UNIT -1 Introduction to IOT

IOT Definition: The **Internet of things** (**IoT**) is a system of interrelated computing devices, mechanical and digital machines provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

What is IoT?

The internet of Things, or "IoT" for short, is about extending the power of the internet beyond computers and smartphones to a whole range of other things, processes, and environments. Those "connected" things are used to gather information, send information back, or both.

Example

In the Internet of Things, all the things that are being connected to the internet can be put into three categories:

1. Things that collect information and then send it.
2. Things that receive information and then act on it.
3. Things that do both.

And all three of these have enormous benefits that feed on each other.

1. Collecting and Sending Information

This means sensors. Sensors could be temperature sensors, motion sensors, moisture sensors, air quality sensors, light sensors, you name it. These sensors, along with a connection, allow us to automatically collect information from the environment which, in turn, allows us to make more intelligent decisions.

(Soil moisture sensor)

On the farm, automatically getting information about the soil moisture can tell farmers exactly when their crops need to be watered. Instead of watering too much (which can be an expensive over-use of irrigation systems and environmentally wasteful) or watering too little (which can be an expensive loss of crops), the farmer can ensure that crops get exactly the right amount of water. More money for farmers and more food for the world!

Just as our sight, hearing, smell, touch, and taste allow us, humans, to make sense of the world, sensors allow machines to make sense of the world.

2. Receiving and Acting on Information

We're all very familiar with machines getting information and then acting. Your printer receives a document and it prints it. Your car receives a signal from your car keys and the doors open. The examples are endless.

Whether it's a simple as sending the command "turn on" or as complex as sending a 3D model to a 3D printer, we know that we can tell machines what to do from far away. So what?

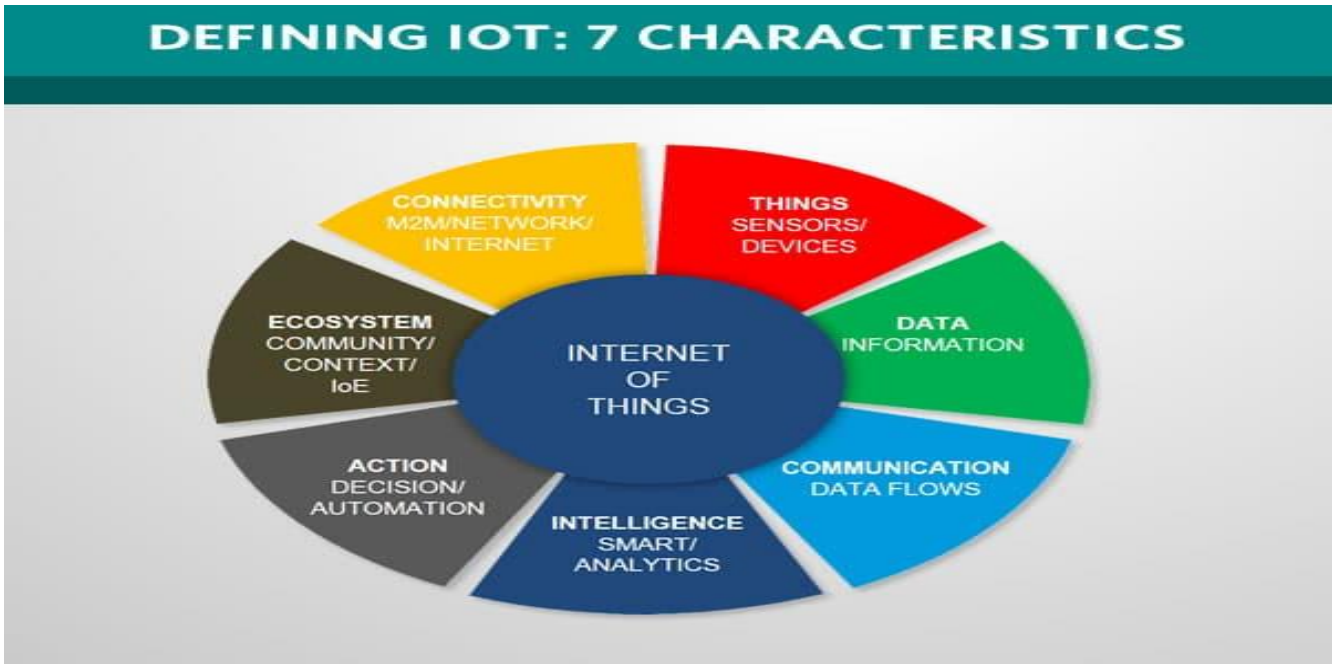
The real power of the Internet of Things arises when things can do both of the above. Things that collect information and send it, but also receive information and act on it.

3. Doing Both

Let's quickly go back to the farming example. The sensors can collect information about the soil moisture to tell the farmer how much to water the crops, but you don't actually need the farmer. Instead, the irrigation system can automatically turn on as needed, based on how much moisture is in the soil.

