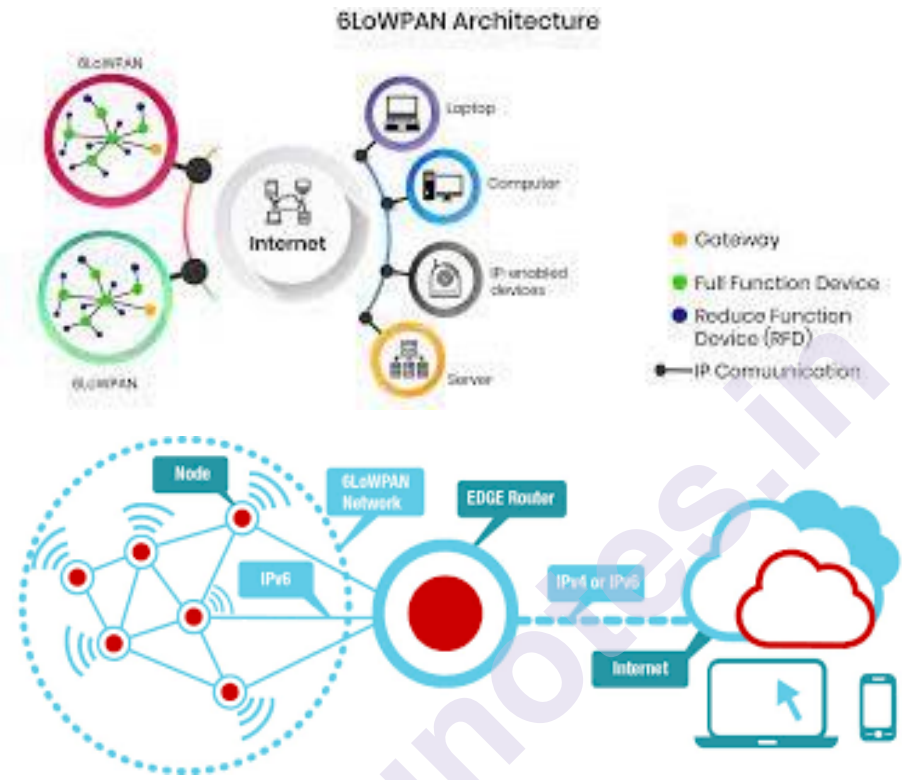
- Designed to send IPv6 packets over IEEE802.15.4-based networks and implementing open IP standards including TCP, UDP, HTTP, COAP, MQTT, and WebSocket, the standard offers end-to-end addressable nodes, allowing a router to connect the network to IP. 6LowPAN is a mesh network that is robust, scalable and self-healing. Mesh router devices can route data destined for other devices, while hosts are able to sleep for long periods of time.

Frames in 6LoWPAN use four types of headers:

- No 6loWPAN header (00),
- Dispatch header (01),
- Mesh header (10) and
- Fragmentation header (11).

- In No 6loWPAN header case, any frame that does not follow 6loWPAN specifications is discarded.

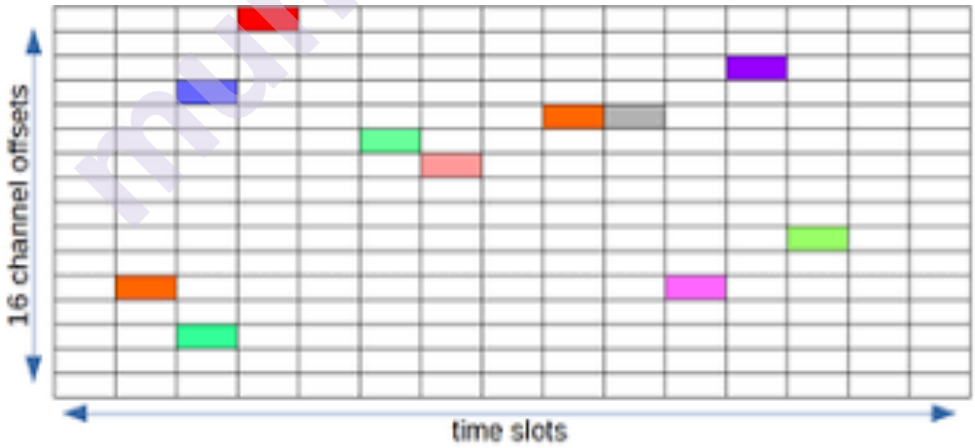
- Dispatch header is used for multicasting and IPv6 header compressions.
- Mesh headers are used for broadcasting; while Fragmentation headers are used to break long IPv6 header to fit into fragments of maximum 128-byte length.

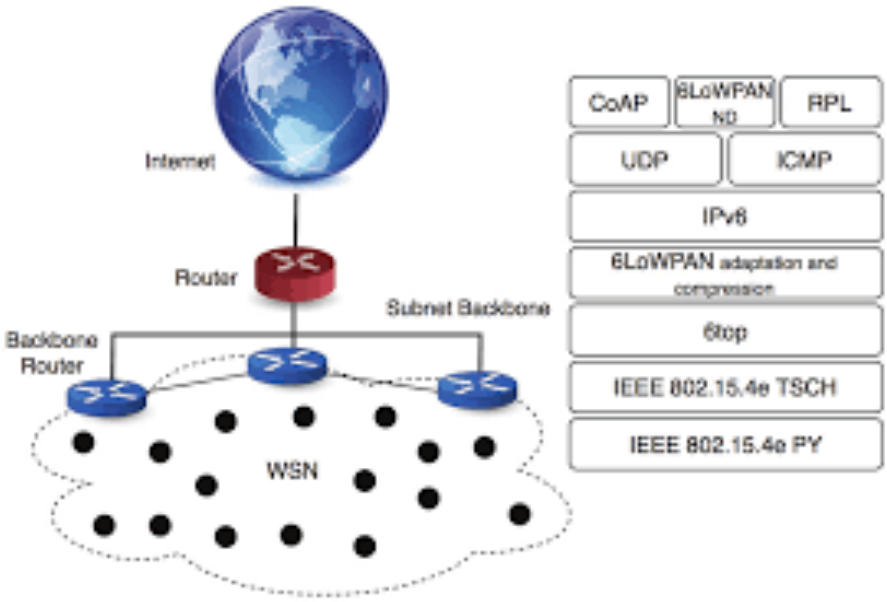


3.3.4 6TiSCH

- 6TiSCH has been developed by the Internet Engineering Task Force (IETF), the standards body behind most of the technical solutions used in the today's Internet.
- The IETF adopts an open standardization approach: participation to the standardization activities is open to all, contributions are judged on their technical merit only, and the resulting standards are available at no charge.
- Industrial networks using 6TiSCH seamlessly integrate into the Internet architecture, without the need to bridge or handle protocol translation at the application layer. Therefore, 6TiSCH fully enables the vision of a "cloudified" industry, where sensor and actuator devices connect to cloud-based SCADA (Supervisory control and data acquisition) systems
- 6TiSCH working group in IETF is developing standards to allow IPv6 to pass through Time-Slotted Channel Hopping (TSCH) mode of IEEE 802.15.4e datalinks.

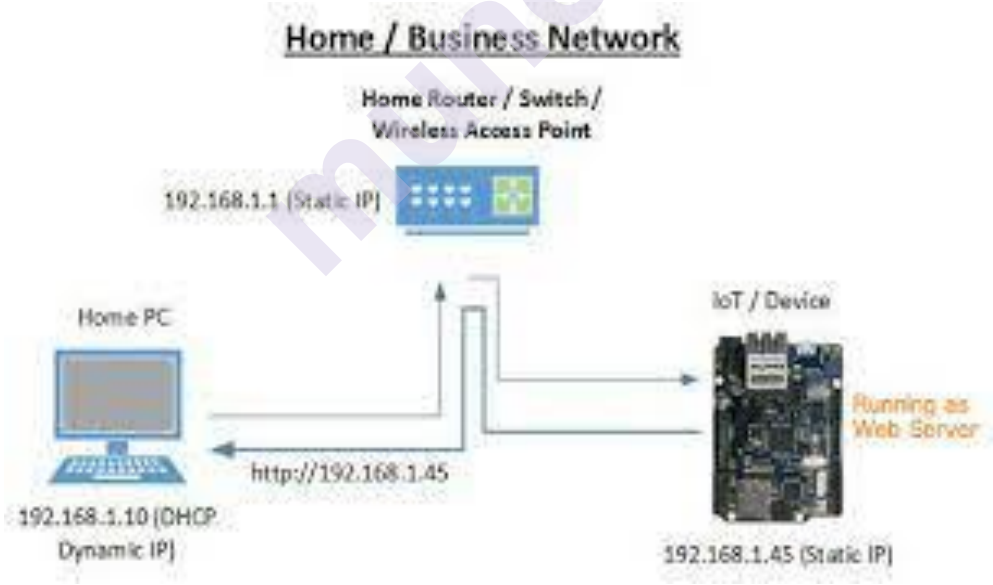
- 6TiSCH builds on IEEE 802.15.4 compliant hardware and its TSCH MAC layer. The IEEE 802.15.4 standard expects an “upper layer” to perform several critical management tasks that are needed for the network operation.
- The 6TiSCH specifies these solutions: inter-operable and zero-configuration network bootstrap, network access authentication and parameter distribution and the management of the wireless medium through scheduling.
- The component in the 6TiSCH architecture that is in charge of dynamically adapting the schedule is called a Scheduling Function (SF).
- It defines a Channel Distribution usage matrix consisting of available frequencies in columns and time-slots available for network scheduling operations in rows. This matrix is portioned into chucks where each chunk contains time and frequencies and is globally known to all nodes in the network.
- The nodes within the same interference domain negotiate their scheduling so that each node gets to transmit in a chunk within its interference domain. Scheduling becomes an optimization problem where time slots are assigned to a group of neighboring nodes sharing the same application. The standard does not specify how the scheduling can be done and leaves that to be an application specific problem in order to allow for maximum flexibility for different IoT applications. The scheduling can be centralized or distributed depending on application or the topology used in the MAC layer.





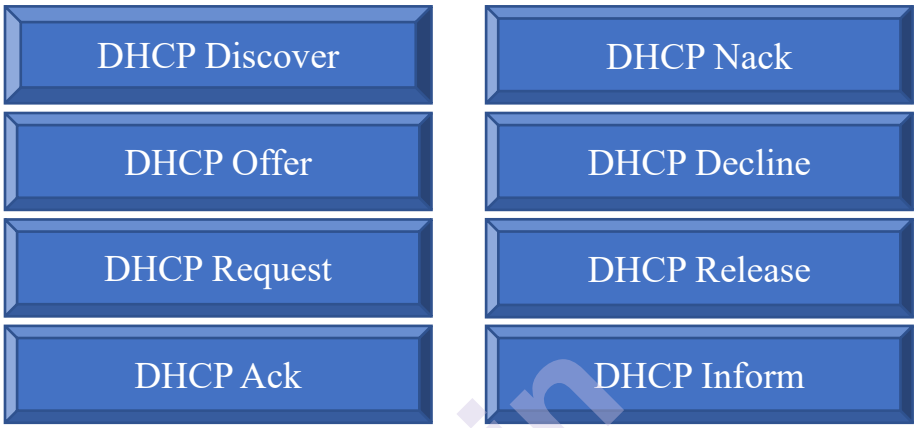
3.3.5 DHCP

- **DHCP Architecture**
  - The DHCP architecture consists of DHCP clients, DHCP servers, and DHCP relay agents on a network. The clients interact with servers using DHCP messages in a DHCP conversation to obtain and renew IP address leases.



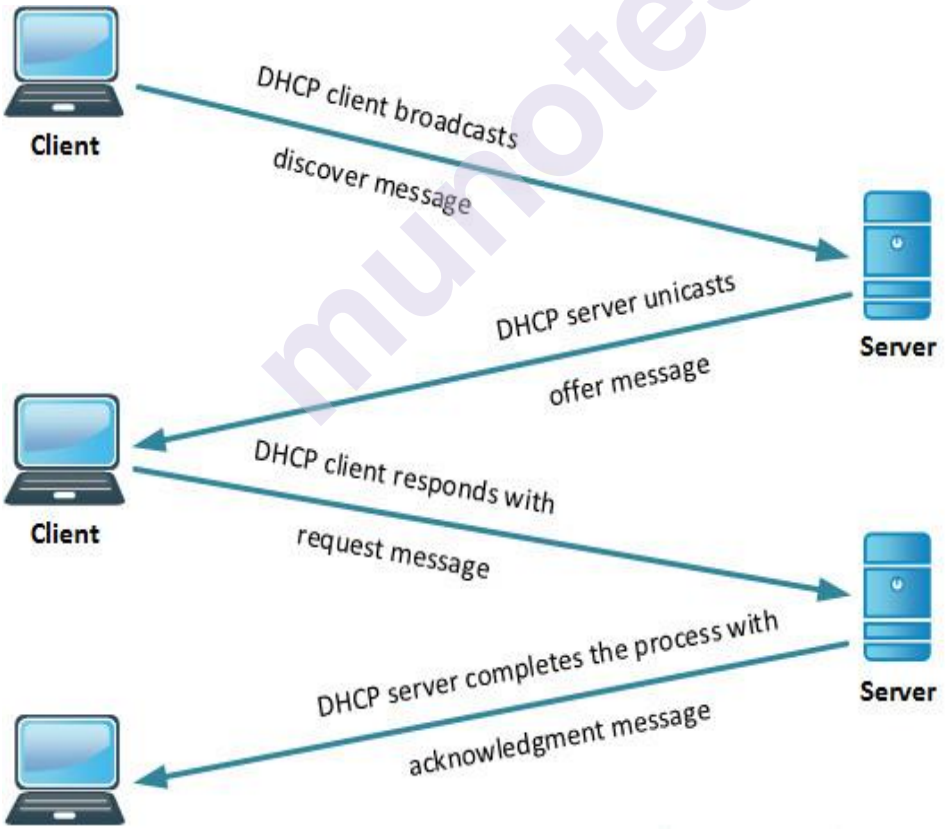
- **DHCP Client Functionality**
  - A DHCP client is any network-enabled device that supports the ability to communicate with a DHCP server in compliance with RFC 2131, for the purpose of obtaining dynamic leased IP configuration and related optional information.

