R20

III-CSE II-SEM

INTERNET OF THINGS
(20A05603T)

INTERNET OF THINGS

Syllabus

UNIT I : Introduction to IoT

Definition and Characteristics of IoT, physical design of IoT, IoT protocols, IoT communication models, IoT Communication APIs, Communication protocols, Embedded Systems, IoT Levels and Templates.

UNIT II : Prototyping IoT Objects using Microprocessor/Microcontroller

Working principles of sensors and actuators, setting up the board – Programming for IoT, Reading from Sensors, Communication: communication through Bluetooth, Wi-Fi.

UNIT III : IoT Architecture and Protocols

Architecture Reference Model- Introduction, Reference Model and architecture, IoT reference Model, Protocols- 6LowPAN, RPL, CoAP, MQTT, IoT frameworks- Thing Speak.

UNIT IV : Device Discovery and Cloud Services for IoT

Device discovery capabilities- Registering a device, Deregister a device, Introduction to Cloud Storage models and communication APIs Web-Server, Web server for IoT.

UNIT V : UAV IoT

Introduction toUnmanned Aerial Vehicles/Drones, Drone Types, Applications: Defense, Civil, Environmental Monitoring; UAV elements and sensors- Arms, motors, Electronic Speed Controller(ESC), GPS, IMU, Ultra sonic sensors; UAV Software –Arudpilot, Mission Planner, Internet of Drones(IoD)- Case study FlytBase.

Textbooks:

1. Vijay Madisetti and ArshdeepBahga, " Internet of Things (A Hands-on-Approach)", 1st Edition, VPT, 2014.

2. Handbook of unmanned aerial vehicles, K Valavanis; George J Vachtsevanos, New York, Springer, Boston, Massachusetts : Credo Reference, 2014. 2016.

Reference Books:

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

2. ArshdeepBahga, Vijay Madisetti - Internet of Things: A Hands-On Approach, Universities Press, 2014.

3.The Internet of Things, Enabling technologies and use cases – Pethuru Raj, Anupama C. Raman, CRC Press.

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

UNIT-1

Syllabus :

Introduction to IoT

Definition and Characteristics of IoT, physical design of IoT, IoT protocols, IoT communication models, IoT Communication APIs, Communication protocols, Embedded Systems, IoT Levels and Templates.

Introduction :

Internet of Things (IoT) comprises things that have unique identities and are connected to the Internet. While many existing devices, such as networked computers or 4G-enabled mobile phones. already have some form of unique identities and are also connected to the Internet, the focus on IoT is in the configuration, control and networking via the Internet of devices or "things" that are traditionally not associated with the Internet. These include devices such as thermostats, utility meters, a bluetooth-connected headset, irrigation pumps and sensors, or control circuits for an electric car's engine.

Internet of Things is a new revolution in the capabilities of the endpoints that am connected to the Internet, and is being driven by the advancements in capabilities (in combination with lower costs) in sensor networks, mobile devices, wireless communications, networking and cloud technologies. Experts forecast that by the year 2020 there will be a total of 50 billion devices/things connected to the Internet. Therefore, the major industry players are excited by the prospects of new markets for their products. The products include hardware and software components for IoT endpoints, hubs, or control centers of the IoT universe.

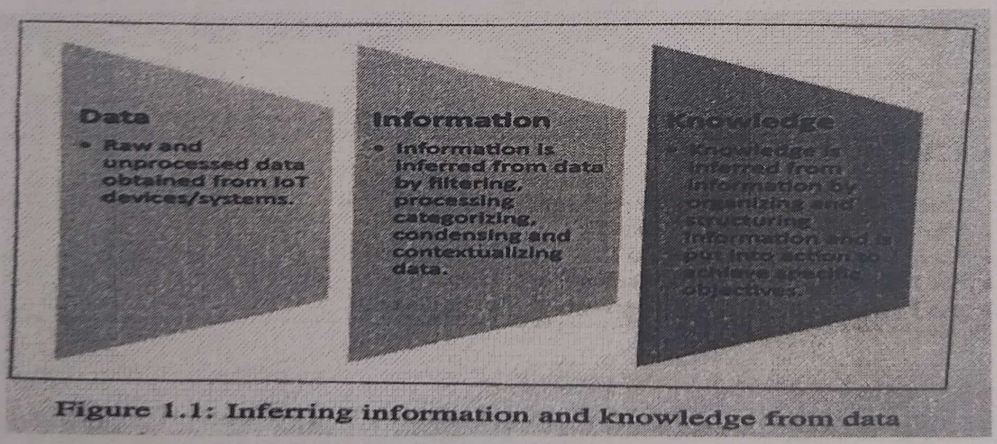


Figure 1.1: Inferring information and knowledge from data

The scope of IoT is not limited to just connecting things (devices, appliances, machines) to the Internet. IoT allows these things to communicate and exchange data (control & information, that could include data associated with users) while executing meaningful applications towards a common user or machine goal. Data itself does not have a meaning until it is contextualized processed into useful

information. Applications on IoT networks extract and create information from lower level data by filtering, processing, categorizing, condensing and contextualizing the data. This information obtained is then organized and structured to infer knowledge about the system and/or its users, its environment, and its operations and progress towards its objectives, allowing a smarter performance, as shown in Figure 1.1. For example, consider a series of raw sensor measurements ((72,45) ; (84, 56)) generated by a weather monitoring station, which by themselves do not have any meaning or context.

To give meaning to the data, a context is added, which in this example can be that each tuple in data represents the temperature and humidity measured every minute. With this context added we know the meaning (or information) of the measured data tuples. Further information is obtained by categorizing, condensing or processing this data. For example, the average temperature and humidity readings for last five minutes is obtained by averaging the last five data tuples. The next step is to organize the information and understand the relationships between pieces of information to infer knowledge which can be put into action. For example, an alert is raised if the average temperature in last five minutes exceeds 120F, and this alert may be conditioned on the user's geographical position as well.

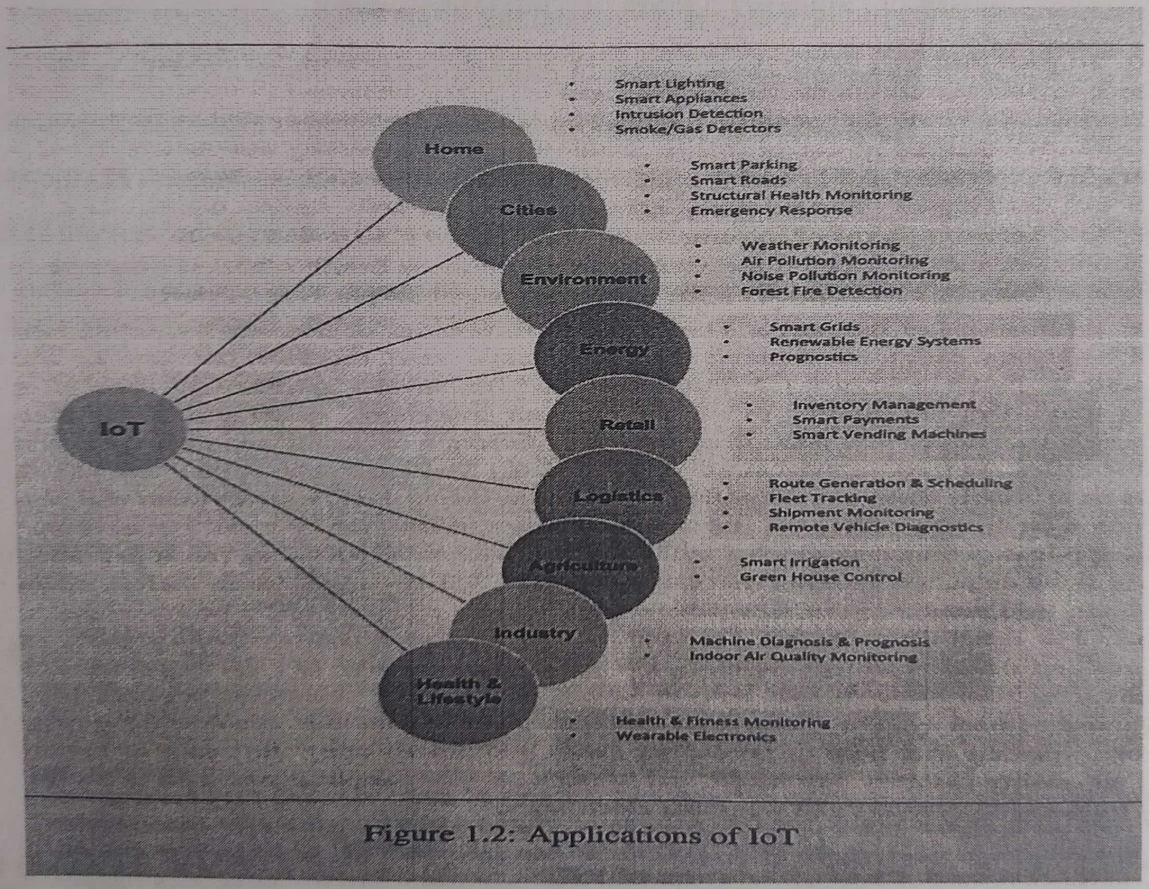


Figure 1.2: Applications of IoT

The applications of Internet of Things span a wide range of domains including (but not limited to) smart cities, environment, energy systems, retail, logistics, industry, agriculture and health as listed in

Figure 1.2. For homes, IoT has several applications such as smart lighting that adapt the lighting to suit the ambient conditions, smart appliances that can be remotely monitored and controlled, intrusion detection systems, smart smoke detectors, etc. For cities, IoT has applications such as smart parking systems that provide status updates on available slots, smart lighting that helps in saving energy, smart roads that provide information on driving conditions and structural health monitoring systems. For environment, IoT has applications such as weather monitoring, air and noise pollution, forest fire detection and river flood detection systems. For energy systems, IoT has applications such as including smart grids, grid integration of renewable energy sources and prognostic health management systems.