Everything Connects to Internet (Smart Irrigation System)

Characteristics of IoT


DEFINING IOT: 7 CHARACTERISTICS
CONNECTIVITY M2M/NETWORK/ INTERNET
THINGS SENSORS/ DEVICES
ECOSYSTEM COMMUNITY/ CONTEXT/ IoE
INTERNET OF THINGS
DATA INFORMATION
ACTION DECISION/ AUTOMATION
COMMUNICATION DATA FLOWS
INTELLIGENCE SMART/ ANALYTICS

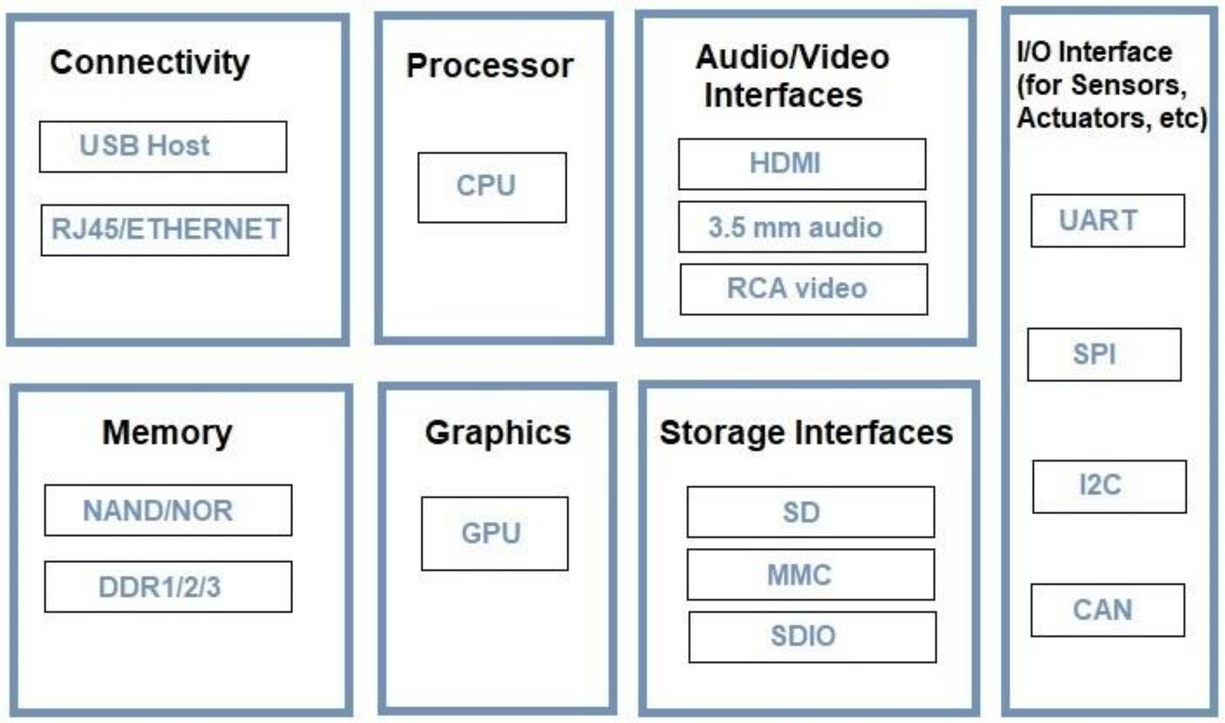
There are 7 crucial IoT characteristics:

1. **Connectivity.** This doesn’t need too much further explanation. With everything going on in IoT devices and hardware, with sensors and other electronics and connected hardware and control systems there needs to be a connection between various levels.
2. **Things**. Anything that can be tagged or connected as such as it’s designed to be connected. From sensors and household appliances to tagged livestock. Devices can contain sensors or sensing materials can be attached to devices and items.
3. **Data**. Data is the glue of the Internet of Things, the first step towards action and intelligence.
4. **Communication**. Devices get connected so they can communicate data and this data can be analyzed. Communication can occur over short distances or over a long range to very long range. Examples: Wi-Fi, LPWA network technologies such as **LoRa** (Long Range) is a low-power wide-area network (LPWAN) or NB-IoT(**Narrowband IoT**).
5. **Intelligence**. The aspect of intelligence as in the sensing capabilities in IoT devices and the intelligence gathered from big data analytics (also artificial intelligence).
6. **Action**. The consequence of intelligence. This can be manual action, action based upon debates regarding phenomena (for instance in smart factory decisions) and automation, often the most important piece.
7. **Ecosystem**. The place of the Internet of Things from a perspective of other technologies, communities, goals and the picture in which the Internet of Things fits. The Internet of Everything dimension, the platform dimension and the need for solid partnerships.

Physical Design of IoT

Things

Basically Things refers to IoT Devices which have unique identities and can perform remote sensing, actuating and monitoring capabilities. Things are is main part of IoT Application. IoT Devices can be various type, Sensing Devices, Smart Watches, Smart Electronics appliances, Wearable Sensors, Automobiles, and industrial machines. These devices generate data in some forms or the other which when processed by data analytics systems leads to useful information to guide further actions locally or remotely.