- **Lease Durations**
  - When a scope is created, the lease duration is set to eight days by default. However, there are situations when the administrator might want to change the lease duration.
- **DHCP Messages**



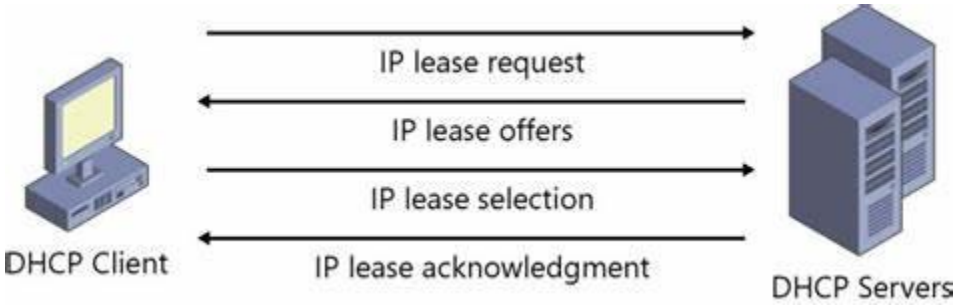
- **DHCP Discover**
  - Broadcast by a DHCP client when it first attempts to connect to the network. The DHCP Discover message requests IP address information from a DHCP server.
- **DHCP Offer**
  - Broadcast by each DHCP server that receives the client DHCP Discover message and has an IP address configuration to offer to the client. The DHCP Offer message contains an unleased IP address and additional TCP/IP configuration information, such as the subnet mask and default gateway. More than one DHCP server can respond with a DHCP Offer message. The client accepts the best offer, which for a Windows DHCP client is the first DHCP Offer message that it receives.
- **DHCP Request**
  - Broadcast by a DHCP client after it selects a DHCP Offer. The DHCP Request message contains the IP address from the DHCP Offer that it selected. If the client is renewing or rebinding to a previous lease, this packet might be unicast directly to the server.
- **DHCP Ack**
  - Broadcast by a DHCP server to a DHCP client acknowledging the DHCP Request message. At this time, the server also forwards any options. Upon receipt of the DHCP Ack, the client can use the leased IP address to participate in the TCP/IP network and complete its system startup. This message is typically broadcast, because the DHCP client does not officially have an IP address that it can use at this point. If the DHCP Ack is in response to a DHCP Inform, then the message is unicast directly to the host that sent the DHCP Inform message.

- **DHCP Nack**
    - Broadcast by a DHCP server to a DHCP client denying the client's DHCP Request message. This might occur if the requested address is incorrect because the client moved to a new subnet or because the DHCP client's lease has expired and cannot be renewed.
- **DHCP Decline**
    - Broadcast by a DHCP client to a DHCP server, informing the server that the offered IP address is declined because it appears to be in use by another computer.
- **DHCP Release**
    - Sent by a DHCP client to a DHCP server, relinquishing an IP address and canceling the remaining lease. This is unicast to the server that provided the lease.
- **DHCP Inform**
    - Sent from a DHCP client to a DHCP server, asking only for additional local configuration parameters; the client already has a configured IP address.



- **DHCP Lease Process**
    - A DHCP-enabled client obtains a lease for an IP address from a DHCP server. Before the lease expires, the DHCP client must

renew the lease or obtain a new lease. Leases are retained in the DHCP server database for a period of time after expiration. By default, this grace period is four hours and cleanup occur once an hour for a DHCP server running Windows Server 2003. This protects a client's lease in case the client and server are in different time zones, the internal clocks of the client and server computers are not synchronized, or the client is off the network when the lease expires.



- **Obtaining a New Lease**
    - DHCP client initiates a conversation with a DHCP server when it is seeking a new lease, renewing a lease, rebinding, or restarting. The DHCP conversation consists of a series of DHCP messages passed between the DHCP client and DHCP servers. The following figure shows an overview of this process when the DHCP server and DHCP client are on the same subnet.

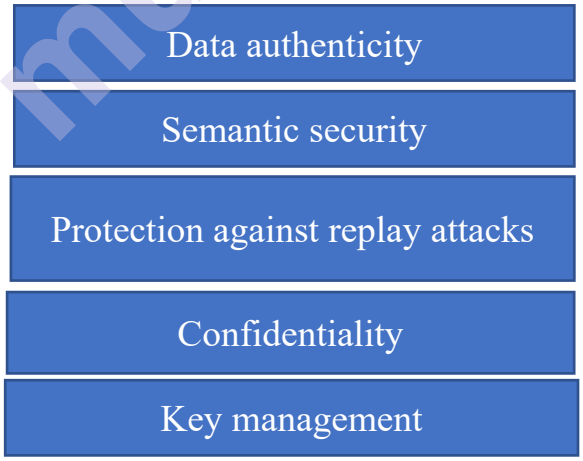
3.3.6 ICMP

- ICMP (Internet Control Message Protocol) is an error-reporting protocol network devices like routers use to generate error messages to the source IP address when network problems prevent delivery of IP packets. ICMP creates and sends messages to the source IP address indicating that a gateway to the Internet that a router, service or host cannot be reached for packet delivery. Any IP network device has the capability to send, receive or process ICMP messages.
- It is a network layer protocol used to address network communication problems across network devices. It is not a transport protocol that sends the data between the machine. The most common internet control message protocol has been used in the router. Internet Control Message protocol has used for sending an IP packet larger than the bytes permitted under the IP Protocol for executing dos attacks.
- ICMP is not a transport protocol that sends data between systems.
- While ICMP is not used regularly in end-user applications, it is used by network administrators to troubleshoot Internet connections in diagnostic utilities including ping and traceroute.
- One of the main protocols of the Internet Protocol suite, ICMP is used by routers, intermediary devices or hosts to communicate error

information or updates to other routers, intermediary devices or hosts. The widely used IPv4 (Internet Protocol version 4) and the newer IPv6 use similar versions of the ICMP protocol (ICMPv4 and ICMPv6, respectively).

3.3.7 RPL

- RPL offers different level of security by utilizing a Security field after the 4-byte ICMPv6 message header. Information in this field indicates the level of security and the cryptography algorithm used to encrypt the message.
- RPL offers support:



➢ **Data Authenticity:**

Data can be assumed to be authentic if it is provable that it has not been corrupted after its creation.

➢ **Semantic security:**

In cryptography, a semantically secure cryptosystem is one where only negligible information about the plaintext can be feasibly extracted from the ciphertext

➢ **Protection against replay attack:**

- Replay attack is a form of network attack in which valid data transmission is maliciously or fraudulently repeated or delayed.
- Replay attacks can be prevented by tagging each encrypted component with a session ID and a component number.

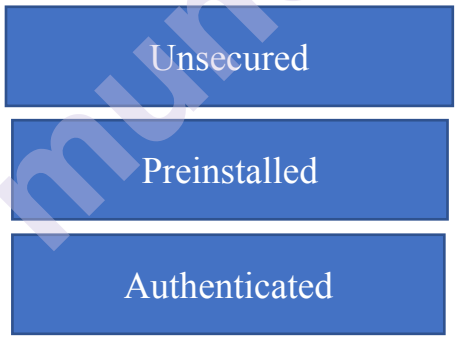
➢ **Confidentiality:**

Confidentiality involves a set of rules or a promise usually executed through confidentiality agreements that limits access or places restrictions on certain types of information.

➢ **Key Management:**

It deals with the key generation, exchange, storage, use, crypto-shredding (destruction) and replacement of keys.

- Levels of security in RPL include:



➢ **Unsecured:**

The RPL messages are sent without any security protection.

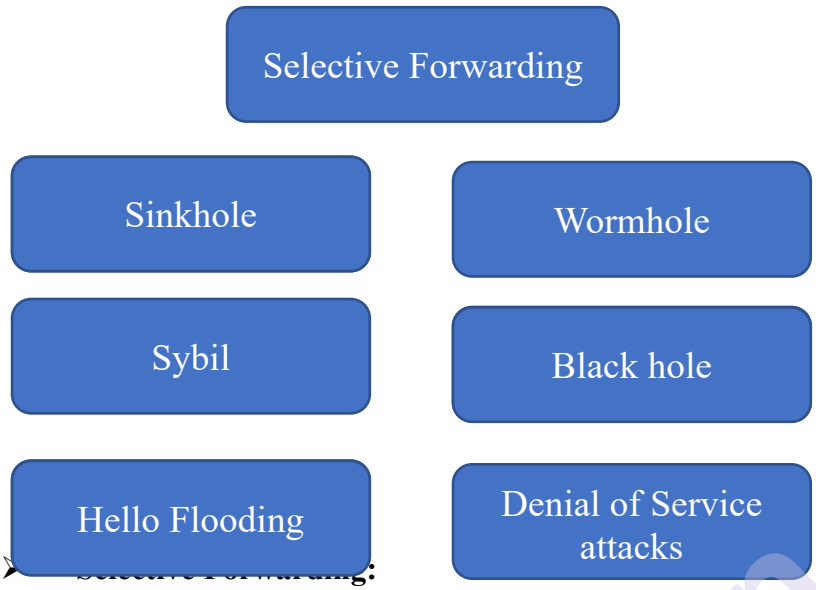
➢ **Preinstalled:**

key assumed to be already present in each node at boot time

➢ **Authenticated:**

The RPL messages are protected

- **RPL attacks include:**



➢ **Selective Forwarding:**

Malicious nodes turn down the request of facilitating some packets of information and makes sure that they are not passed on any further.

➢ **Sinkhole:**

Compromised node tries to attract network traffic by advertise its fake routing update.

➢ **Sybil:**

The attacker subverts the reputation system of a network service by creating a large number of pseudonymous identities and uses them to gain a disproportionately large influence.

➢ **Hello Flooding:**

Illegal node in the network can flood hello request to any legitimate node and break the security of WSN.

➢ **Wormhole:**

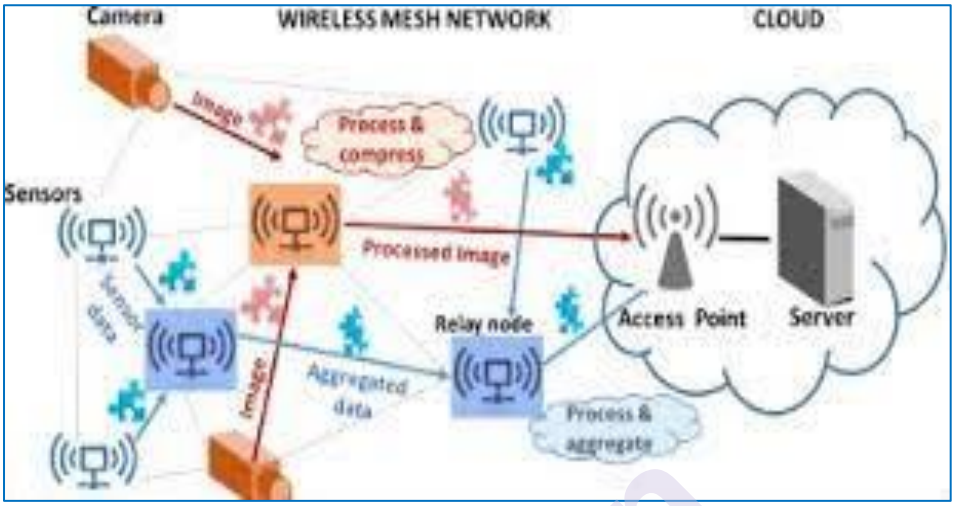
- It is a grave attack in which two attackers locate themselves strategically in the network.
- Then the attackers keep on listening to the network, and record the wireless information.

➢ **Black hole:**

In this attack, malicious node capture all the data and without having any active route it falsely replies for any route request thus prevent the source node to communicate with the destination node.

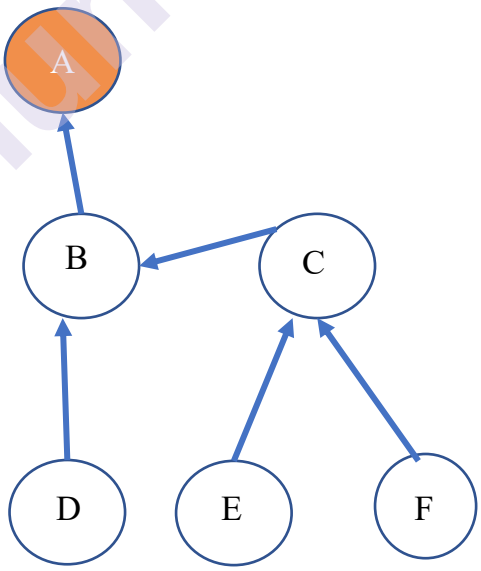
➢ **Denial of Service attacks.**

An attack meant to shut down a machine or network, making it inaccessible to its intended users.



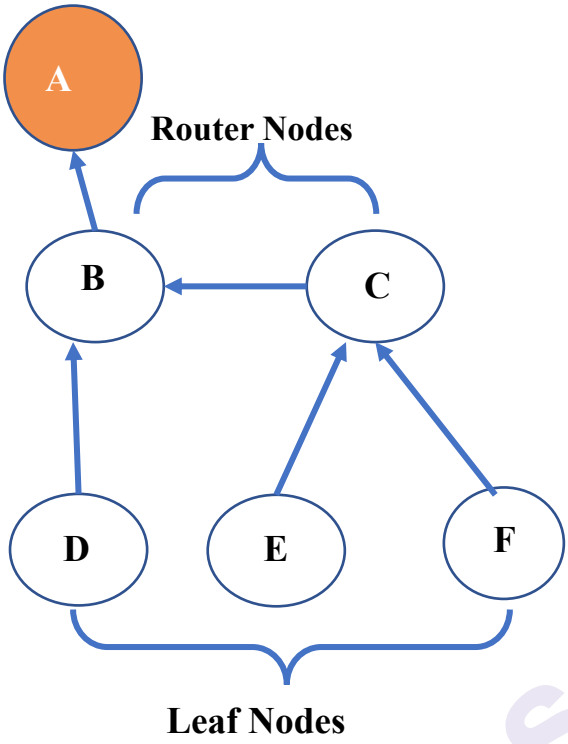
3.3.8 CORPL

- An extension of RPL is CORPL, or cognitive RPL, which is designed for cognitive networks and uses DODAG topology generation but with two new modifications to RPL. CORPL utilizes opportunistic forwarding to forward the packet by choosing multiple forwarders (forwarder set) and coordinates between the nodes to choose the best next hop to forward the packet to.