For retail domain, IoT has applications such as inventory management, smart payments and smart vending machines. For agriculture domain, IoT has applications such as smart irrigation systems that help in saving water while enhancing productivity and green house control systems. Industrial applications of IoT include machine diagnosis and prognosis systems that help in predicting faults and determining the cause of faults and indoor air quality systems. For health and lifestyle, IoT has applications such as health and fitness monitoring systems and wearable electronics.

1.1 Definition & Characteristics of IoT :

The Internet of Things (IoT) has been defined as:

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

Let us examine this definition of IoT further to put some of the terms into perspective.

- **Dynamic & Self-Adapting:** IoT devices and systems may have the capability to dynamically adapt with the changing contexts and take actions based on their operating conditions, user's context, or sensed environment. For example, consider a surveillance system comprising of a number of surveillance cameras. The surveillance cameras can adapt their modes (to normal or infra-red night modes) based on whether it is day or night. Cameras could switch from lower resolution to higher resolution modes when any motion is detected and alert nearby cameras to do the same. In this example, the surveillance system is adapting itself based on the context and changing (e.g., dynamic) conditions.

- **Self-Configuring:** IoT devices may have self-configuring capability, allowing a large number of devices to work together to provide certain functionality (such as weather monitoring). These devices

have the ability configure themselves (in association with the IoT infrastructure), setup the networking, and fetch latest software upgrades with minimal manual or user intervention.

- **Interoperable Communication Protocols:** IoT devices may support a number of interoperable communication protocols and can communicate with other devices and also with the infrastructure. We describe some of the commonly used communication protocols and models in later sections.

- **Unique Identity:** Each IoT device has a unique identity and a unique identifier (such as an IP address or a URI). IoT systems may have intelligent interfaces which adapt based on the context, allow communicating with users and the environmental contexts. IoT device interfaces allow users to query the devices, monitor their status, and control them remotely, in association with the control, configuration and management infrastructure.

- **Integrated into Information Network:** IoT devices are usually integrated into the information network that allows them to communicate and exchange data with other devices and systems. IoT devices can be dynamically discovered in the network, by other devices and/or the network, and have the capability to describe themselves (and their characteristics) to other devices or user applications. For example, a weather monitoring node can describe its monitoring capabilities to another connected node so that they can communicate and exchange data. Integration into the information network helps in making IoT systems "smarter" due to the collective intelligence of the individual devices in collaboration with the infrastructure. Thus, the data from a large number of connected weather monitoring IoT nodes can be aggregated and analyzed to predict the weather.

1.2 Physical Design of IoT :

1.2.1 Things in IoT

The "Things" in IoT usually refers to IoT devices which have unique identities and can perform remote sensing, actuating and monitoring capabilities. IoT devices can exchange data with other connected devices and applications (directly or indirectly), or collect data from other devices and process the data either locally or send the data to centralized servers or cloud-based application back-ends for processing the data, or perform some tasks locally and other tacks within the IoT infrastructure, based on temporal and space constraints (i.e., memory, processing capabilities. communication latencies and speeds. and deadlines).

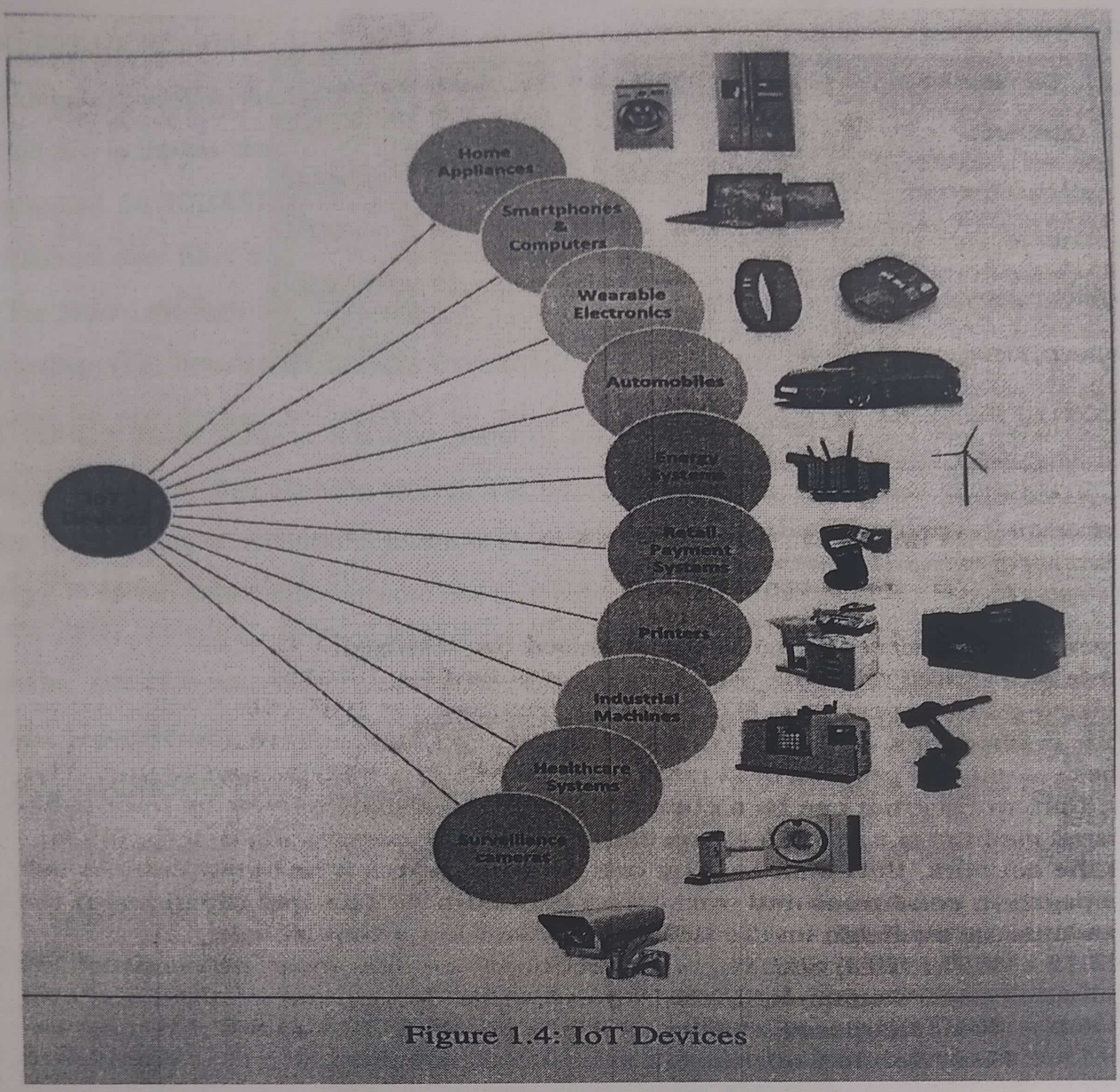
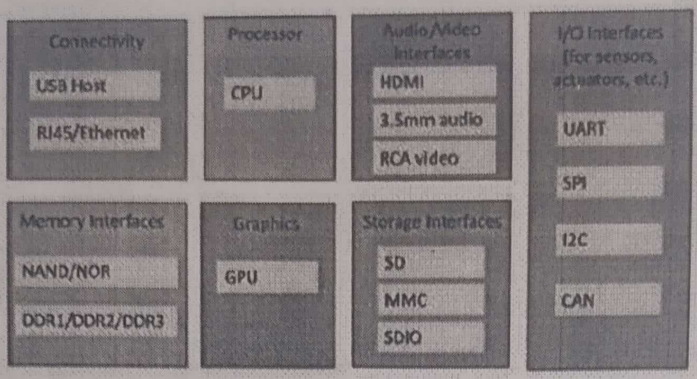