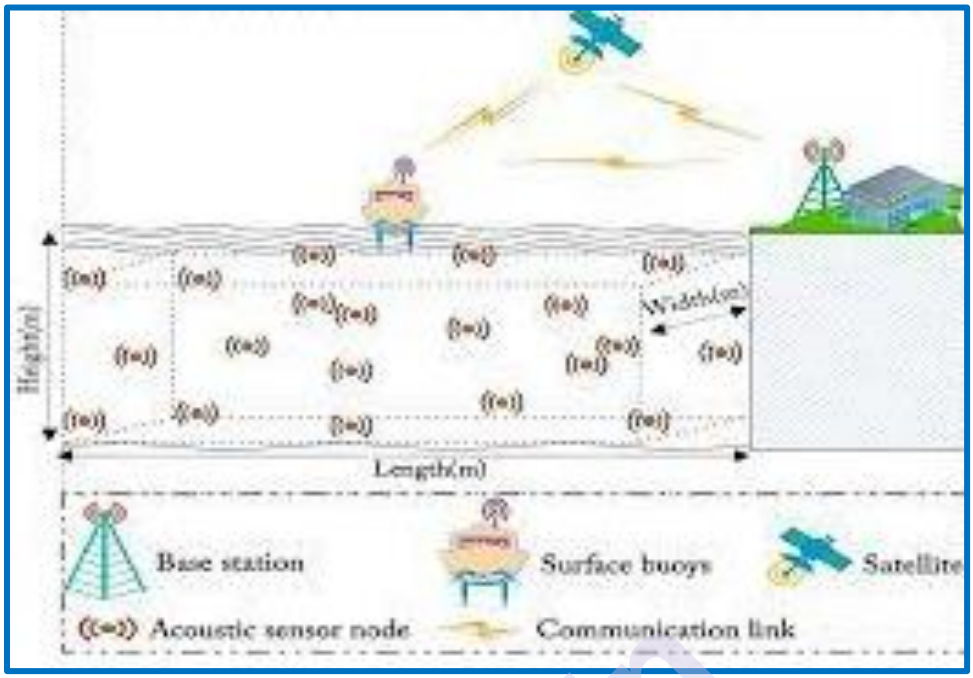
- DODAG is built in the same way as RPL. Each node maintains a forwarding set instead of its parent only and updates its neighbor with its changes using DIO messages. Based on the updated information, each node dynamically updates its neighbor priorities in order to

construct the forwarder set.



3.3.9 CARP

- Channel-Aware Routing Protocol (CARP) is a distributed routing protocol designed for underwater communication. It can be used for IoT due to its lightweight packets. It considers link quality, which is computed based on historical successful data transmission gathered from neighboring sensors, to select the forwarding nodes.
- There are two scenarios: network initialization and data forwarding.
    - In network initialization, a HELLO packet is broadcasted from the sink to all other nodes in the networks.
    - In data forwarding, the packet is routed from sensor to sink in a hop- by-hop fashion. Each next hop is determined independently.
- The main problem with CARP is that it does not support reusability of previously collected data. In other words, if the application requires sensor data only when it changes significantly, then CARP data forwarding is not beneficial to that specific application. An enhancement of CARP was done in E-CARP by allowing the sink node to save previously received sensory data. When new data is needed, E-CARP sends a Ping packet which is replied with the data from the sensor's nodes. Thus, E-CARP reduces the communication overhead drastically.



3.4 LIST OF REFERENCES

1. From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence, Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle,1st Edition, Academic Press, 2014.

2. Building the Internet of Things with IPv6 and MIPv6: The Evolving World of M2M Communications, Daniel Minoli, Wiley Publications,2013

3.5 BIBLIOGRAPHY

1. https://www.dataversity.net/brief-history-internet-things/

2. https://iot-analytics.com/internet-of-things-definition/

3.6 UNIT END EXERCISES

1. Which level is the network layer in the OSI model?
   a) Third level
   b) Fourth level
   c) Second level
   d) Fifth layer

2. Data in network layer is transferred in the form of _____

   a) Layers
   b) Packets
   c) Bytes
   d) Bits

3. What are the functions of the transport layer?

a) Multiplexing/ Demultiplexing
b) Connection less Services
c) Connection oriented service
d) Congestion control

4. Which services are provided by transport layer?

a) Error control
b) Connection service
c) Connection less service
d) Congestion control

5. TCP and UDP are called _____

a) Application protocols
b) Session protocols
c) Transport protocols
d) Network protocols

6. Security based connection is provided by which layer?

a) Network layer
b) Session layer
c) Application layer
d) Transport layer

7. Using which method in transport layer data integrity can be ensured?

a) Checksum
b) Repetition codes
c) Cyclic redundancy checks
d) Error correcting codes

8. Transport layer can identify the symptoms of overload nodes using _____

a) Flow control
b) Traffic control
c) Byte orientation
d) Data integrity

9. Transport layer receives data in the form of _____

a) Packets
b) Byte streams
c) Bits stream
d) Both packets and Byte stream

10. UDP packets are called as _____

a) Segments
b) Checksum
c) Frames
d) Datagrams

11. Which of the following IoT networks has a very short range?
    a) Short Network
    b) LPWAN
    c) Sig Fox
    d) Short-range Wireless Network
12. What is the full form of the LPWAN?
    a) Low Protocol Wide Area Network
    b) Low Power Wide Area Network
    c) Long Protocol Wide Area Network
    d) Long Power Wide Area Network
13. An IoT network is a collection of _____ devices.
    a) Signal
    b) Machine to Machine
    c) Interconnected
    d) Network to Network
14. What is the full form of HART?
    a) Highway Application Remote Transport
    b) Highway Addressable Remote Transducer
    c) High Address Reduce Transport
    d) High Application Remote Transport
15. What is the frequency rate of z-wave?
    a) 908.42 GHz
    b) 928.49 GHz
    c) 888.42 GHz
    d) 708.49 GHz

❄❄❄❄❄❄❄

4

TRANSPORT LAYER

Unit Structure :

4.1 OBJECTIVES

At the end of this unit, the student will able to

- ✓ Understand the use of all IOT transport layer protocol.
- ✓ Explain the challenges associated with these protocols in IOT architecture.
- ✓ Illustrate the architecture of these transport layer protocol.

4.2 TRANSPORT LAYER

1. TCP (Transmission Control Protocol) is a widely used transport layer protocol that provides reliable and ordered delivery of data between applications over IP networks. TCP is one of the most commonly used protocols in the Internet of Things (IoT) as it ensures reliable data transfer between IoT devices and applications.

2. In the IoT, TCP is used to provide a reliable and secure communication channel between IoT devices and applications. IoT devices often have limited processing power and memory, and may operate in low-bandwidth or high-latency environments.

3. TCP's reliability mechanisms such as flow control, congestion control, and error detection and recovery help ensure that data is delivered correctly and in a timely manner, even in these challenging environments.

4. TCP operates on top of the Internet Protocol (IP) and uses a connection-oriented communication model. A connection is established between two devices (the sender and the receiver) before data is exchanged.

5. The connection setup process involves a three-way handshake, where the sender sends a SYN (synchronize) packet to the receiver, the receiver responds with a SYN-ACK (synchronize-acknowledge) packet, and the sender acknowledges the receipt of the SYN-ACK packet with an ACK (acknowledge) packet.

6. Once the connection is established, data is exchanged using TCP segments. TCP segments consist of a header and a payload. The header contains control information such as sequence and acknowledgement numbers, window size, and flags. The payload contains the data being transferred.

7. TCP provides several mechanisms to ensure reliable data transfer. These include are as follows

8. Flow control: TCP uses a sliding window mechanism to control the amount of data sent by the sender at any given time. The receiver advertises a window size that specifies the amount of data it is willing to receive. The sender can then send data up to the size of the window. This helps prevent the receiver from being overwhelmed with data it cannot handle.

9. Congestion control: TCP's congestion control mechanism prevents the network from becoming congested with too much traffic. TCP monitors network congestion by tracking packet loss and adjusting the sending rate accordingly.

10. Error detection and recovery: TCP uses a checksum to detect errors in transmitted data. If an error is detected, TCP requests the retransmission of the corrupted data.

11. TCP is a reliable and secure protocol, but it has some drawbacks in the IoT. It requires a significant amount of overhead for connection setup and maintenance, which can be a problem in low-power, low-bandwidth devices. TCP is also not well-suited for real-time applications that require low latency and high throughput.

12. To address these limitations, several variants of TCP have been developed for the IoT, such as Lightweight TCP (LwTCP) and Constrained Application Protocol (CoAP). These protocols are designed to be lightweight and energy-efficient, while still providing reliable and secure communication.

13. In conclusion, TCP is an important protocol in the IoT that provides reliable and secure communication between IoT devices and applications. Its reliability mechanisms ensure that data is delivered correctly and in a timely manner, even in challenging environments.

However, its high overhead and latency make it unsuitable for some IoT applications.

14. TCP (Transmission Control Protocol) is a widely used transport layer protocol in the Internet Protocol (IP) suite. It provides reliable, ordered, and error-checked delivery of data between applications running on devices connected to the Internet. In the context of the Internet of Things (IoT), TCP is an important protocol for facilitating communication between IoT devices and applications.

15. TCP provides a number of important features that make it well-suited for IoT applications. One of the most important is reliable, error-checked data delivery. TCP uses a system of acknowledgments and retransmissions to ensure that data sent from one device to another is received without errors. This is particularly important in IoT applications where data may be critical for safety, security, or operational purposes.

16. Another important feature of TCP is its ability to manage congestion in the network. TCP uses a mechanism called "congestion control" to detect and respond to congestion in the network. When congestion is detected, TCP reduces the rate at which data is transmitted to prevent the network from becoming overloaded. This is important in IoT applications where devices may be located in remote or difficult-to-reach locations and network resources may be limited.

17. TCP is also designed to work seamlessly with IP, the underlying protocol used for communication on the Internet. TCP and IP together form the TCP/IP protocol suite, which is the foundation of the Internet. This means that TCP can be used to transmit data between any two devices connected to the Internet, regardless of their location or the type of device.

18. One of the challenges of using TCP in IoT applications is that TCP was originally designed for use in traditional computer networks and may not be optimized for the unique characteristics of IoT networks. For example, IoT networks may include devices with limited processing power and memory, and may operate in low-power, low-bandwidth environments. These factors may affect the performance of TCP in IoT applications.

19. To address these challenges, there have been efforts to develop TCP variants that are specifically designed for use in IoT networks. One example is TCP-LP (Low-Priority Transport Control Protocol), which is designed to reduce the amount of energy consumed by IoT devices during communication by using a low-priority data transmission mechanism. Another example is TCP-NUD (Network Unresponsive Detection), which is designed to improve the performance of TCP in mobile IoT networks by detecting network congestion and adjusting TCP's congestion control algorithms accordingly.

20. In summary, TCP is an important protocol for facilitating communication between IoT devices and applications. Its reliable, error-checked data delivery and congestion control mechanisms make it well-suited for IoT applications where data may be critical for safety, security, or operational purposes. However, challenges remain in optimizing TCP for the unique characteristics of IoT networks, and ongoing efforts are being made to develop TCP variants that are specifically designed for use in IoT applications.

4.3 MPTCP

1. MPTCP (Multipath TCP) is an extension to the traditional TCP protocol that enables the simultaneous use of multiple network paths for data transmission. In the context of the Internet of Things (IoT), MPTCP can provide improved reliability and performance by enabling data to be transmitted over multiple paths simultaneously.

2. Traditional TCP is designed to work over a single network path between two devices. If the network path fails, TCP will automatically attempt to retransmit the data over the same path until it is successful or until a timeout occurs. This can be problematic in IoT applications where devices may be located in remote or difficult-to-reach locations, and network paths may be prone to failure.

3. MPTCP enables data to be transmitted over multiple network paths simultaneously. This can provide improved reliability by reducing the likelihood of data loss due to network failures. In addition, MPTCP can provide improved performance by using multiple paths to transmit data, which can increase the available bandwidth and reduce the overall latency of the transmission.

4. MPTCP works by splitting data into subflows and sending them over different network paths. Each subflow is assigned a sequence number, which enables the receiver to reconstruct the original data once all subflows have been received. MPTCP also includes mechanisms for detecting and responding to network failures and congestion, including the ability to dynamically adjust the number of subflows used for data transmission.

5. MPTCP has been shown to be effective in improving the reliability and performance of IoT applications. For example, in a study of MPTCP for IoT sensor networks, researchers found that MPTCP was able to significantly improve data delivery rates and reduce the amount of data loss due to network failures.

6. However, there are also challenges associated with using MPTCP in IoT applications. One of the main challenges is the increased overhead associated with using multiple network paths, which can consume additional processing power and memory on IoT devices. In addition, MPTCP may not be well-suited for certain types of IoT applications, such as those that require real-time data transmission.

7. In summary, MPTCP is an extension to the traditional TCP protocol that enables the simultaneous use of multiple network paths for data transmission. It can provide improved reliability and performance for IoT applications by reducing the likelihood of data loss due to network failures and increasing the available bandwidth for data transmission. However, challenges remain in optimizing MPTCP for the unique characteristics of IoT networks, and ongoing research is being conducted to address these challenges.

8. Traditional TCP uses a single network path for communication between two devices. If this path is congested or experiences network failures, the performance of the communication can be negatively affected. MPTCP addresses this issue by allowing data to be transmitted over multiple network paths simultaneously. This provides several benefits for IoT applications:

9. Improved reliability: By using multiple network paths, MPTCP can provide more reliable data transmission. If one path becomes congested or fails, data can be seamlessly rerouted through another path.