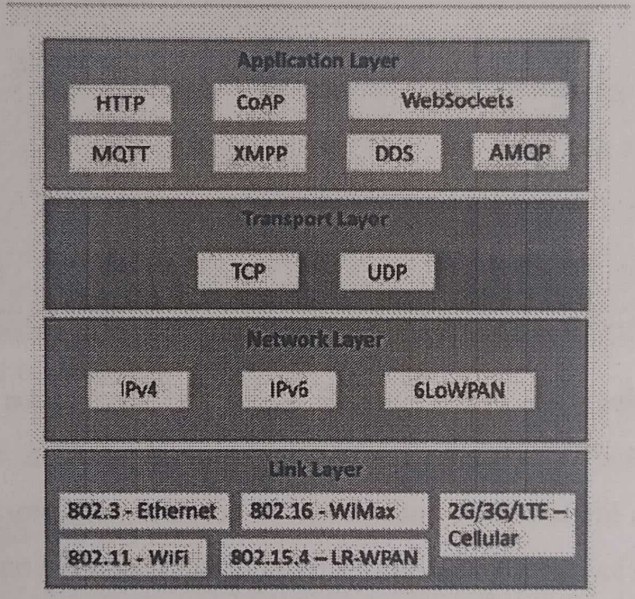
Figure 1.4: IoT Devices

Figure 1.3 shows a block diagram of a typical IoT device. An IoT device may consist of several interfaces for connections to other devices, both wired and wireless. These include (i) I/O interfaces for sensors. (ii) interfaces for Internet connectivity. (iii) memory and storage interfaces and (iv) audio/video interfaces. An IoT device can collect various types of data from the on-board or attached sensors, such as temperature, humidity, light intensity. The sensed data can be communicated either to other devices or cloud-based servers/storage. IoT devices can be connected to actuators that allow them to interact with other physical entities (including non-IoT devices and systems) in the vicinity of the device. For example, a relay switch connected to an IoT device can turn an appliance on/off based on the commands sent to the IoT device over the Internet.



IoT devices can also be of varied types, for instance, wearable sensors, smart watches, LED lights, automobiles and industrial machines. Almost all IoT devices generate data in some form or the other which when processed by data analytics systems leads to useful information to guide further actions locally or remotely. For instance, sensor data generated by a soil moisture monitoring device in a garden, when processed can help in determining the optimum watering schedules.

1.3. IoT Protocols :



→ Link Layer

Link layer protocols determine how the data is physically sent over the network's physical layer or medium (e.g., copper wire, coaxial cable, or a radio wave). The scope of the link layer is the local network connection to which host is attached. Hosts on thc same link exchange data packets over the link layer using link layer protocols. Link layer determines how the packets are coded and signaled by the hardware device over the medium to which the host is attached (such as a coaxial cable). Let us now look at some link layer protocols which are relevant in the context of IoT.

- **802.3 - Ethernet :** IEEE 802.3 is a collection of wired Ethernet standards for the link layer. For example, 802.3 is the standard for 10BASE5 Ethernet that uses coaxial cable as a shared medium, 802.3.1 is the standard for I OBASE-T Ethernet over copper twisted-pair connections, 802.3.j is the standard for IOBASE-F Ethernet over fiber optic connections, 802.3ae is the standard for 10 Gbit/s Ethernet over fiber, and so on. These standards provide data rates from 10 Mb/s to 40 Gb/s and higher. The shared medium in Ethernet can be a coaxial cable. twisted-pair wire or an optical fiber. The shared medium (i.e., broadcast medium) carries the communication for all the devices on the network, thus data sent by one device can received by all devices subject to propagation conditions and transceiver capabilities.

- **802.11 - WiFi :** IEEE 802.11 is a collection of wireless local area network (WLAN) communication standards, including extensive description of the link layer. For example, 802.11a operates in the 5 GHz band, 802.11 b and 802.11g operate in the 2.4 Gliz band, 802.1In operates in the 2.4/5 GHz bands, 802.11ac operates in the 5 GHz band and 802.11 ad operates in the 60 GHz band. These standards provide data rates from I Mb/s to upto 6.75 Gb/s.

- **802.16 - WiMax :** IEEE 802.16 is a collection of wireless broadband standards, including extensive descriptions for the link layer (also called WiMax). WiMax standards provide data rates from 1.5 Mb/s to I Gb/s. The recent update (802.16m) provides data rates of 100 Mhii/s for mobile stations and I Gbit/s for fixed stations.

- **802.15.4 - LR-WPAN :** IEEE 802.15.4 is a collection of standards for low-rate wireless personal area networks (LR-WPANs). These standards form the basis of specifications for high level communication protocols such as ZigBee. LR-WPAN standards provide data rates from 40 Kb/s 250 Kb/s. These standards provide low-cost and low-speed communication for power constrained devices.

- **2G/ 3G/ 4G - Mobile Communication:** There are different generations of mobile communication standards including second generation (2G including GSM and CDMA), third generation (36 - including UMTS and CDMA2000) and fourth generation (4G -including crE). IoT devices based on these standards can communicate over cellular networks. Data rates for these standards range from 9.6 Kb/s (for 2G) to upto 100 Mb/s (for 40) and are available from the 3GPP websites.

→ **Network/Internet Layer**

The network layers are responsible for sending of IP datagram from the source network to the destination network. This layer performs the host addressing and packet routing. The datagrams contain the source and destination addresses which are used to route diem from the source to destination across multiple networks. Host identification is done using hierarchical II' addressing schemes such as IPv4 or IPv6.

- **IPv4 :** Internet Protocol version 4 (IPv4) is the most deployed Internet protocol that is used to identify the devices on a network using a hierarchical addressing scheme. IPv4 uses a 32-bit address scheme that allows total of 232 or 4.294,967.296 addresses. As more and more devices got connected to the Internet, these addresses got exhausted in the year 2011. 1Pv4 has been succeeded by IPv6. The IP protocols establish connections on packet networks, but do not guarantee delivery of packets. Guaranteed delivery and data integrity are handled by the upper layer protocols (such as TCP).

- **IPv6 :** Internet Protocol version 6 (IPv6) is the newest version of Internet protocol and successor to 1Pv4. IPv6 uses 128-bit address scheme that allows total of 2128 or 3.4 x 1038 addresses.

- **6LoWPAN :** 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) brings IP protocol to the low-power devices which have limited processing capability.

6LoWPAN operates in the 2.4 (iHz frequency range and provides data transfer rates of 250 Kb/s. 6LoWPAN works with the 802.15.4 link layer protocol and defines compression mechanisms for IPv6 datagrams over IEEE 802.15.4-based networks 181.

→**Transport Layer**

The transport layer protocols provide end-to-end message transfer capability independent of the underlying network. The message transfer capability can be set up on connections, either using handshakes (as in TCP) or without handshakes/acknowledgements (as in UDP). The transport layer provides functions such as error control, segmentation. flow control and congestion control.

- **TCP :** Transmission Control Protocol (TCP) is the most widely used transpon layer protocol, that is used by web browsers (along with Hfll Firms application layer protocols). email programs (SMTP application layer protocol) and file transfer (FTP). TCP is a connection oriented and stateful protocol. While 1P protocol deals with sending packets. TO' ensures reliable transmission of packets in-order. TO' also provides error detection capability so that duplicate packets can be discarded and lost packets are

retransmitted. The flow control capability of TCP ensures that rate at which the sender sends the data is not too high for the receiver to process. The congestion control capability of TCP helps in avoiding network congestion and congestion collapse which can lead to degradation of network performance.

• **UDP :** Unlike TCP, which requires carrying out an initial setup procedure, UDP is a connectionless protocol. UDP is useful for dine-sensitive applications that have very small data units to exchange and do not want the overhead of connection setup. UDP is a transaction oriented and stateless protocol. UDP does not provide guaranteed delivery, ordering of messages and duplicate elimination. Higher levels of protocols can ensure reliable delivery or ensuring connections created arc reliable.

→ **Application Layer**

Application layer protocols define how the applications interface with the lower layer protocols to send the data over the network. The application data, typically in files, is encoded by the application layer protocol and encapsulated in the transport layer protocol which provides connection or transaction oriented communication over the network. Port numbers arc used for application addressing (for example port 80 for HTTP, port 22 for SSH, etc.). Application layer protocols enable process-to-process connections using pons.

- **HTTP :** Hypertext Transfer Protocol (HTTP) is the application layer prottxol that forms the foundation of the World Widc Web (WWW). Hire includes commands such as GET, PUT, POST, DELETE, HEAD, TRACE, OPTIONS, etc. The protocol follows a request-response model where a client sends requests to a server using the HTTP commands. HTTP is a stateless protocol and each HTTP request is independent of the other requests. An HTTP client can be a browser or an application running on the client (e.g., an application running on an IoT device, a mobile application or other software). HTTP protocol uses Universal Resource Identifiers (URIs) to identify HTTP resources.

- **CoAP :** Constrained Application Protocol (CoAP) is an application layer protocol for machine-to-machine (M2M) applications, meant for constrained environments with constrained devices and constrained networks. Like HTTP. CoAP is a web transfer protocol and uses a request-response model, however it runs on top of UDP instead of TCP. CoAP uses a client-server architecture where clients communicate with servers using connectionless datagrams. CoAP is designed to easily interface with HTTP. Like HTTP, CoAP supports methods such as GET, PUT, POST, and DELETE.

- **WebSocket :** WebSocket protocol allows full-duplex communication over a single socket connection for sending messages between client and server. WebSocket is based on TCP and allows streams of messages to be sent back and forth between the client and server while keeping the TCP connection open. The client can be a browser, a mobile application or an IoT device.