10. Faster data transmission: MPTCP can use multiple paths to transmit data simultaneously, resulting in faster data transmission rates. This is particularly useful for IoT applications that require real-time data processing.

11. Better resource utilization: MPTCP can distribute data transmission across multiple network paths, which can help to balance the load on each path and reduce the risk of network congestion. This can help to improve the overall performance of the network.

12. MPTCP is particularly well-suited for IoT applications that involve devices with multiple network interfaces, such as smartphones, tablets, and IoT gateways. These devices can use multiple wireless or wired network connections to communicate with other devices or applications. MPTCP can be used to take advantage of these multiple network interfaces, providing improved performance and reliability.

13. There are several challenges associated with using MPTCP in IoT applications. One challenge is the increased complexity of managing multiple network paths. MPTCP requires additional processing power and memory to manage multiple network paths, which may be a limitation for devices with limited computing resources.

14. Another challenge is the increased power consumption associated with using multiple network paths. MPTCP may increase power consumption by requiring devices to maintain multiple network connections simultaneously. This can be a limitation for IoT devices that operate on battery power.

15. In summary, MPTCP is an extension of the TCP protocol that enables the use of multiple network paths for communication between two devices. MPTCP can provide improved reliability, faster data transmission, and better resource utilization for IoT applications. However, there are several challenges associated with using MPTCP in IoT applications, including increased complexity and power consumption.

4.4 UDP

1. UDP (User Datagram Protocol) is a transport layer protocol that provides a connectionless and unreliable data transfer mechanism between devices in the Internet of Things (IoT) ecosystem. Unlike TCP (Transmission Control Protocol), which provides reliable data transfer, UDP does not establish a virtual circuit between devices and does not guarantee the delivery of data packets.

2. UDP is commonly used in IoT applications that require fast and lightweight data transfer. Examples include real-time data streaming, such as video and audio streaming, and IoT applications that require low-latency communication, such as smart home automation systems.

3. One advantage of using UDP in IoT applications is its low overhead, which makes it an efficient protocol for small, low-power devices that have limited processing power and battery life. UDP also allows for multicast and broadcast communication, enabling a single message to be sent to multiple devices simultaneously, which is particularly useful in IoT applications that involve group communication.

4. However, the lack of reliability in UDP means that data packets may be lost or arrive out of order, which can be problematic for some IoT applications. For example, in a smart home automation system, a lost packet could result in a failure to control a device, such as a thermostat or light switch. In addition, UDP does not provide congestion control, which can lead to network congestion and degraded performance in high-traffic environments.

5. To address these issues, some IoT applications may use a combination of UDP and other protocols, such as TCP or SCTP (Stream Control Transmission Protocol), to provide a balance between reliability and efficiency. For example, an IoT application may use UDP for real-time data streaming, while using TCP for configuration and control messages.

6. In summary, UDP is a lightweight and efficient protocol that is well-suited for IoT applications that require fast and low-latency communication. However, the lack of reliability and congestion control in UDP can be a limitation for some IoT applications, which may require a combination of UDP and other protocols to provide a balance between efficiency and reliability.

7. UDP (User Datagram Protocol) is a connectionless, unreliable transport protocol that is often used in IoT applications. Unlike TCP (Transmission Control Protocol), which guarantees reliable data transmission by implementing a range of error checking mechanisms, UDP does not provide any such guarantees. Instead, UDP is a lightweight protocol that is designed for applications that require fast data transmission and can tolerate some level of data loss.

8. In the context of IoT, UDP is often used for applications that require real-time data processing, such as multimedia streaming, voice over IP (VoIP), and sensor data collection. These applications can benefit from the reduced overhead and latency that UDP provides.

9. One advantage of UDP in IoT applications is its simplicity. UDP is a lightweight protocol that requires fewer computing resources than TCP, making it ideal for devices with limited processing power, such as sensors and actuators. Additionally, UDP is a connectionless protocol, which means that it does not require the establishment and maintenance of a connection before data transmission can occur. This can help to reduce network latency and improve the responsiveness of IoT applications.

10. However, one disadvantage of UDP is its lack of reliability. Because UDP does not implement any error checking mechanisms, there is no guarantee that data will be transmitted successfully from the sender to the receiver. This can be a limitation for IoT applications that require reliable data transmission, such as those that involve critical infrastructure or medical devices.

11. Another disadvantage of UDP is the potential for network congestion. Because UDP does not implement any congestion control mechanisms, it is possible for UDP packets to be dropped or delayed if network congestion occurs. This can lead to a reduction in overall network performance and can impact the performance of other applications that share the same network resources.

12. In summary, UDP is a lightweight transport protocol that is often used in IoT applications that require fast data transmission and can tolerate some level of data loss. While UDP provides advantages in terms of simplicity and reduced latency, its lack of reliability and potential for network congestion can be limitations for certain IoT applications. As with any protocol, the decision to use UDP in an IoT application should be based on an understanding of the specific requirements and limitations of the application.

4.5 DCCP

1. DCCP (Datagram Congestion Control Protocol) is a transport protocol designed for use in applications that require congestion control over unreliable network connections, such as the Internet of Things (IoT). Unlike TCP (Transmission Control Protocol) and UDP

(User Datagram Protocol), DCCP is designed specifically for real-time multimedia streaming and other time-sensitive applications.

2. One of the main advantages of DCCP is its ability to provide congestion control while maintaining low latency. This is achieved through the use of a congestion control mechanism that is designed to be responsive to changes in network conditions. DCCP uses feedback mechanisms to monitor network congestion and adjust the rate at which data is transmitted accordingly. This can help to prevent network congestion and maintain a high level of network performance.

3. Another advantage of DCCP is its support for different congestion control algorithms. This allows applications to choose the algorithm that is best suited for their specific requirements. For example, some applications may require a more aggressive congestion control algorithm to ensure reliable data transmission, while others may prioritize low latency and prefer a less aggressive algorithm.

4. DCCP also provides support for a range of reliability options. Applications can choose between reliable and unreliable delivery modes, depending on their specific requirements. Reliable delivery mode ensures that all packets are delivered to the destination, while unreliable delivery mode allows for faster data transmission at the cost of some potential data loss.

5. However, one disadvantage of DCCP is its relatively limited support in comparison to TCP and UDP. DCCP is not widely supported by all network devices and operating systems, which can make it difficult to implement in certain IoT environments. Additionally, DCCP is not designed to provide the same level of reliability as TCP, which can be a limitation for some applications that require reliable data transmission.

6. In summary, DCCP is a transport protocol designed for use in IoT applications that require congestion control over unreliable network connections. DCCP provides low latency and support for different congestion control algorithms, but its limited support and lack of reliability may be limitations in certain IoT environments. As with any protocol, the decision to use DCCP in an IoT application should be based on an understanding of the specific requirements and limitations of the application.

7. DCCP (Datagram Congestion Control Protocol) is a transport protocol that is designed for use in applications that require low latency and congestion control, making it a good fit for IoT applications. DCCP is based on UDP (User Datagram Protocol) and provides congestion control capabilities that are not present in UDP.

8. One advantage of DCCP is its ability to support congestion control for real-time applications, such as multimedia streaming, voice over IP (VoIP), and gaming. Unlike UDP, which does not provide any

congestion control, DCCP implements a range of congestion control mechanisms that enable it to adapt to changing network conditions and ensure that data is transmitted efficiently and reliably.

9. Another advantage of DCCP is its support for multiple streams within a single connection. This can help to improve network efficiency by enabling multiple data streams to be transmitted over a single connection, reducing the number of connections required and the associated overhead.

10. In the context of IoT, DCCP is well-suited for applications that require low latency and reliable data transmission, such as industrial automation, remote sensing, and monitoring. These applications can benefit from the congestion control capabilities provided by DCCP, which can help to ensure that data is transmitted efficiently and reliably over the network.

11. However, one limitation of DCCP is its lack of widespread adoption. While DCCP has been standardized and is supported by some operating systems and network devices, it is not as widely adopted as other transport protocols such as TCP and UDP. This can limit its usefulness in certain IoT applications where interoperability with existing systems is important.

12. In summary, DCCP is a transport protocol that provides congestion control capabilities for applications that require low latency and reliable data transmission. While DCCP has some advantages over UDP and other transport protocols, its lack of widespread adoption may limit its usefulness in certain IoT applications. As with any protocol, the decision to use DCCP in an IoT application should be based on an understanding of the specific requirements and limitations of the application.

4.6 SCTP

1. SCTP (Stream Control Transmission Protocol) is a transport protocol that is designed to provide reliable, ordered, and flow-controlled delivery of data over IP networks. It is particularly well-suited for IoT applications that require high reliability and fault tolerance, such as industrial automation, medical devices, and smart grids.

2. One of the main advantages of SCTP is its support for multiple streams and the ability to transmit data in separate streams within a single connection. This can be useful in IoT applications where different types of data are transmitted over the same connection, as it allows for better organization and management of the data. In addition, SCTP provides several congestion control mechanisms that enable it to adapt to changing network conditions and ensure reliable transmission of data.

3. Another advantage of SCTP is its support for multi-homing, which allows a single endpoint to have multiple IP addresses and network

interfaces. This provides additional fault tolerance and can help to ensure that data is transmitted even in the event of network failures or outages.

4. In the context of IoT, SCTP is particularly useful in applications that require high reliability and fault tolerance, such as those in industrial automation and healthcare. These applications can benefit from the advanced features provided by SCTP, such as support for multiple streams and multi-homing, which help to ensure reliable transmission of data over the network.

5. However, like any protocol, there are some limitations to the use of SCTP. One potential drawback is its relatively low adoption rate compared to other transport protocols such as TCP and UDP. This can limit interoperability with other devices and systems, which may be important in some IoT applications.

6. SCTP is a transport protocol that provides advanced features such as support for multiple streams and multi-homing, making it particularly well-suited for IoT applications that require high reliability and fault tolerance. While SCTP has some advantages over other transport protocols, its relatively low adoption rate may limit its usefulness in certain IoT applications. As with any protocol, the decision to use SCTP in an IoT application should be based on an understanding of the specific requirements and limitations of the application.

7. SCTP (Stream Control Transmission Protocol) is a transport protocol that is designed to provide reliable, message-oriented communication between network endpoints. SCTP is a good fit for IoT applications that require reliable transmission of messages, as it provides features such as message fragmentation, retransmission, and flow control.

8. One advantage of SCTP is its ability to support multiple streams within a single connection. This can help to improve network efficiency by enabling multiple data streams to be transmitted over a single connection, reducing the number of connections required and the associated overhead. In addition, SCTP provides congestion control mechanisms that enable it to adapt to changing network conditions and ensure that data is transmitted efficiently and reliably.

9. Another advantage of SCTP is its support for multi-homing, which allows a single endpoint to have multiple IP addresses. This can provide increased reliability and resilience in IoT applications where network connectivity may be unreliable or where redundancy is required.

10. In the context of IoT, SCTP is well-suited for applications that require reliable, message-oriented communication, such as smart grid systems, remote sensing, and monitoring. These applications can benefit from the message-oriented nature of SCTP, which can help to ensure that data is transmitted reliably and efficiently over the network.

11. However, one limitation of SCTP is its lack of widespread adoption. While SCTP has been standardized and is supported by some operating systems and network devices, it is not as widely adopted as other transport protocols such as TCP and UDP. This can limit its usefulness in certain IoT applications where interoperability with existing systems is important.

12. In summary, SCTP is a transport protocol that provides reliable, message-oriented communication for IoT applications. While SCTP has some advantages over other transport protocols, its lack of widespread adoption may limit its usefulness in certain IoT applications. As with any protocol, the decision to use SCTP in an IoT application should be based on an understanding of the specific requirements and limitations of the application.

4.7 TLS

1. TLS (Transport Layer Security) is a cryptographic protocol that is used to provide secure communication over the Internet. TLS is widely used in IoT applications to ensure that data is transmitted securely and confidentially over the network.

2. One of the main advantages of TLS is its ability to provide end-to-end encryption of data. This means that data is encrypted before it is transmitted over the network, and decrypted only by the intended recipient. This helps to ensure that data is protected from unauthorized access and that it remains confidential throughout its transmission.

3. Another advantage of TLS is its support for authentication and integrity verification. TLS uses digital certificates to verify the identity of the sender and receiver of data, and to ensure that data has not been tampered with during transmission. This helps to prevent man-in-the-middle attacks and other types of attacks that can compromise the security of IoT applications.

4. In addition, TLS provides support for different encryption algorithms and key sizes, which can be customized to meet the specific security requirements of an IoT

5. application. This allows IoT applications to balance security requirements against performance and resource constraints.

6. However, the use of TLS in IoT applications can also introduce some challenges. For example, TLS requires significant processing power and memory resources, which can be a limitation for IoT devices with limited resources. In addition, TLS can introduce additional latency and overhead, which can impact the performance of real-time IoT applications.

7. In summary, TLS is a widely used cryptographic protocol that provides end-to-end encryption, authentication, and integrity verification for IoT applications. While TLS has some advantages

over other security protocols, it can also introduce some challenges related to resource constraints and performance. As with any security protocol, the decision to use TLS in an IoT application should be based on an understanding of the specific requirements and limitations of the application.

8. TLS (Transport Layer Security) is a protocol that provides security for communication over the internet. TLS is widely used to secure web traffic and is also used in many IoT applications to provide secure communication between devices and servers.

9. TLS provides encryption of data that is transmitted over the network, which helps to prevent unauthorized access to sensitive information. In addition, TLS provides authentication of the communicating devices, which helps to prevent spoofing and man-in-the-middle attacks.

10. In the context of IoT, TLS is often used to secure communication between IoT devices and cloud servers. For example, a smart home device may use TLS to securely communicate with a cloud server that provides remote management and control of the device. TLS can also be used to secure communication between IoT devices, which is important in applications where the devices are located in public or unsecured areas.