- **MQTT :** Message Queue Telemetry Transport (MQTT) is a light-weight messaging protocol based on the publish-subscribe model. MQTT uses a client-server architecture where the client (such as an IoT device) connects to the server (also called MQTT Broker) and publishes messages to topics on the server. The broker forwards the messages to the clients subscribed to topics. MQTT is well suited for constrained environments where the devices have limited processing and memory resources and the network bandwidth is low.

- **XMPP :** Extensible Messaging and Presence Protocol (XMPP) is a protocol for real-time communication and streaming XML data between network entities. XMPP powers wide range of applications including messaging. presence, data syndication, gaming, multi-party chat and voice/video calls. XMPP allows sending small chunks of XML data from one network entity to another in near real-time. XMPP is a decentralized protocol and uses a client-server architecture. XMPP supports both client-to-server and server-to-server communication paths. In the context of IoT, XMPP allows real-time communication between IoT devices.

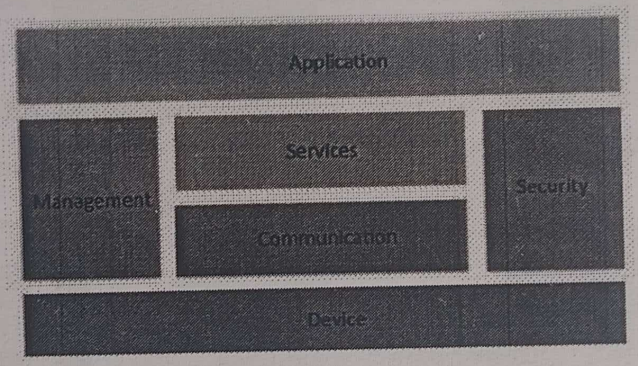
- **XMPP :** Extensible Messaging and Presence Protocol (XMPP) is a protocol for real-time communication and streaming XML data between network entities. XMPP powers wide range of applications including messaging. presence, data syndication, gaming, multi-party chat and voice/video calls. XMPP allows sending small chunks of XML data from one network entity to another in near real-time. XMPP is a decentralized protocol and uses a client-server architecture. XMPP supports both client-to-server and server-to-server communication paths. In the context of IoT, XMPP allows real-time communication between IoT devices.

- **DDS :** Data Distribution Service (DDS) is a data-centric middleware standard for device-to-device or machine-to-machine communication. DDS uses a publish-subscribe model where publishers (e.g. devices that generate data) create topics to which subscribers (e.g., devices that want to consume data) can subscribe. Publisher is an object responsible for data distribution and the subscriber is responsible for itceiving published data DDS provides quality-of-service (QoS) control and configurable reliability.

- **AMQP :** Advanced Message Queuing Protocol (AMQP) is an open application layer protocol for business messaging. AMQP supports both point-to-point and publisher/subscriber models, routing and queuing. AMQP brokers receive messages from publishers (e.g., devices or applications that generate data) and route them over connections to consumers (applications that process data). Publishers publish the messages to exchanges which then distribute message copies to queues. Messages are either delivered by the broker to the consumers which have subscribed to the queues or the consumers can pull the messages from the queues.

Logical Design of IoT

Logical design of an IoT system refers to an abstract representation of the entities and processes without going into the low-level specifics of the implementation. In this section we describe the functional blocks of an IoT system and the communication APIs



IoT Functional Blocks

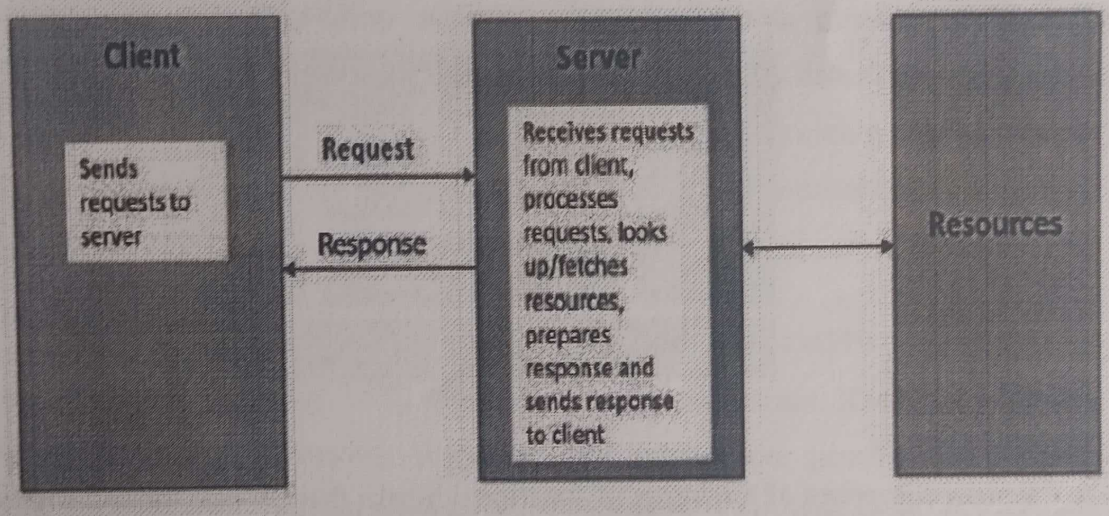
An IoT system comprises of a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, communication, and management as shown in Figure 1.6. These functional blocks are described as follows:

- **Device :** An IoT system comprises of devices that provide sensing, actuation, monitoring and control functions.
- **Communication:** The communication block handles the communication for the IoT system. You learned about various protocols used for communication by IoT systems in section 1.2.
- **Services:** An IoT system uses various types of IoT services such as services for device monitoring, device control services, data publishing services and services for device discovery.
- **Management:** Management functional block provides various functions to govern the IoT system.

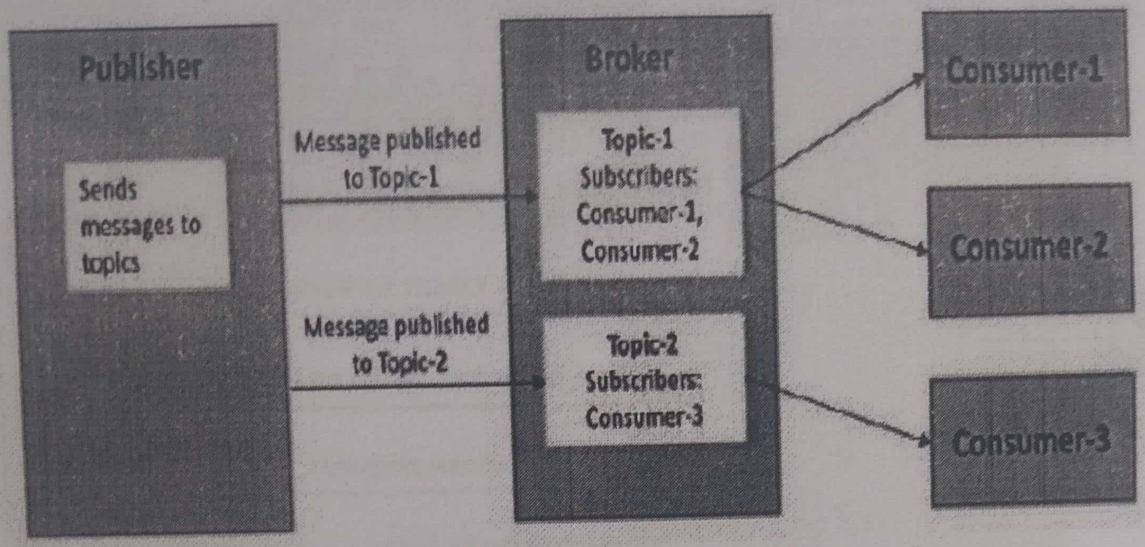
- **Security:** Security functional block secures the IoT system and by providing functions such as authentication. authorization, message and content integrity, and data security.
- **Application:** IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view or analyze the processed data.