11. TLS uses a system of digital certificates to authenticate the communicating devices. When a device connects to a server using TLS, the server sends a certificate to the device, which contains a public key that is used to encrypt the data transmitted between the two devices. The device verifies the certificate to ensure that it was issued by a trusted authority and that the public key belongs to the server that it is trying to connect to. This helps to prevent man-in-the-middle attacks, where an attacker intercepts the communication and impersonates one of the devices.

12. One challenge with using TLS in IoT applications is that it can be resource-intensive, requiring significant processing power and memory. This can be a problem for resource-constrained IoT devices that have limited processing power and memory. To address this issue, lightweight versions of TLS have been developed, such as TLS-PSK (Pre-Shared Key) and TLS-PSK-Identity, which are optimized for use in IoT devices with limited resources.

13. In summary, TLS is a protocol that provides security for communication over the internet and is widely used in IoT applications to provide secure communication between devices and servers. TLS provides encryption of data, authentication of devices, and protection against man-in-the-middle attacks. While TLS can be resource-intensive, lightweight versions of the protocol have been developed to support IoT devices with limited resources.

4.8 DTLS

1. DTLS (Datagram Transport Layer Security) is a protocol that is used to provide security for communication between devices in the Internet of Things (IoT). DTLS is a variant of the TLS protocol that is designed to work with datagram transport protocols, such as User Datagram Protocol (UDP), which are commonly used in IoT applications.

2. DTLS provides encryption, authentication, and integrity protection for datagram traffic, which helps to protect against eavesdropping, tampering, and other attacks. Like TLS, DTLS uses digital certificates to authenticate devices and provides protection against man-in-the-middle attacks.

3. In IoT applications, DTLS is commonly used to secure communication between devices, such as sensors and gateways, or between gateways and cloud servers. For example, a smart city application may use DTLS to secure communication between sensors and a gateway, and between the gateway and a cloud server that provides analytics and management services.

4. One advantage of using DTLS in IoT applications is that it can provide security for real-time communication, such as video streaming or voice over IP (VoIP), which may require low latency and high throughput. DTLS can also support devices with low processing power and memory by using a lightweight version of the protocol.

5. However, DTLS has some limitations in IoT applications. One limitation is that it can be vulnerable to denial-of-service (DoS) attacks, where an attacker floods the network with traffic, causing the devices to become unresponsive. To address this issue, techniques such as rate limiting and session resumption can be used to reduce the impact of DoS attacks.

6. DTLS is a protocol that is used to provide security for communication between devices in IoT applications. DTLS provides encryption, authentication, and integrity protection for datagram traffic, and is commonly used to secure real-time communication and low-power devices. However, DTLS has some limitations in IoT applications, such as vulnerability to DoS attacks, which need to be addressed to ensure secure and reliable communication.

7. DTLS (Datagram Transport Layer Security) is a variation of the TLS protocol that is specifically designed for use with datagram protocols such as UDP. DTLS is commonly used in IoT applications to provide secure communication between devices over unreliable networks such as wireless or cellular networks.

8. Like TLS, DTLS provides encryption of data transmitted over the network and authentication of the communicating devices. However, because DTLS is designed to work with datagram protocols, it uses a

different set of security mechanisms that are optimized for datagram transmission. DTLS uses a similar system of digital certificates to TLS to authenticate the communicating devices, but also includes mechanisms to handle packet loss, reordering, and duplication.

9. DTLS is particularly useful in IoT applications where devices have limited processing power and memory, as it can be implemented with minimal overhead. Additionally, because DTLS is designed to work with datagram protocols, it can provide real-time performance and low latency, making it well-suited for use in IoT applications such as smart homes and industrial automation.

10. One challenge with using DTLS in IoT applications is that it can be vulnerable to certain attacks, such as DoS (Denial of Service) attacks, which can overwhelm the device with a large number of requests. To address this issue, DTLS includes mechanisms to limit the number of requests that can be sent to a device, and to detect and respond to malicious requests.

11. In summary, DTLS is a variation of the TLS protocol that is optimized for use with datagram protocols such as UDP. DTLS provides encryption of data, authentication of devices, and mechanisms to handle packet loss, reordering, and duplication. DTLS is particularly useful in IoT applications where devices have limited processing power and memory, and where real-time performance and low latency are important. However, DTLS can be vulnerable to certain attacks, and must be implemented with appropriate security mechanisms to ensure the security and reliability of the IoT system.

4.9 SUMMARY

In this chapter we learnt about the following things

- TCP is an important protocol for facilitating communication between IoT devices and applications. Its reliable, error-checked data delivery and congestion control mechanisms make it well-suited for IoT applications where data may be critical for safety, security, or operational purposes.
- DTLS is a variation of the TLS protocol that is optimized for use with datagram protocols such as UDP.
- TLS can be resource-intensive, lightweight versions of the protocol have been developed to support IoT devices with limited resources.
- SCTP is a transport protocol that provides reliable, message-oriented communication for IoT applications.
- DCCP is a transport protocol that provides congestion control capabilities for applications that require low latency and reliable data transmission.

4.10 QUESTIONS

Q1. Explain in brief about TCP

Q2. Explain in brief about UDP

Q3. Illustrate the use of DCCP in IOT

Q4. Determine why there is need of SCTP protocol in IOT

Q5. Explain the importance of MPTCP in IOT

4.11 REFERENCES

1. https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot/index.html

❄❄❄❄❄❄❄❄

5

SESSION LAYER

Unit Structure :

5.1 OBJECTIVES

At the end of this unit, the student will be able to

- Understand the difference between the HTTP and MQTT.
- Illustrate the concept of COAP and XMPP.
- Explain the Use of AMQP with respect to IOT.

5.2 HTTP

1. It is used by World Wide Web (WWW) for defining how its messages are going to be transmitted and formatted. This protocol is responsible for the action that a server has to take while sending information over the network.

2. When a URL is being entered into the browser, this protocol sends an HTTP request to the server and then an HTTP response is sent back to the browser. This protocol is also responsible for the controlling of webpages on the World Wide Web for their formatting and representation.

3. HTTP was created to make documents available across the internet. HTTP servers only respond to requests from clients. HTTP deals with requests one at a time, with overhead such as authentication being carried out each time. HTTP does have some ability to pool TCP connections, but the overhead for each message remains.

4. It is most familiar to us as one of the enabling technologies that allows web browsers to work. Servers contain resources that are identified

by the URLs that have the basic familiar form: https://www.hivemq.com/mqtt-essentials/

5. To which HTTP clients can make requests: GET, PUT, DELETE and POST, to name the most common. In a normal web environment the simple model is a web browser retrieving web pages from a server with the GET method:

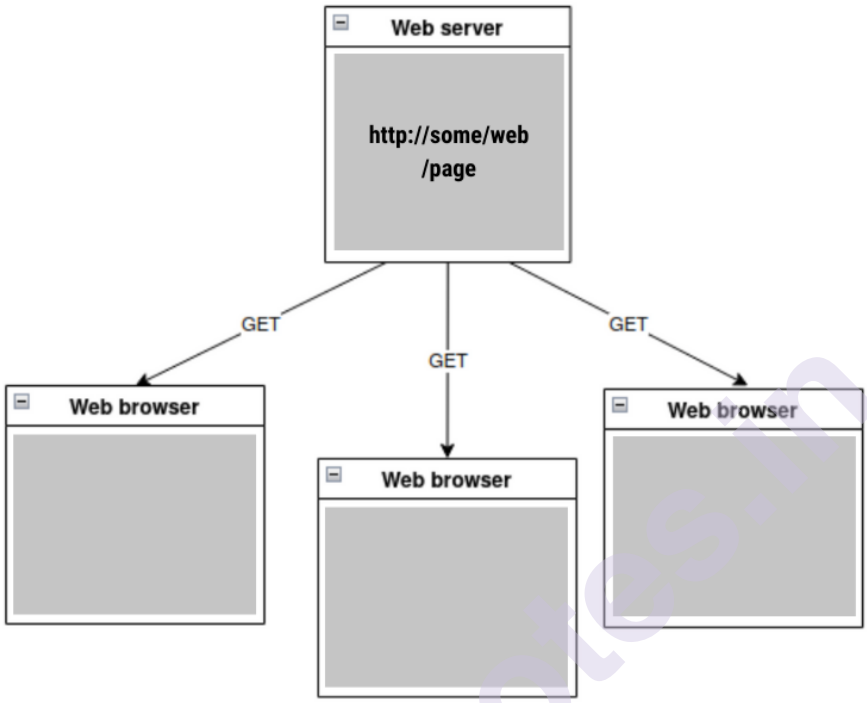


Fig 1 HTTP Get request

5.1 In an IoT environment, a common use of HTTP is to allow devices to POST to a resource that represents the device state on the IoT service:

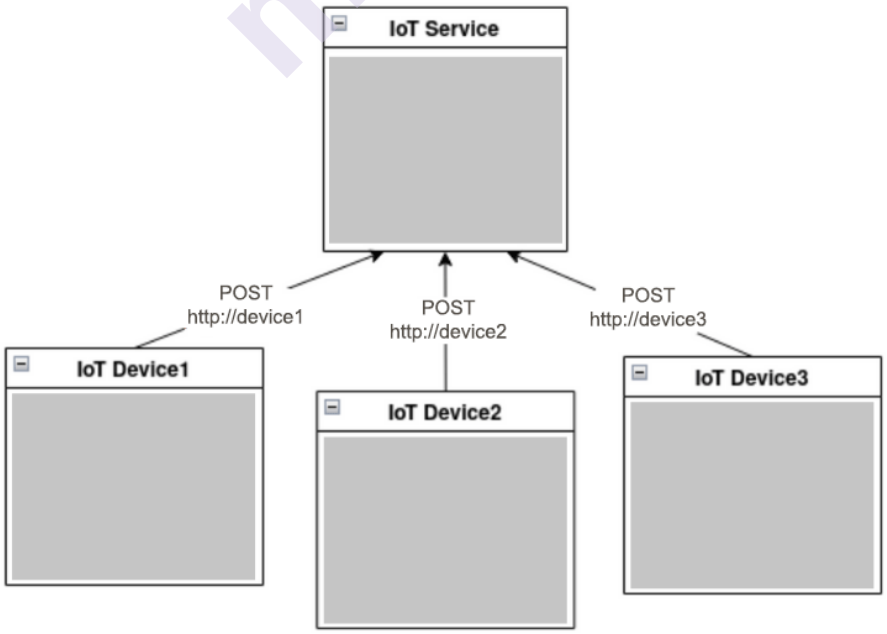


Fig 2 HTTP Put request

TABLE I HTTP performance

Points	HTTP bytes
Establish connection	2261
Disconnect	0
For each message published	3285
Sum for 1 message	5546
Sum for 10 messages	55,460
Sum for 100 messages	554,600

6. The equivalent “overhead” for HTTP has to be included with every request which is why the per message overhead is much bigger.

7. In the context of IoT, HTTP is often used for communication between IoT devices and the cloud. IoT devices collect data and send it to the cloud via HTTP requests. The cloud processes the data and sends back instructions to the devices via HTTP responses.

8. One of the benefits of using HTTP in IoT is its simplicity and widespread use. Many devices and software systems are already designed to support HTTP, making it easy to integrate IoT devices with existing systems. In addition, HTTP supports encryption through the use of SSL/TLS, which provides security for data transmission.

9. However, there are also some challenges with using HTTP in IoT. For example, HTTP was designed for use in human-readable web browsers, and as a result, it may not be the most efficient protocol for transmitting large amounts of data between IoT devices. In addition, the use of HTTP may result in higher power consumption and latency, which can be important considerations in IoT applications where energy efficiency and real-time response are critical.

10. Overall, HTTP can be a useful protocol for transmitting data in IoT applications, but its suitability depends on the specific requirements of the application. Other protocols such as MQTT, CoAP, and WebSocket may also be appropriate in different IoT contexts.

11. HTTP is a stateless protocol, which means that each request is treated as an independent transaction without any knowledge of previous requests. In IoT applications, HTTP is used to send and receive data between devices and web services, allowing for remote control and monitoring of devices.

12. Some common use cases for HTTP in IoT include:

12.1 Remote control: HTTP can be used to control IoT devices remotely from a web browser or mobile application. For example, a user could use a mobile app to turn on the lights in their home while they are away.

12.2 Data collection: HTTP can be used to send sensor data from IoT devices to a web service or cloud platform for processing and analysis. For example, a smart thermostat could send temperature data to a cloud platform to optimize energy usage.

12.3 Firmware updates: HTTP can be used to update the firmware on IoT devices remotely. For example, a security camera could receive a firmware update over HTTP to fix a security vulnerability.

12.4 Overall, HTTP is a versatile and widely used protocol in IoT applications, allowing for seamless communication between devices and web services or cloud platforms.

13. The principal advantage of HTTP for use in IoT is its familiarity to developers, many of whom have implemented web solutions of one kind or another. A consequence of this is the availability of client libraries and servers. However, MQTT is not far behind these days, a search for MQTT software will show up lots of options including HiveMQ's open source software. HTTP has no equivalent of many MQTT features that are useful for IoT such as queuing, QoS, retained messages and message push. Top of Form

5.3 COAP

1. CoAP (Constrained Application Protocol) is a protocol designed specifically for IoT devices with limited processing power and memory, low bandwidth and unreliable networks.

2. It is a lightweight protocol that uses UDP (User Datagram Protocol) for transmission and is suitable for low-power and lossy networks.

3. CoAP is based on the REST (Representational State Transfer) architectural style and allows IoT devices to interact with each other using simple request-response transactions.

4. It provides features such as caching, resource discovery, and observability, making it a suitable protocol for resource-constrained devices.

5. Some common use cases for CoAP in IoT include:

5.1 Smart homes: CoAP can be used to control smart home devices such as thermostats, lighting systems, and security cameras. For example, a user can send a CoAP request to turn on the lights in their home.

5.2 Industrial IoT: CoAP can be used in industrial IoT applications to monitor and control devices such as sensors and actuators. For example, a factory can use CoAP to monitor temperature and humidity levels in a production line.

5.3 Healthcare: CoAP can be used in healthcare applications to monitor patients' vital signs and send alerts in case of emergencies. For example, a wearable device can use CoAP to send heart rate and blood pressure data to a healthcare provider.

6. Overall, CoAP is a lightweight and efficient protocol that is well-suited for IoT devices with limited resources. It allows IoT devices to communicate with each other and with web services, making it an essential component of the IoT ecosystem.

7. CoAP is a lightweight protocol that is designed to be more efficient than HTTP (Hypertext Transfer Protocol) in terms of bandwidth usage, memory usage, and power consumption.

8. CoAP is a client-server protocol that uses UDP (User Datagram Protocol) as the transport layer protocol, rather than TCP (Transmission Control Protocol) used by HTTP.

9. CoAP has built-in support for reliable messaging, resource discovery, and lightweight security mechanisms.

10. CoAP in IoT include:

    10.1 Low-power devices: CoAP is designed to be used on low-power devices that have limited processing capabilities and memory. CoAP can be used to communicate between these devices and web services or cloud platforms.

    10.2 Resource-constrained networks: CoAP is designed to be used in networks with limited bandwidth and high latency, such as sensor networks. CoAP allows for efficient communication between devices and web services or cloud platforms in these networks.

    10.3 Machine-to-machine communication: CoAP is designed to be used for machine-to-machine communication, where devices need to communicate with each other without human intervention. CoAP can be used to enable devices to exchange information and control each other.

    10.4 Sensor data collection: CoAP can be used to collect sensor data from IoT devices and transmit it to a server or cloud platform for analysis. CoAP's lightweight design makes it suitable for use in low-power devices such as sensors.

    10.5 Actuator control: CoAP can be used to control actuators such as switches or motors in IoT devices. For example, a smart thermostat could use CoAP to adjust the temperature in a room.

    10.6 Firmware updates: CoAP can be used to remotely update firmware in IoT devices. For example, a security camera could receive a firmware update over CoAP to fix a security vulnerability.

11. Overall, CoAP is a useful protocol for IoT applications, particularly in resource-constrained environments where efficiency and low power consumption are critical.

5.4 XMPP

1. XMPP (Extensible Messaging and Presence Protocol) is a protocol that allows for real-time messaging and presence information exchange between devices over the internet. It is an open standard protocol that is widely used in instant messaging and chat applications.
2. In recent years, XMPP has gained popularity as a protocol for IoT (Internet of Things) applications due to its simplicity, scalability, and security.
3. XMPP protocol in the context of IoT applications, covering its architecture, key features, and use cases.
4. XMPP Architecture:
   - 4.1 XMPP is based on a client-server model, where clients communicate with each other or with servers over the internet. Each client or server is identified by a unique JID (Jabber ID), which is similar to an email address. XMPP messages are XML (eXtensible Markup Language) documents that can contain various types of information, including text, files, and commands.
   - 4.2 XMPP uses a publish-subscribe model for messaging, where devices subscribe to specific topics or channels and receive messages published by other devices or servers. XMPP servers also support presence information exchange, where devices can share their availability and status with other devices.

Fig 1 A simple XMPP Architecture

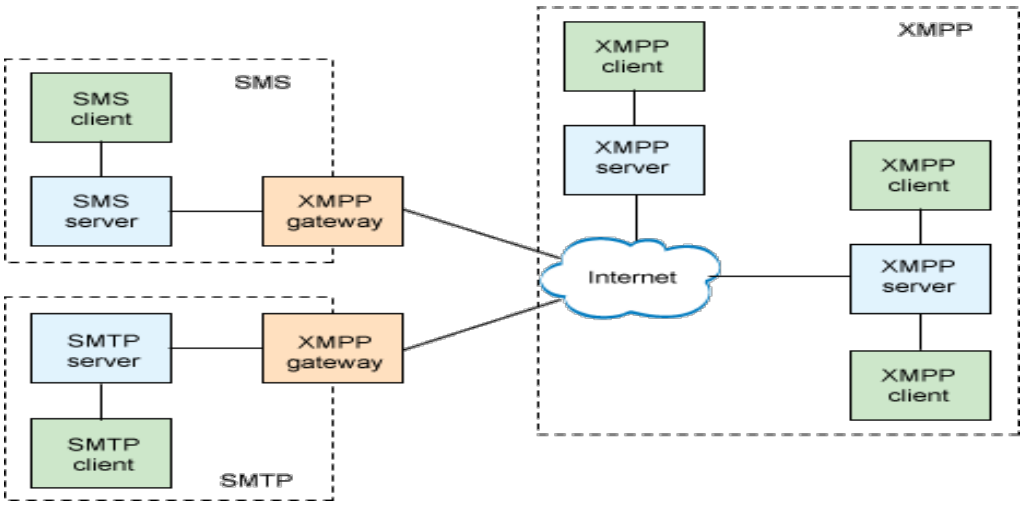


Fig 2 A Complex XMPP Architecture

5. XMPP Key Features:

   XMPP has several key features that make it a popular choice for IoT applications:

   5.1. Scalability: XMPP is designed to be scalable, allowing for large numbers of devices to connect and communicate over the internet.

   5.2. Security: XMPP supports encryption and authentication, ensuring that messages are sent and received securely.

   5.3. Real-time messaging: XMPP is designed for real-time messaging, allowing for fast and responsive communication between devices.

   5.4. Customization: XMPP is highly customizable, allowing for developers to add their own extensions and features to the protocol.

6. XMPP Use Cases:

   XMPP is used in a variety of IoT applications, including:

   6.1 Home automation: XMPP can be used to control and monitor home automation devices such as thermostats, lights, and security systems.

   6.2 Health monitoring: XMPP can be used to transmit health data from wearable devices to healthcare providers for monitoring and analysis.

   6.3 Industrial IoT: XMPP can be used to control and monitor industrial equipment and machines, allowing for real-time monitoring and predictive maintenance.

7. XMPP is a flexible and scalable protocol that is well suited for IoT applications. Its simplicity, real-time messaging capabilities, and security features make it a popular choice for developers looking to build IoT applications.

8. With its open standard and customizable architecture, XMPP is likely to continue to be a key protocol for IoT applications in the future.

9. Decentralized architecture: XMPP is based on a decentralized architecture, meaning that there is no single point of failure. This makes it more resilient and scalable than centralized protocols.

10. Real-time communication: XMPP is designed for real-time communication, with low latency and high throughput. This makes it well suited for IoT applications where quick response times are critical.

11. Security: XMPP supports end-to-end encryption and authentication, ensuring that messages are secure and private.

12. Extensibility: XMPP is highly extensible, with a wide range of extensions and plugins available to support various use cases.

13. Advantages of XMPP in IoT:

    XMPP has several advantages when used in IoT applications. These include:

    13.1 Interoperability: XMPP is a standardized protocol, ensuring interoperability between devices and services from different vendors.

    13.2 Flexibility: XMPP can be used for a wide range of IoT applications, including sensor data collection, device control, and firmware updates.

    13.3 Scalability: XMPP's decentralized architecture makes it highly scalable, allowing it to support large-scale IoT deployments.

    13.4 Low cost: XMPP is an open-source protocol, with no licensing fees, making it a cost-effective solution for IoT applications.

14 Limitations of XMPP in IoT:

    14.1 While XMPP has several advantages, there are also some limitations to its use in IoT applications. These include:

    14.2 Resource consumption: XMPP can consume a significant amount of resources, such as memory and processing power, particularly in low-power IoT devices.

    14.3 Complex implementation: XMPP's decentralized architecture and extensibility can make it complex to implement and maintain, particularly for small-scale deployments.

    14.4 Limited support for certain use cases: While XMPP can be used for a wide range of IoT applications, it may not be the best choice for certain use cases, such as high-bandwidth video streaming.

15 In conclusion, XMPP is a promising protocol for IoT applications, offering several advantages such as interoperability, flexibility, scalability, and low cost. However, it also has some limitations, particularly in terms of resource consumption and complex implementation. Overall, XMPP is well suited for IoT deployments where low latency and real-time communication are critical, and where interoperability and security are important considerations.

5.5 AMQP

1. AMQP (Advanced Message Queuing Protocol) is a messaging protocol that is commonly used in IoT deployments to enable communication between devices, sensors, and applications.

2. AMQP provides a reliable, secure, and interoperable way to exchange messages between IoT devices, as well as between IoT devices and backend systems.

3. AMQP is designed to handle large-scale, distributed systems, making it well-suited for IoT deployments where multiple devices and sensors need to communicate with each other and with backend systems. It uses a publish-subscribe model, where messages are published to a topic or exchange, and subscribers can receive messages from that topic or exchange based on their subscription.

4. Features of AMQP in IoT:

   AMQP has several features that make it well-suited for IoT deployments, including:

   4.1 Reliability: AMQP provides a reliable way to exchange messages between devices, ensuring that messages are delivered even in the event of network failures or other disruptions.

   4.2 Security: AMQP provides a secure way to exchange messages between devices, with support for encryption and authentication.

   4.3 Interoperability: AMQP is a standardized protocol, ensuring interoperability between devices and applications from different vendors.

   4.4 Scalability: AMQP is designed to handle large-scale, distributed systems, making it well-suited for IoT deployments with multiple devices and sensors.

5. Advantages of AMQP in IoT:

   AMQP has several advantages when used in IoT deployments, including:

   5.1 High throughput: AMQP is designed for high throughput, with support for both asynchronous and synchronous messaging.

   5.2 Low latency: AMQP supports real-time messaging, with low latency and high reliability.

   5.3 Flexibility: AMQP can be used for a wide range of IoT applications, including sensor data collection, device control, and firmware updates.

   5.4 Wide support: AMQP is supported by a wide range of devices, platforms, and programming languages.

6. Limitations of AMQP in IoT:

   6.1 While AMQP has several advantages, there are also some limitations to its use in IoT deployments. These include:

   6.2 Complexity: AMQP can be complex to implement and maintain, particularly for small-scale deployments.

   6.3 Resource consumption: AMQP can consume significant resources, such as memory and processing power, particularly in low-power IoT devices.

   6.4 Cost: Some AMQP implementations may require licensing fees, which can increase the cost of IoT deployments.

7. AMQP is a reliable, secure, and interoperable messaging protocol that is well-suited for IoT deployments. It provides high throughput, low latency, and flexibility, making it a good choice for a wide range of IoT applications.

8. However, it can be complex to implement and may consume significant resources, particularly in low-power IoT devices. Despite these limitations, AMQP is a popular choice for IoT deployments where reliability and interoperability are important considerations.

9. The Internet of Things (IoT) is a rapidly growing field with a wide range of applications, from home automation to industrial control systems. As the number of connected devices continues to grow, the need for standardized communication protocols becomes increasingly important. One such protocol that has gained popularity in recent years is the Advanced Message Queuing Protocol (AMQP).

10. AMQP is an open-source messaging protocol designed for reliable, asynchronous communication between applications. It was developed by a group of vendors and industry experts in response to the growing need for a standardized messaging protocol.

11. AMQP provides a way for applications to exchange messages in a scalable, secure, and efficient manner.

12. Features of AMQP:

    AMQP has several features that make it well suited for IoT applications. These include:

    12.1 Reliable messaging: AMQP guarantees reliable messaging, ensuring that messages are delivered and received in the correct order.

    12.2 Asynchronous communication: AMQP supports asynchronous communication, allowing messages to be sent and received without the need for a synchronous request/response model.

12.3 Scalability: AMQP is highly scalable, allowing it to handle large volumes of messages and devices.

12.4 Security: AMQP supports encryption and authentication, ensuring that messages are secure and private.

13 Advantages of AMQP in IoT:

AMQP has several advantages when used in IoT applications. These include:

13.1 Interoperability: AMQP is a standardized protocol, ensuring interoperability between devices and services from different vendors.

13.2 Flexibility: AMQP can be used for a wide range of IoT applications, including sensor data collection, device control, and firmware updates.

13.3 Scalability: AMQP's scalability makes it well suited for large-scale IoT deployments.

13.4 Fault tolerance: AMQP supports fault tolerance, allowing devices to recover from errors and continue operation.

14 Limitations of AMQP in IoT:

14.1 While AMQP has several advantages, there are also some limitations to its use in IoT applications. These include:

14.2 Complex implementation: AMQP can be complex to implement and maintain, particularly for small-scale deployments.

14.3 Resource consumption: AMQP can consume significant resources, such as memory and processing power, particularly in low-power IoT devices.

14.4 Latency: AMQP's reliability guarantees can lead to increased latency, which may not be suitable for real-time applications.

15 In conclusion, AMQP is a promising protocol for IoT applications, offering several advantages such as interoperability, flexibility, scalability, and fault tolerance. However, it also has some limitations, particularly in terms of complex implementation and resource consumption. Overall, AMQP is well suited for IoT deployments where reliable messaging and fault tolerance are critical, and where interoperability and security are important considerations.

5.6 MQTT

1. The Internet of Things (IoT) is rapidly evolving, and communication protocols play a critical role in connecting IoT devices. One of the

most popular IoT protocols is the Message Queuing Telemetry Transport (MQTT) protocol.

2. MQTT is an open-source, lightweight, and flexible messaging protocol that allows devices to communicate with each other over the Internet. In this essay, we will discuss MQTT in the context of IoT, exploring its features, advantages, and limitations.

3. MQTT is a messaging protocol that enables devices to communicate with each other over a network. It was developed by IBM in 1999 and has since become an open-source standard maintained by the Eclipse Foundation.

4. MQTT is designed to be lightweight and efficient, making it suitable for use in resource-constrained environments such as IoT devices.

5. Features of MQTT:

   MQTT has several features that make it well suited for IoT applications. These include:

   5.1 Lightweight: MQTT is designed to be lightweight, making it suitable for use in low-power, low-bandwidth IoT devices.

   5.2 Publish-subscribe architecture: MQTT uses a publish-subscribe architecture, where devices can subscribe to topics and receive messages from other devices that publish to those topics.

   5.3 Quality of Service (QoS): MQTT supports three levels of QoS, allowing devices to choose the level of reliability they require for message delivery.