1.4. IoT Communication Models :

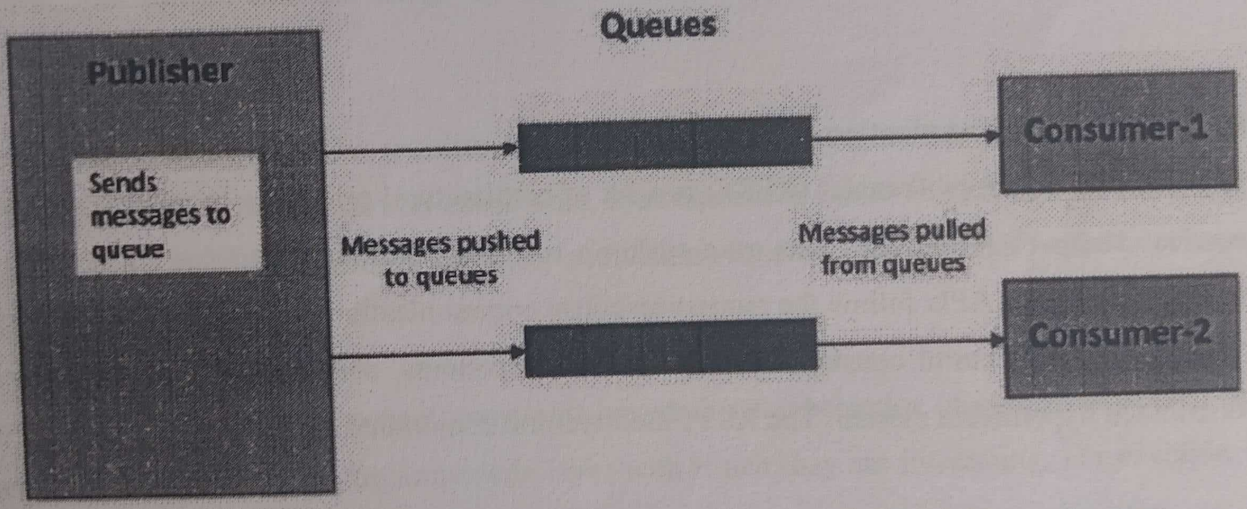
- **Request-Response:** Request-Response is a communication model in which the client sends requests to the server and the server responds to the requests. When the server receives a request, it decides how to respond, fetches the data, retrieves resource representations, prepares the response, and then sends the response to the client. Request-Response model is a stateless communication model and each request-response pair is independent of others.



- **Publish-Subscribe:** Publish-Subscribe is a communication model that involves publishers, brokers and consumers. Publishers are the source of data. Publishers send the data to the topics which are managed by the broker. Publishers are not aware of the consumers. Consumers subscribe to the topics which are managed by the broker. When the broker receives data for a topic from the publisher. it sends the data to all the subscribed consumers. Figure 1.8 shows the publisher-broker-consumer interactions in the publish-subscribe model.

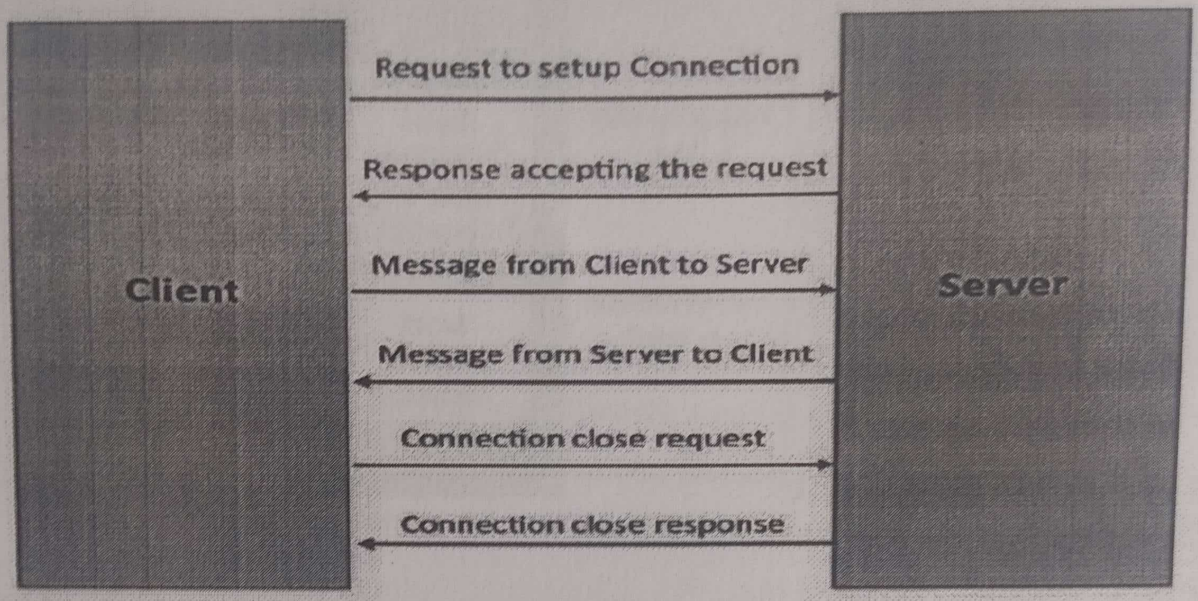


- **Push-Pull:** Push-Pull is a communication model in which the data producers push the data to queues and the consumers pull the data from the queues. Producers do not need to be aware of the consumers. Queues help in decoupling the messaging between the producers and consumers. Queues also act as a buffer which helps in situations when there is a mismatch between the rate at which the producers push data and the rate at which the consumers pull data. Figure IS shows the publisher-queue-consumer interactions in the push-pull model.



- **Exclusive Pair:** Exclusive Pair is a bi-directional, fully duplex communication model that uses a persistent connection between the client and server. Once the connection is setup it remains open until the client sends a request to close the connection. Client and server can send messages to each other after

connection setup. Exclusive pair is a stateful communication model and the server is aware of all the open connections.

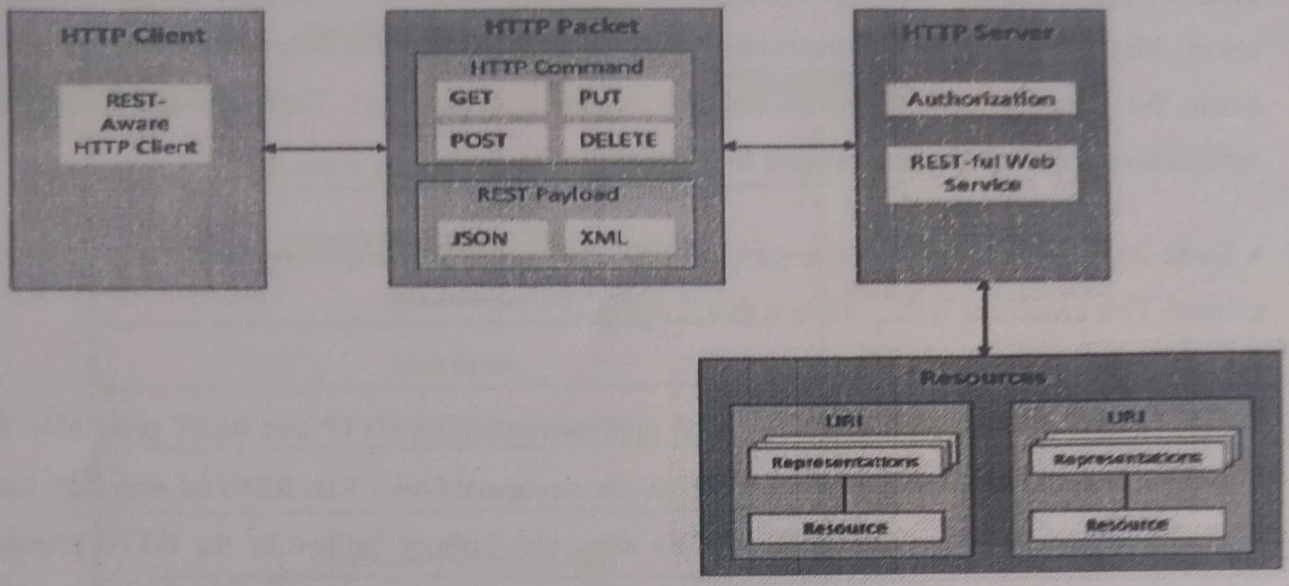


1.5. IoT Communication APIs :

In the previous section you learned about various communication models. In this section you will learn about two specific communication APIs.

REST-based Communication APIs

Representational State Transfer (REST) is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred. REST APIs follow the request-response communication model described in previous section. The REST architectural constraints apply to the components, connectors, and data elements, within a distributed hypermedia system. The REST architectural constraints are as follows:



- **Client-Server:** The principle behind the client-server constraint is the separation of concerns. For example, clients should not be concerned with the storage of data which is a concern of the server. Similarly, the server should not be concerned about the user interface, which is a concern of the client. Separation allows client and server to be independently developed and updated.

- **Stateless:** Each request from client to server must contain all the information necessary to understand the request, and cannot take advantage of any stored context on the server. The session state is kept entirely on the client.

- **Cache-able:** Cache constraint requires that the data within a response to a request be implicitly or explicitly labeled as cache-able or non-cache-able. If a response is cache-able, then a client cache is given the right to reuse that response data for later, equivalent requests. Caching can partially or completely eliminate some interactions.

- **Layered System:** Layered system constraint, constrains the behavior of components such that each component cannot see beyond the immediate layer with which they are interacting. For example, a client cannot tell whether it is connected directly to the end server, or to an intermediary along the way. System scalability can be improved by allowing intermediaries to respond to requests instead of the end server, without the client having to do anything different.

- **Uniform Interface:** Uniform Interface constraint requires that the method of communication between a client and a server must be uniform. Resources are identified in the requests (by URIs in web based

systems) and are themselves separate from the representations of the resources that are returned to the client. When a client holds a representation of a resource it has all the information required to update or delete the resource (provided the client has required permissions). Each message includes enough information to describe how to process the message.