   5.4 Security: MQTT supports encryption and authentication, ensuring that messages are secure and private.

6. Advantages of MQTT in IoT:

   MQTT has several advantages when used in IoT applications. These include:

   6.1 Efficiency: MQTT's lightweight design makes it highly efficient, reducing the resource requirements of IoT devices.

   6.2 Scalability: MQTT is highly scalable, allowing it to handle large volumes of messages and devices.

   6.3 Flexibility: MQTT can be used for a wide range of IoT applications, including sensor data collection, device control, and firmware updates.

   6.4 Reliability: MQTT's QoS levels ensure reliable message delivery, making it suitable for critical IoT applications.

7 Limitations of MQTT in IoT:

While MQTT has several advantages, there are also some limitations to its use in IoT applications. These include:

7.1 Limited data size: MQTT has a limited data size of 256 MB, which may not be sufficient for some IoT applications.

7.2 Centralized broker: MQTT requires a centralized broker, which can be a point of failure in large-scale IoT deployments.

7.3 Security risks: MQTT's security features may not be sufficient for highly sensitive IoT applications.

8. MQTT is a popular messaging protocol for IoT applications, offering several advantages such as efficiency, scalability, flexibility, and reliability. However, it also has some limitations, particularly in terms of limited data size, reliance on a centralized broker, and security risks.

9. Overall, MQTT is well suited for IoT deployments where lightweight messaging and reliable message delivery are critical, and where security and scalability are important considerations.

10. The Internet of Things (IoT) has revolutionized the way devices communicate with each other, and one of the key components of this communication is the messaging protocol. One such protocol that has gained popularity in recent years is the Message Queuing Telemetry Transport (MQTT).

11. Extra added features of MQTT:

MQTT has several features that make it well suited for IoT applications. These include:

11.1 Lightweight: MQTT is a lightweight protocol that can be used in low-bandwidth and high-latency environments.

11.2 Scalable: MQTT is highly scalable, allowing it to handle large volumes of messages and devices.

11.3 Asynchronous communication: MQTT supports asynchronous communication, allowing messages to be sent and received without the need for a synchronous request/response model.

11.4 Quality of service (QoS): MQTT supports three levels of QoS, allowing devices to control the level of message delivery reliability.

12. Advantages of MQTT in IoT:

MQTT has several advantages when used in IoT applications. These include:

12.1 Efficiency: MQTT's lightweight nature makes it efficient in terms of bandwidth and power consumption, making it well suited for low-power IoT devices.

12.2 Scalability: MQTT's scalability makes it well suited for large-scale IoT deployments.

12.3 Asynchronous communication: MQTT supports asynchronous communication, allowing devices to operate independently of each other.

12.4 QoS: MQTT's QoS levels allow devices to control the reliability of message delivery.

13. Limitations of MQTT in IoT:

While MQTT has several advantages, there are also some limitations to its use in IoT applications. These include:

13.1 Security: MQTT does not provide built-in security, requiring additional security measures to be implemented.

13.2 Compatibility: MQTT is not compatible with all IoT devices, requiring the use of a gateway or broker to translate between MQTT and other protocols.

13.3 Complexity: MQTT's publish/subscribe model can be complex to implement, particularly for small-scale deployments.

14. MQTT is a promising protocol for IoT applications, offering several advantages such as efficiency, scalability, and QoS. However, it also has some limitations, particularly in terms of security, compatibility, and complexity. Overall, MQTT is well suited for IoT deployments where low-power devices, scalability, and asynchronous communication are critical, and where additional security measures can be implemented to ensure secure messaging.

5.7 SUMMARY

In this chapter we learned the following topics

- HTTP is a versatile and widely used protocol in IoT applications, allowing for seamless communication between devices and web services or cloud platforms.
- MQTT supports three levels of QoS, allowing devices to control the level of message delivery reliability.
- AMQP is a promising protocol for IoT applications, offering several advantages such as interoperability, flexibility, scalability, and fault tolerance.

- XMPP is well suited for IoT deployments where low latency and real-time communication are critical, and where interoperability and security are important considerations.

5.8 QUESTIONS

Q1. Explain the HTTP Protocol in deail?

Q2. Explain why MQTT is better than HTTP.

Q3. Illustrate the concept of COAP in brief.

Q4. Explain the importance of XMPP in IOT session layer.

5.9 REFERENCES

1. https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot/index.html

❄❄❄❄❄❄❄

6

SERVICE LAYER PROTOCOL

Unit Structure :

6.1 OBJECTIVES

At the end of this unit, the student will be able to

- Understand the concept of OneM2m, ETSI.
- Create differentiate between M2M and OneM2m.
- Illustrate the concept of OMA and BBF.

6.2 ONEM2M

1. The Internet of Things (IoT) has enabled a wide range of connected devices, sensors, and platforms to communicate with each other, creating a network of interconnected devices. To achieve this interoperability and connectivity, a standardized framework is required.

2. The oneM2M standard is an initiative that aims to provide a common platform for IoT devices and services to communicate with each other. In this essay, we will discuss oneM2M in the context of IoT, its architecture, and its benefits.

3. oneM2M is a global standard for Machine to Machine (M2M) communications and the IoT. It was established in 2012, and it is based on the work of a number of existing standards organizations.

4. oneM2M provides a common framework for IoT devices, platforms, and services to communicate with each other. The main goal of oneM2M is to ensure interoperability between different IoT devices and services.

5. oneM2M Architecture:

The oneM2M architecture is based on a horizontal layering approach that consists of four layers, as shown below:

5.1 Application Layer: The application layer is responsible for managing the IoT applications and services that interact with the oneM2M platform.

5.2 Platform Services Layer: The platform services layer provides a set of common services that are used by IoT applications and services, such as security, device management, and data management.

5.3 Infrastructure Layer: The infrastructure layer provides the underlying network and computing infrastructure required to support IoT applications and services.

5.4 Device Layer: The device layer consists of the IoT devices and sensors that are connected to the oneM2M platform.

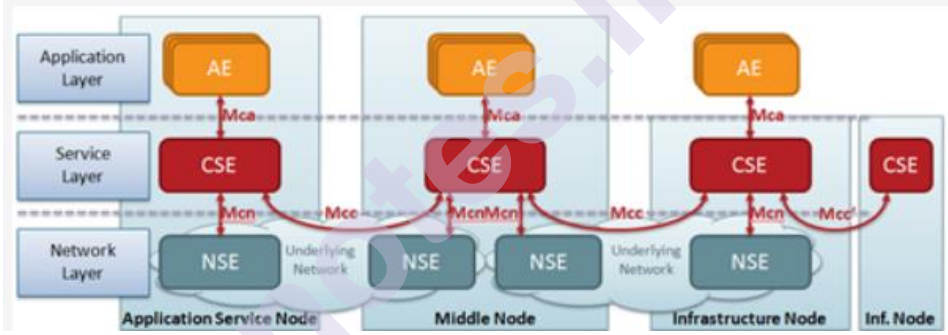


Fig 1 OneM2m Functional Architecture

6. Benefits of oneM2M in IoT:

oneM2M offers several benefits when used in IoT applications. These include:

6.1 Interoperability: oneM2M provides a common platform for IoT devices and services to communicate with each other, enabling interoperability between different IoT devices and platforms.

6.2 Scalability: oneM2M can support a large number of devices and services, making it well suited for large-scale IoT deployments.

6.3 Security: oneM2M provides a set of security mechanisms that can be used to secure IoT communications and data.

6.4 Flexibility: oneM2M is a flexible standard that can be adapted to different IoT use cases and requirements.

6.5 Global Reach: oneM2M is a global standard that can be used by IoT deployments worldwide, ensuring compatibility between different regions and countries.

6.6 oneM2M provides a common platform for IoT devices and services to communicate with each other, enabling interoperability between different IoT devices and platforms. The oneM2M architecture consists of four layers, providing a flexible and scalable framework for IoT deployments. The benefits of oneM2M include interoperability, scalability, security, flexibility, and global reach, making it a promising standard for the future of IoT.

6.7 OneM2M is a standard developed by a consortium of global organizations to provide a common platform for the interoperability of different IoT devices and services. It is a hierarchical architecture that enables seamless communication between different IoT devices and services, regardless of the underlying network technology.

7 OneM2M has several features that make it well suited for IoT applications. These include:

7.1 Interoperability: OneM2M provides a common platform for different IoT devices and services to communicate with each other, regardless of the underlying network technology.

7.2 Scalability: OneM2M is highly scalable, enabling it to support a large number of devices and services.

7.3 Security: OneM2M provides built-in security features, including authentication, authorization, and encryption.

7.4 Device management: OneM2M provides a standardized way of managing IoT devices, including registration, discovery, and configuration.

8 Advantages of OneM2M in IoT:

OneM2M has several advantages when used in IoT applications. These include:

8.1 Interoperability: OneM2M's common platform enables different IoT devices and services to communicate with each other, regardless of the underlying network technology.

8.2 Scalability: OneM2M's scalability makes it well suited for large-scale IoT deployments.

8.3 Security: OneM2M's built-in security features provide a secure platform for IoT communications.

8.4 Standardization: OneM2M provides a standardized platform that enables different IoT devices and services to communicate with each other, reducing the complexity of integrating different devices and services.

9 Limitations of OneM2M in IoT:

While OneM2M has several advantages, there are also some limitations to its use in IoT applications. These include:

9.1 Complexity: OneM2M's hierarchical architecture can be complex to implement and maintain.

9.2 Compatibility: OneM2M is not compatible with all IoT devices and services, requiring the use of gateways or translators to translate between OneM2M and other protocols.

9.3 Adoption: OneM2M is a relatively new standard, and its adoption may be limited in some industries.

10 In conclusion, OneM2M is a promising standard for IoT applications, offering several advantages such as interoperability, scalability, security, and standardization. However, it also has some limitations, particularly in terms of complexity, compatibility, and adoption. Overall, OneM2M is well suited for IoT deployments where interoperability, security, and scalability are critical, and where additional security measures can be implemented to ensure secure messaging.

6.3 ETSI

1. The Internet of Things (IoT) has gained significant attention in recent years as it has become increasingly popular to use interconnected devices to automate and optimize a variety of processes.

2. To support this, a range of standards and protocols have been developed to ensure interoperability and facilitate the development of IoT applications.

3. One of these standards is ETSI M2M. In this essay, we will discuss ETSI M2M in the context of IoT, outlining its features, advantages, and limitations.

4. ETSI M2M is a standard developed by the European Telecommunications Standards Institute (ETSI) to provide a common platform for the interoperability of different IoT devices and services. It is a set of specifications and guidelines that define the requirements for M2M (machine-to-machine) communication.

5. Features of ETSI M2M:

ETSI M2M has several features that make it suitable for IoT applications. These include:

5.1 Interoperability: ETSI M2M provides a common platform for different IoT devices and services to communicate with each other, regardless of the underlying network technology.

5.2 Scalability: ETSI M2M is highly scalable, enabling it to support a large number of devices and services.

5.3 Security: ETSI M2M provides built-in security features, including authentication, authorization, and encryption.

5.4 Device management: ETSI M2M provides a standardized way of managing IoT devices, including registration, discovery, and configuration.

6 Advantages of ETSI M2M in IoT:

ETSI M2M has several advantages when used in IoT applications. These include:

6.1 Interoperability: ETSI M2M's common platform enables different IoT devices and services to communicate with each other, regardless of the underlying network technology.

6.2 Scalability: ETSI M2M's scalability makes it well suited for large-scale IoT deployments.

6.3 Security: ETSI M2M's built-in security features provide a secure platform for IoT communications.

6.4 Standardization: ETSI M2M provides a standardized platform that enables different IoT devices and services to communicate with each other, reducing the complexity of integrating different devices and services.

7. Limitations of ETSI M2M in IoT:

7.1 While ETSI M2M has several advantages, there are also some limitations to its use in IoT applications. These include:

7.2 Complexity: ETSI M2M's specifications and guidelines can be complex to implement and maintain.

7.3 Compatibility: ETSI M2M is not compatible with all IoT devices and services, requiring the use of gateways or translators to translate between ETSI M2M and other protocols.

7.4 Adoption: ETSI M2M is a relatively new standard, and its adoption may be limited in some industries.

8 ETSI M2M is a promising standard for IoT applications, offering several advantages such as interoperability, scalability, security, and standardization. However, it also has some limitations, particularly in terms of complexity, compatibility, and adoption.

9 Overall, ETSI M2M is well suited for IoT deployments where interoperability, security, and scalability are critical, and where additional security measures can be implemented to ensure secure messaging.

10 The ETSI M2M architecture is designed to provide a standardized framework for machine-to-machine (M2M) communications in the Internet of Things (IoT). It is a layered architecture that defines the key components and interfaces necessary for M2M communications. The ETSI M2M architecture comprises the following layers:

10.1 Application Layer: This layer includes the M2M applications that use the ETSI M2M platform for communication. The applications communicate with the platform using standard protocols, such as HTTP or CoAP, and use ETSI M2M APIs to interact with other components of the architecture.

10.2 Service Layer: This layer provides services to M2M applications, such as device management, data management, and security. The services are exposed as RESTful APIs, and applications can use them to manage and interact with M2M devices.

10.3 Network Layer: This layer provides connectivity for M2M devices, including cellular networks, Wi-Fi, Ethernet, and others. The network layer also includes gateways that provide translation between different network technologies and protocols.

10.4 Device Layer: This layer includes M2M devices that connect to the network layer. The devices can be sensors, actuators, or other types of devices that collect or act on data. The devices are managed by the service layer and can be configured, monitored, and updated remotely.

11 The ETSI M2M architecture also includes several interfaces that enable communication between the layers. These interfaces include:

M2M Service Interface: This interface provides access to M2M services in the service layer.