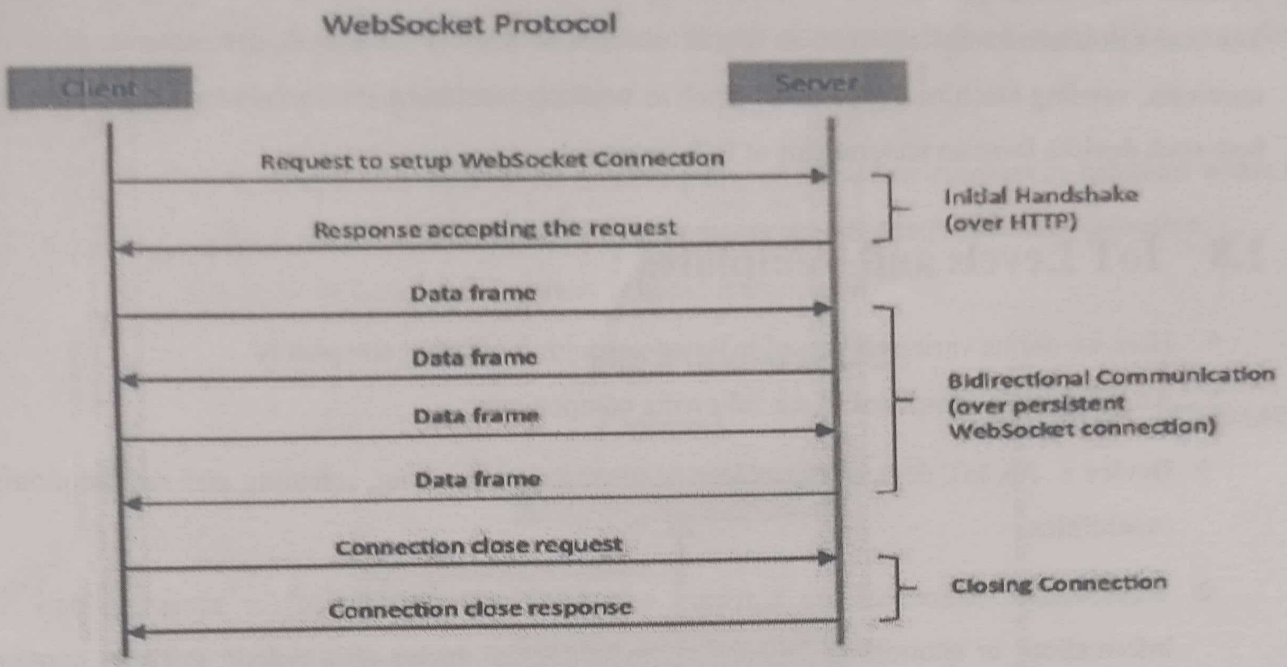
- **Code on demand:** Servers can provide executable code or scripts for clients to execute in their context. This constraint is the only one that is optional.

A **RESTful web service** is a "web API" implemented using HTTP and REST principles. RESTful web service is a collection of resources which are represented by URIs. RESTful web API has a base URI. The clients send requests to these URIs using the methods defined by the HTTP protocol (e.g., GET, PUT, POST, or DELETE). A RESTful web service can support various Internet media types (JSON being the most popular media type for RESTful web services).

WebSocket-based Communication APIs

WebSocket APIs allow bi-directional, full duplex communication between clients and servers. WebSocket APIs follow the exclusive pair communication model. Unlike request-response APIs such as REST, the WebSocket APIs allow full duplex communication and do not require a new connection to be setup for each message to be sent. WebSocket communication begins with a connection setup request sent by the client to the server.

This request (called a WebSocket handshake) is sent over HTTP and the server interprets it as an upgrade request. If the server supports WebSocket protocol, the server responds to the WebSocket handshake response. After the connection is setup, the client and server can send data/messages to each other in full-duplex mode. WebSocket APIs reduces the network traffic and latency as there is no overhead for connection setup and termination requests for each message. WebSocket is suitable for IoT applications that have low latency or high throughput requirements.



1.6 Communication Protocols :

Communication protocols form the backbone of IoT systems and enable network connectivity and coupling to applications. Communication protocols allow devices to exchange data over the network. In section 1.2.2 you learned about various link, network, transport and application layer protocols. These protocols define the data exchange formats, data encoding, addressing schemes for devices and routing of packets from source to destination. Other functions of the protocols include sequence control (that helps in ordering packets determining lost packets), flow control (that helps in controlling the rate at which the sender is sending the data so that the receiver or the network is not overwhelmed) and retransmission of lost packets.

1.7 Embedded Systems :

An Embedded System is a computer system that has computer hardware and software embedded to perform specific tasks. In contrast to general purpose computers or personal computers (ICs) this can perform various types of tasks. Embedded systems are designed to perform a specific set of tasks. Key components of an embedded system include microprocessor or microcontroller, memory (RAM, ROM, and cache), networking units (Ethernet, Wifi adapters), input/output units (display, keyboard, etc.) and storage (such as flash memory). Some embedded systems have specialized processors such as digital signal processors (DSPs), graphics processors and application specific processors. Embedded systems run

embedded operating systems such as real-time operating systems (RTOS). Embedded systems range from low-cost miniaturized devices such as digital watches to devices such as digital cameras, point of sale terminals, vending machines, appliances (such as washing machines) etc. In the next chapter we describe how such devices form an integral part of IoT systems.

1.8 IoT Levels and Templates :

- Here we define various levels of IoT systems with increasing completely.
- An IoT system comprises of the following components :

→ **Device :** An IoT device allow identification, remote sensing, actuating and remote monitoring capabilities.

→ **Resources :** Resources are software components on the device for accessing and storing information or controlling actuator connected to the device also include software components that enable network access for the device .

→ **Controller Service :** Controller Service is a native service that runs on the device and interact with the web services. Controller service sends data from the device to the web service and receive command from the application for controlling the device.

→ **Database:** Database can be either local or in the cloud and stores the data generated by the IoT device.

→ **Web Service:** Web services serve as a link between the IoT device, application, database and analysis components. Web service can be either implemented using HTTP and REST principles (REST service) or using WebSocket protocol (WebSocket service).

→ **Analysis Component**: The Analysis Component is responsible for analyzing the IoT data and generate results in a form which are easy for the user to understand. Analysis of IoT data can be performed either locally or in the cloud. Analyzed results are stored in the local or cloud databases.

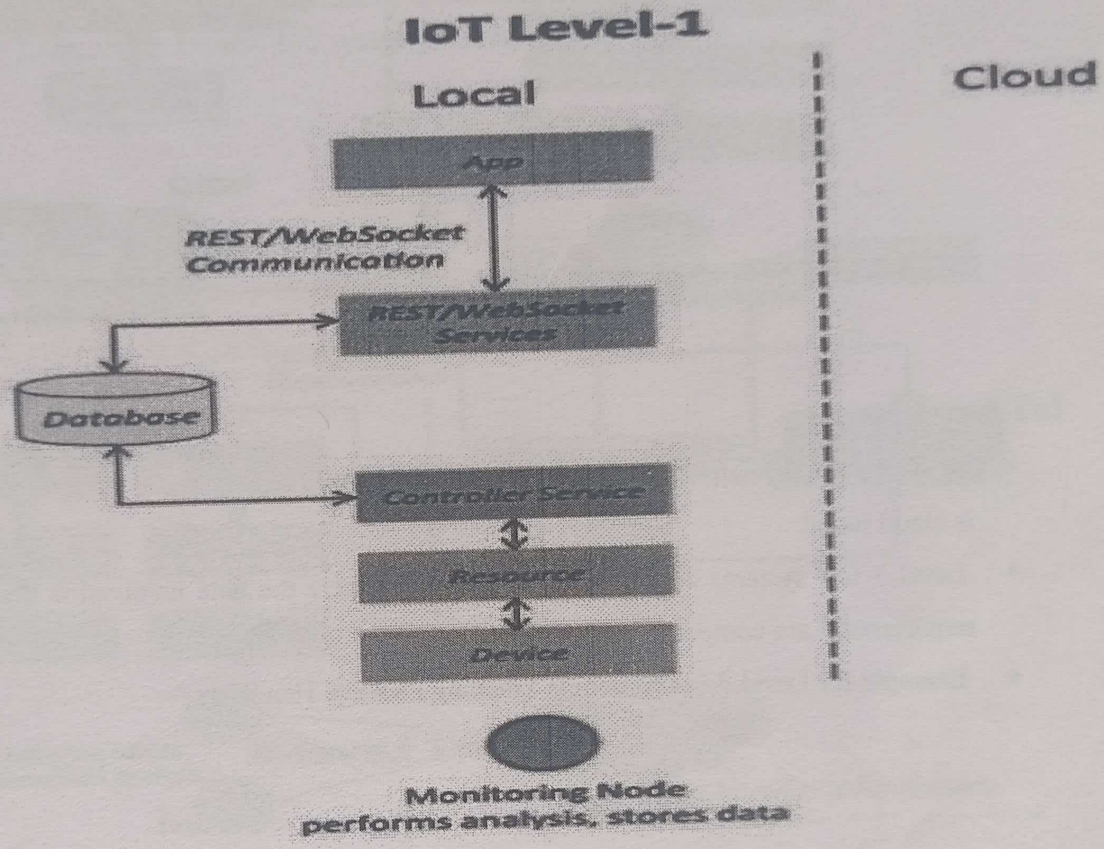
→ **Application:** IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view the processed data.

IoT level 1:

- A level-1 IoT system has a single node/device that performs sensing and/or actuation, stores data, performs analysis and hosts the application

• Level-1 IoT systems are suitable for modeling low-cost and low-complexity solutions where the data involved is not big and the analysis requirements are not computationally intensive.

Example for Level-1 IoT system is home Automation.



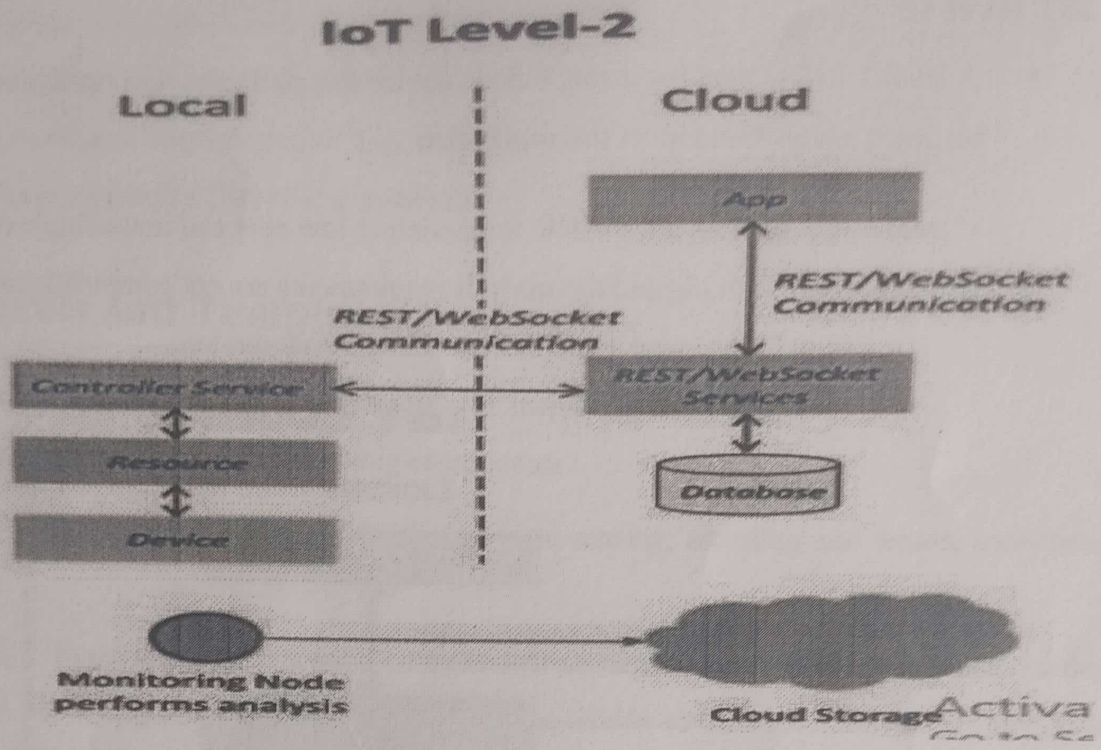
IoT Level-2 :

- A level-2 IoT system has a single node that performs sensing and/or actuation and local analysis.

• Data is stored in the cloud and application is usually cloud based.

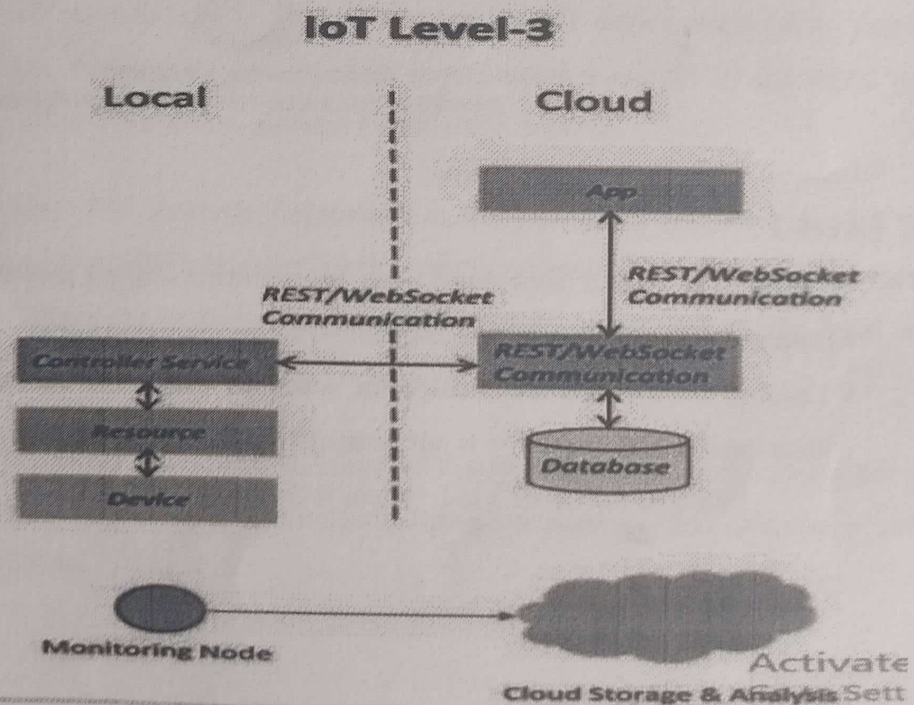
• Level-2 IoT systems are suitable for solutions where the data involved is big, however, the primary analysis requirement is not computationally intensive and can be done locally itself.

Example for Level-2 IoT system is Smart Irrigation.



IoT Level-3 :

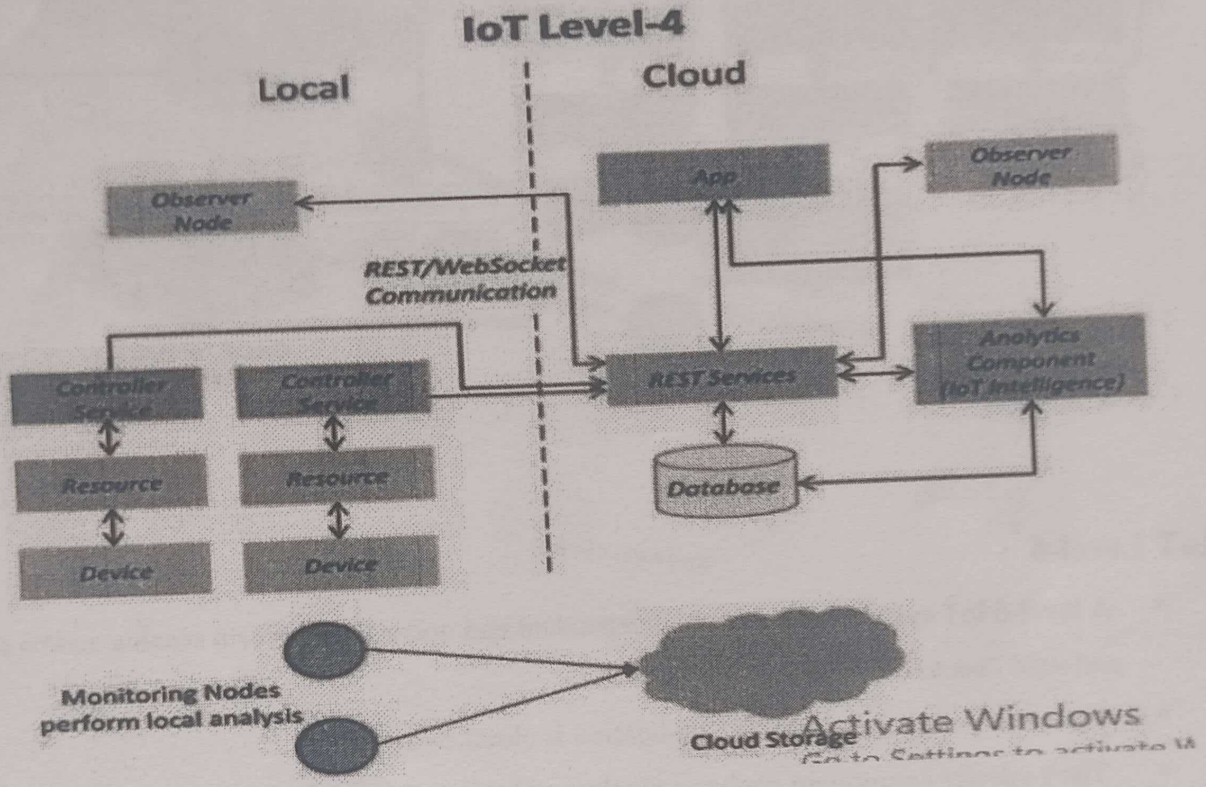
- A level-3 IoT system has a single node. Data is stored and analyzed in the cloud and application is cloud based.
- Level-3 IoT systems are suitable for solutions where the data involved is big and the analysis requirements are computationally intensive.
- Example for Level-3 IoT system is Tracking Package Handling.