11.1 Device Management Interface: This interface enables the service layer to manage M2M devices in the device layer.

11.2 Application Service Interface: This interface enables M2M applications to interact with the service layer.

11.3 Network Service Interface: This interface provides access to network services in the network layer.

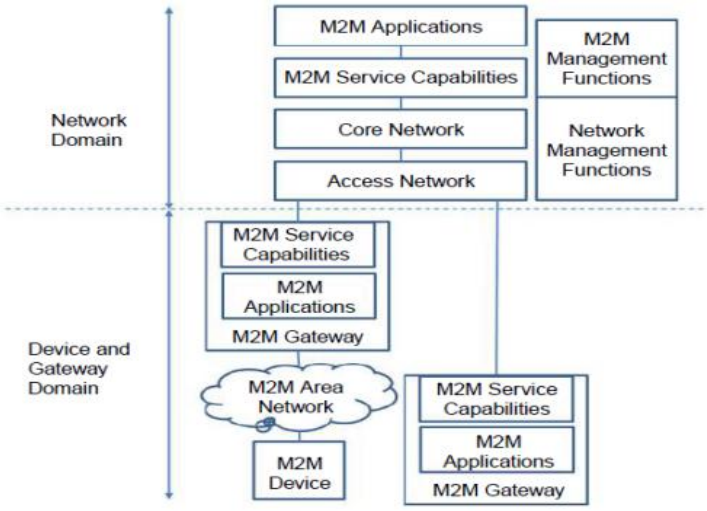


Fig 2 ETSI Architecture

12 The ETSI M2M architecture is designed to be modular and flexible, allowing different components to be added or removed as needed. It is also designed to be scalable and secure, with built-in security features such as authentication, authorization, and encryption.

13 Overall, the ETSI M2M architecture provides a comprehensive framework for M2M communications in the IoT, enabling interoperability, scalability, and security across different devices, networks, and applications.

7.4 OMA

1. OMA (Open Mobile Alliance) is a standards organization that develops specifications for mobile and IoT devices. In the context of IoT, OMA has developed several standards that define communication protocols, data models, and device management techniques. These standards are intended to enable interoperability between different IoT devices and platforms.

2. Some of the key standards developed by OMA for IoT include:

   2.1 Lightweight M2M (LwM2M): This is a device management protocol that enables remote management of IoT devices. It is designed to be lightweight and efficient, making it suitable for use in constrained environments. LwM2M defines a set of standard objects and interfaces for device management, and it uses CoAP as its underlying protocol.

   2.2 OMA DM (Device Management): This is a device management protocol that is used in mobile devices and IoT devices. It enables remote management of devices over various types of networks, including cellular, Wi-Fi, and Ethernet. OMA DM defines a set of standard objects and interfaces for device management, and it uses HTTP or CoAP as its underlying protocol.

   2.3 OMA Lightweight Machine-to-Machine (OMA LwM2M) Enabler: This is a set of specifications that define a standardized framework for IoT device management. It includes specifications for device management, data management, and security.

   2.4 OMA LwM2M Enabler is designed to be scalable and flexible, allowing it to support a wide range of IoT devices and applications.

   2.5 OMA SensorThings API: This is a standard API for IoT sensors and devices that collect data. It provides a standardized way to access and manage sensor data, making it easier to integrate data from different sources. SensorThings API is based on RESTful principles and uses JSON as its data format.

3. Overall, the OMA standards for IoT provide a comprehensive set of specifications that enable interoperability, security, and manageability in IoT devices and platforms. These standards are widely used in the industry and are supported by a large ecosystem of vendors and developers.

4. OMA (Open Mobile Alliance) architecture for IoT is designed to enable interoperability, security, and manageability in IoT devices and platforms. The OMA architecture is based on a client-server model, where the IoT devices act as clients and the servers manage the devices and provide services.

5. The OMA architecture consists of several layers, each with a specific function:

   5.1 Application layer: This layer contains the applications and services that run on the IoT devices. The application layer interacts with the services provided by the server layer.

   5.2 Device layer: This layer consists of the hardware and firmware that make up the IoT devices. The device layer interacts with the services provided by the server layer through the application layer.

   5.3 Server layer: This layer provides the services that manage and control the IoT devices. The server layer consists of several components, including the device management server, data management server, and security server.

   5.4 Gateway layer: This layer provides connectivity between the IoT devices and the server layer. The gateway layer may include devices such as routers, gateways, and access points.

6. The OMA architecture uses standardized protocols and data models to enable interoperability between different IoT devices and platforms. Some of the key protocols and data models used in the OMA architecture include:

   6.1 Lightweight M2M (LwM2M): This protocol is used for device management and enables remote management of IoT devices. LwM2M uses CoAP as its underlying protocol.

   6.2 OMA DM (Device Management): This protocol is used for device management and enables remote management of IoT devices. OMA DM uses HTTP or CoAP as its underlying protocol.

   6.3 OMA Lightweight Machine-to-Machine (OMA LwM2M) Enabler: This set of specifications defines a standardized framework for IoT device management. It includes specifications for device management, data management, and security.

6.4 OMA Sensor Things API: This API is used to access and manage sensor data from IoT devices.

7 Overall, the OMA architecture provides a comprehensive framework for building and managing IoT devices and platforms. It enables interoperability, security, and manageability, making it easier to develop and deploy IoT solutions.

7.5 BBF

1. BBF (Broadband Forum) is a global consortium of over 100 industry-leading companies that develop and promote broadband network technologies, including those related to the Internet of Things (IoT).

2. The BBF's work in the IoT space is focused on defining standards and best practices for connecting IoT devices to broadband networks, enabling the development of more reliable, scalable, and secure IoT solutions.

3. The BBF's IoT work is organized around several key areas:

   3.1 Device management: The BBF's work in this area is focused on developing standards and best practices for managing large numbers of IoT devices connected to broadband networks. This includes defining protocols for device discovery, configuration, firmware updates, and diagnostics.

   3.2 Security: The BBF recognizes the importance of security in IoT deployments and has developed a range of security standards and best practices for IoT devices and networks. These include guidelines for securing IoT devices, as well as protocols for secure communication between devices and the network.

   3.3 Data analytics: The BBF is working to develop standards and best practices for collecting, analyzing, and using data generated by IoT devices. This includes developing standards for data formats, protocols for data transfer and storage, and best practices for data analytics and machine learning.

   3.4 Interoperability: The BBF is committed to promoting interoperability between IoT devices and networks. This includes developing standards for device interoperability, as well as protocols for interoperable data exchange and communication.

   3.5 One of the BBF's key contributions to the IoT space is its Open Broadband – IoT (OB-IoT) project, which aims to define a standardized architecture for IoT deployments on broadband networks. The OB-IoT architecture consists of several layers, each with a specific function:

3.6 Device layer: This layer consists of the IoT devices themselves, as well as the software and firmware that enable them to connect to broadband networks. The BBF has developed standards and best practices for device management, security, and data analytics in this layer.

3.7 Gateway layer: This layer provides connectivity between IoT devices and the broadband network. Gateways may be dedicated devices or integrated into broadband modems or routers. The BBF has developed standards and best practices for gateway management, security, and interoperability.

3.8 Network layer: This layer includes the broadband network infrastructure, including switches, routers, and other network devices. The BBF has developed standards and best practices for network management, security, and interoperability.

3.9 Application layer: This layer includes the applications and services that run on top of the broadband network, including cloud-based services and analytics platforms. The BBF has developed standards and best practices for application development, security, and interoperability.

4 The OB-IoT project is supported by a range of BBF members, including network operators, equipment vendors, and service providers. The project aims to accelerate the development and deployment of IoT solutions on broadband networks, promoting interoperability, security, and scalability.

5 In addition to the OB-IoT project, the BBF is involved in a range of other initiatives related to IoT, including the development of standards and best practices for smart home networks, smart cities, and industrial IoT deployments.

6 Overall, the BBF's work in the IoT space is focused on defining standards and best practices that enable the development of more reliable, scalable, and secure IoT solutions. The OB-IoT project is a key part of this work, providing a standardized architecture for IoT deployments on broadband networks. With the support of its members, the BBF is well-positioned to drive innovation and promote interoperability in the IoT space.

7. Provide an overview of the BBF's work in IoT, focusing on the key standards and specifications developed by the consortium.

8. TR-069 is a specification developed by the BBF for the management of broadband networks. It provides a standard protocol for the remote management of network devices, including IoT devices.

9. The TR-069 specification enables the management of devices over a wide range of network types, including Ethernet, Wi-Fi, and cellular networks. The specification supports a range of management

functions, including configuration, software updates, and performance monitoring.

10. The TR-069 specification has been widely adopted by the industry and is used by many service providers to manage their IoT networks. It has also been adopted by the European Telecommunications Standards Institute (ETSI) as a standard for the management of IoT devices.

11. The User Services Platform (USP) is a specification developed by the BBF for the management of IoT devices. USP is designed to provide a standard, secure, and scalable framework for the management of IoT devices, including those that are deployed in homes and businesses.

12. The USP specification is based on the TR-069 protocol and provides a standardized framework for the management of IoT devices, regardless of the type of network they are connected to.

13. USP includes a range of management functions, including device discovery, configuration, and software updates. It also includes support for remote diagnostics, troubleshooting, and security management.

14. USP has been adopted by the industry and is being used by many service providers to manage their IoT networks. It is also being promoted by the BBF as a standard for the management of IoT devices.

15. G.hn is a standard developed by the BBF for the provision of high-speed broadband services over any wired home network, including powerline, coaxial, and telephone line networks.

16. G.hn provides a standard protocol for the transmission of data over these networks, enabling the delivery of high-speed broadband services to IoT devices.

17. The G.hn standard has been widely adopted by the industry and is being used by many service providers to deliver broadband services to homes and businesses.

18. It is also being used to provide connectivity for IoT devices, enabling the deployment of IoT solutions over existing home networks.

19. FAN (Fixed Access Network) is a specification developed by the BBF for the management of broadband access networks. FAN provides a standardized framework for the management of broadband access networks, including those that are used for the delivery of IoT services.

20. FAN includes a range of management functions, including network discovery, topology discovery, and service discovery. It also includes support for network security, device management, and service management.

21. FAN has been adopted by the industry and is being used by many service providers to manage their broadband access networks. It is also being used to manage IoT networks, providing a standardized framework for the management of IoT devices.

22. Open Broadband - Broadband Access Abstraction (OB-BAA)-OB-BAA is a specification developed by the BBF for the management of broadband access networks.

23. OB-BAA provides a standardized framework for the management of broadband access networks, including those that are used for the delivery of IoT services.

24. OB-BAA includes a range of management functions, including network discovery, topology discovery, and service discovery. It also includes support for network security, device management, and service management.

25. OB-BAA is being promoted by the BBF as a standard for the management of broadband access networks.

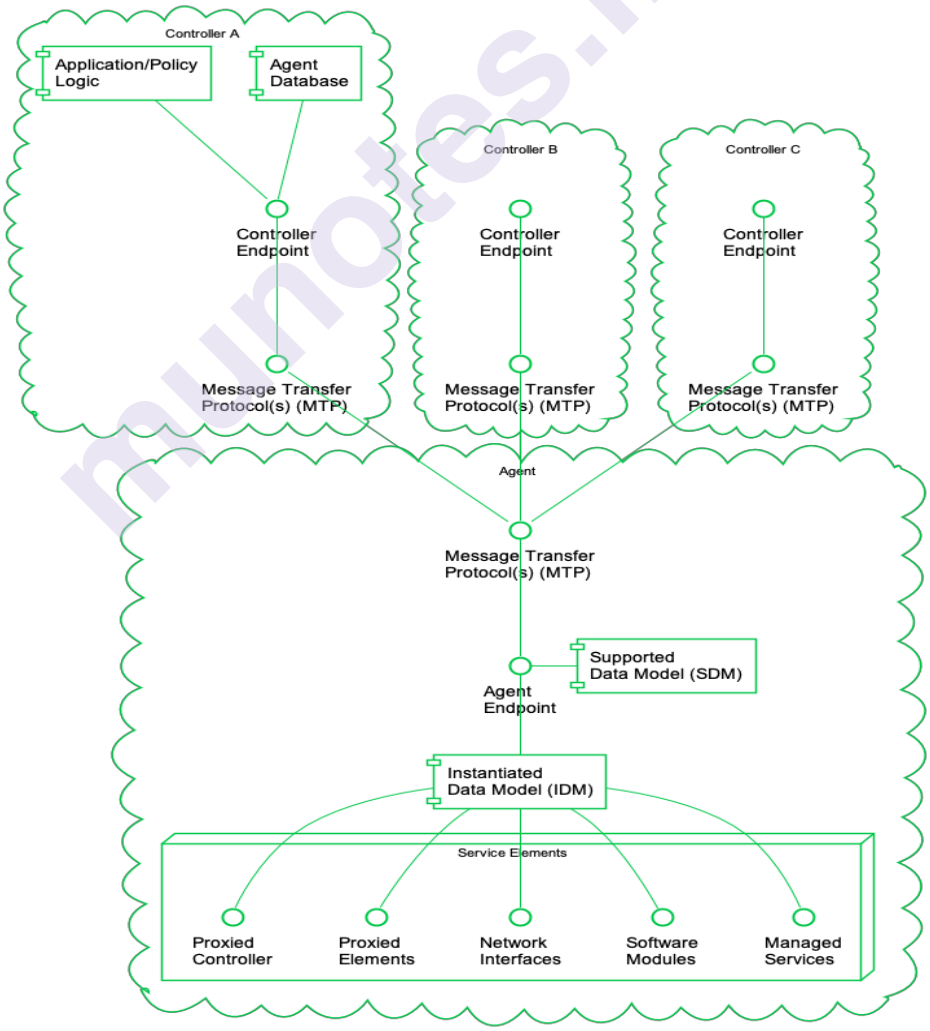


Fig 3 USP Agent and controller Architecture

6.6 SUMMARY

In this chapter we learned about various protocols under service layer

1. OneM2M is a promising standard for IoT applications, offering several advantages such as interoperability, scalability, security, and standardization.
2. The OMA architecture provides a comprehensive framework for building and managing IoT devices and platforms.
3. The ETSI M2M architecture is designed to be modular and flexible, allowing different components to be added or removed as needed.
4. OB-BAA is being promoted by the BBF as a standard for the management of broadband access networks

6.7 QUESTIONS

1. Explain the concept of OneM2M in detail?
2. What is the need for ETSI framework in IOT service layer?
3. Explain in detail the OMA framework?
4. Illustrate the concept of BBF in brief. Justify its protocols which are a part of BBF?

6.8 REFERENCES

1. https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot/index.html

❄❄❄❄❄❄❄