IoT Level-4 :

- A level-4 IoT system has multiple nodes that perform local analysis. Data is stored in the cloud and application is cloud-based.
- Level-4 contains local and cloudbased observer nodes which can subscribe to and receive information collected in the cloud from IoT devices.
- Level-4 IoT systems are suitable for solutions where multiple nodes are required, the data involved is big and the analysis requirements are computationally intensive.

Example for Level-4 IoT system is Noise Monitoring

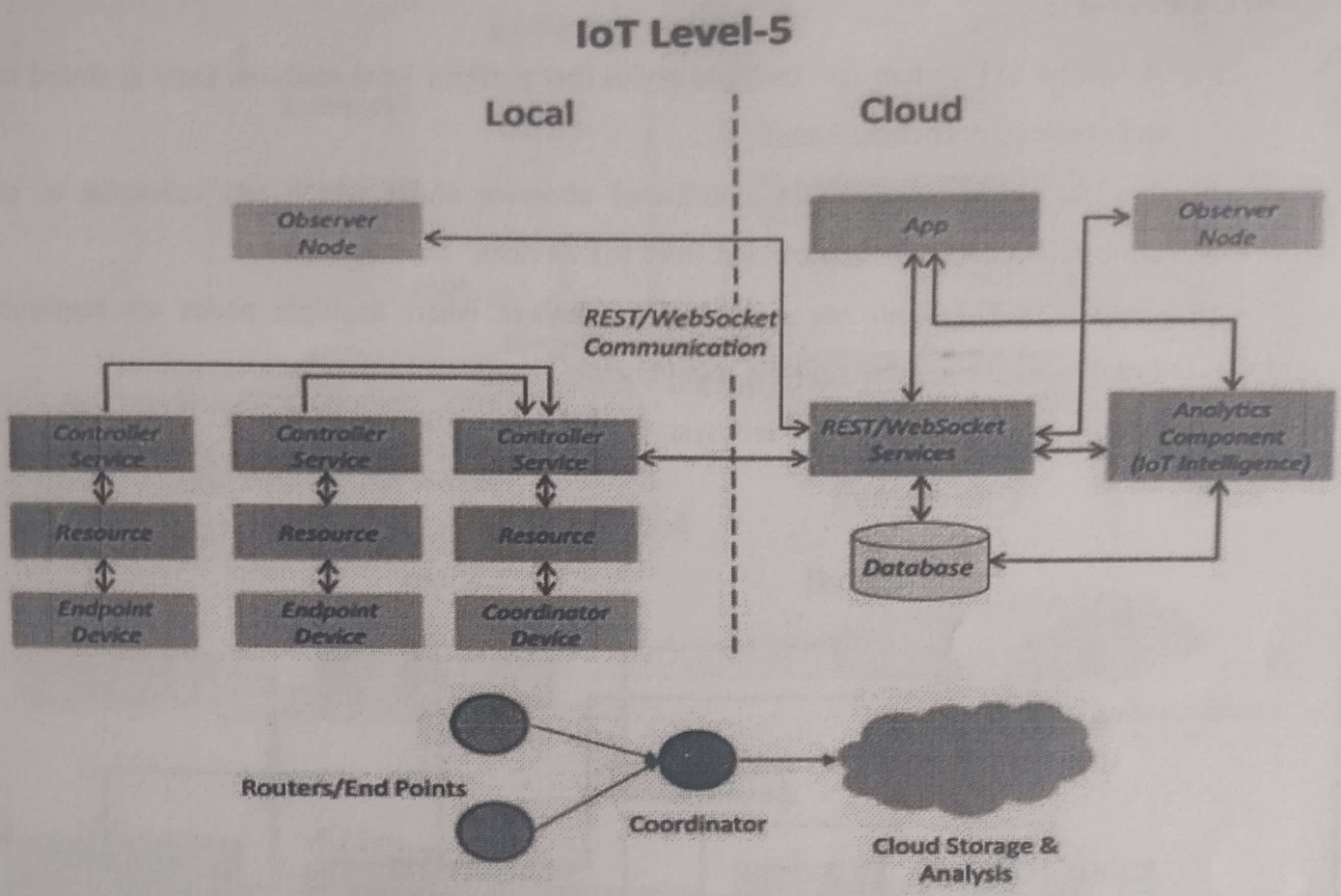


IoT Level-5 :

- A level-5 IoT system has multiple end nodes and one coordinator node.
- The end nodes that perform sensing and/or actuation.
- Coordinator node collects data from the end nodes and sends to the cloud.
- Data is stored and analyzed in the cloud and application is cloud-based.

Level-5 IoT systems are suitable for solutions based on wireless sensor networks, in which the data involved is big and the analysis requirements are computationally intensive.

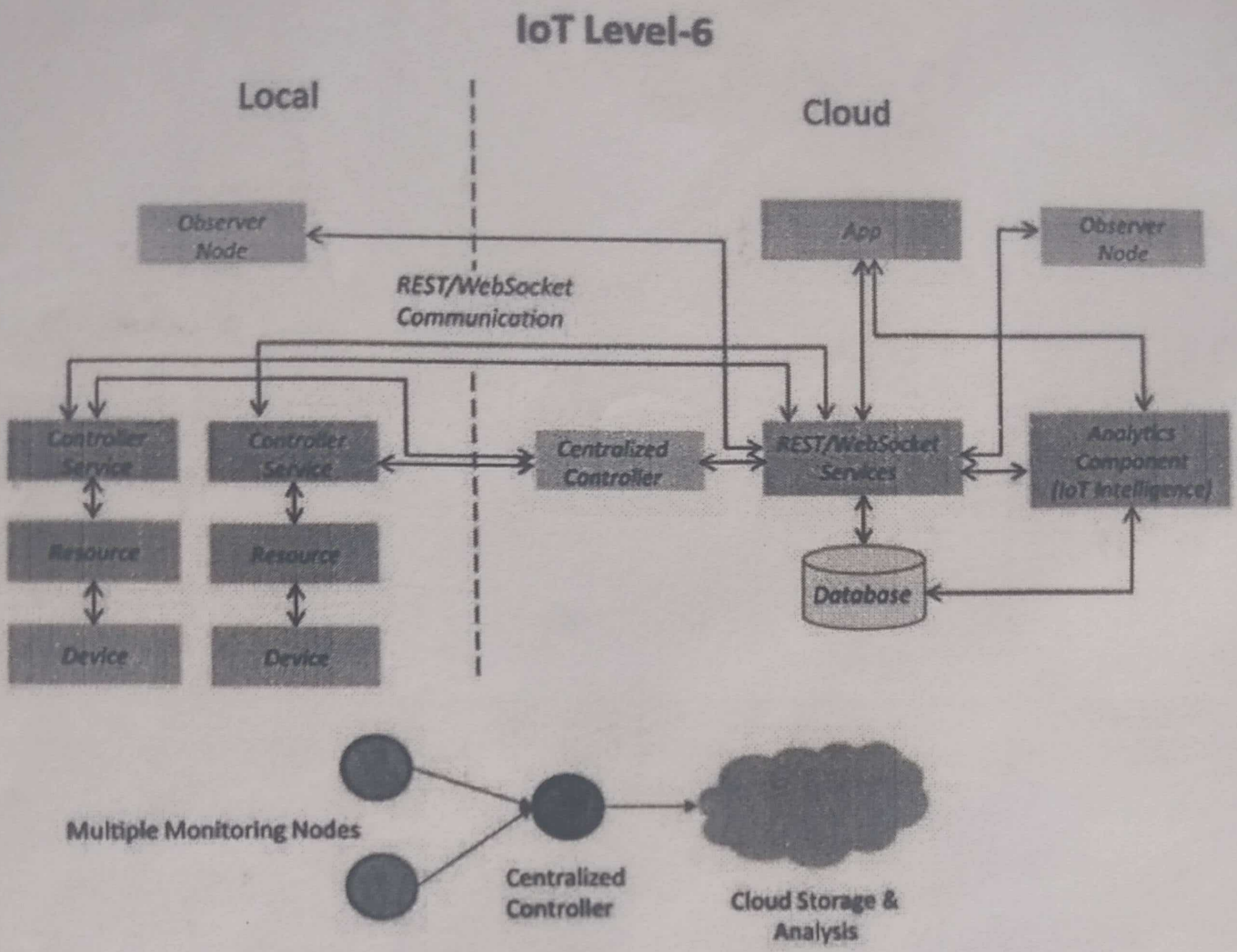
Example for Level-5 IoT system is forest fire detection.



IoT Level-6 :

- A level-6 IoT system has multiple independent end nodes that perform sensing and/or actuation and send data to the cloud.
- Data is stored in the cloud and application is cloud-based.
- The analytics component analyzes the data and stores the results in the cloud database.
- The results are visualized with the cloud-based application.
- The centralized controller is aware of the status of all the end nodes and sends control commands to the nodes.

Example for Level-6 IoT system is weather Monitoring.



****