Generic Block Diagram of IoT Devices

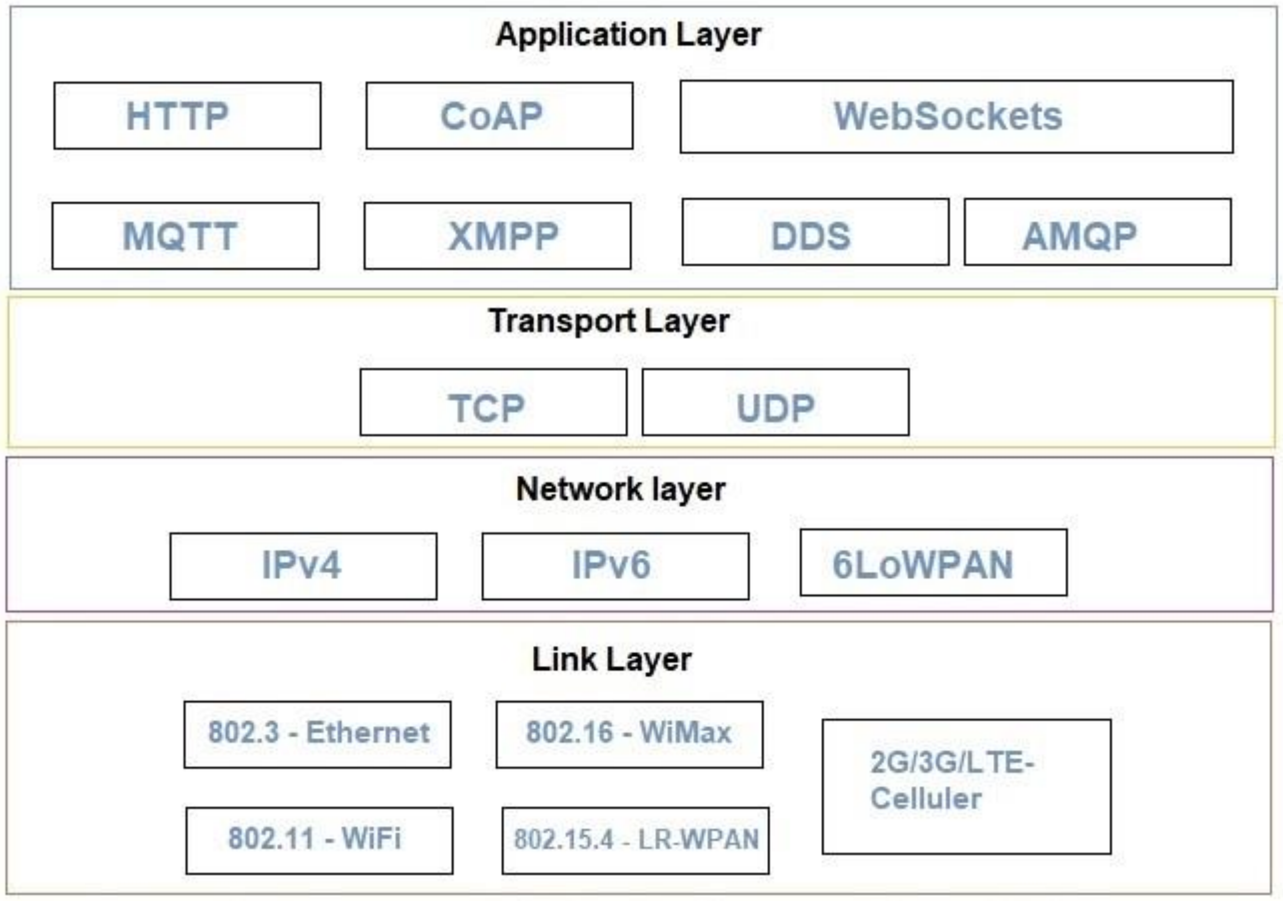
For example, Temperature data generated by a Temperature Sensor in Home or other place, when processed can help in determining temperature and take action according to users. Above picture, shows a generic block diagram of IoT device. It may consist of several interfaces for connections to other devices. IoT Device has I/O interface for Sensors, Similarly for Internet connectivity, Storage and Audio/Video. IoT Device collect data from on-board or attached Sensors and Sensed data communicated either to other device or Cloud based sever. Today many cloud servers available for especially IoT System. These Platfrom known as IoT Platform. Actually these cloud especially design for IoT purpose. So here we can analysis and processed data easily.

How it works?

For example, if relay switch connected to an IoT device can turn On/Off an appliance on the commands sent to the IoT device over the Internet.

IoT Protocols

IoT protcols help to establish Communication between IoT Device (Node Device) and Cloud based Server over the Internet. It helps to send commands to IoT Device and received data from an IoT device over the Internet. An image is given below. By this image you can understand which protocols used.



Link Layer

Link layer protocols determine how data is physically sent over the network's physical layer or medium (Coaxial cable or other or radio wave). This Layer determines how the packets are coded and signaled by the hardware device over the medium to which the host is attached (eg. coaxial cable).

Here we explain some Link Layer Protocols:

802.3 – Ethernet : Ethernet is a set of technologies and protocols that are used primarily in LANs. It was first standardized in 1980s by IEEE 802.3 standard. IEEE 802.3 defines the physical layer and the medium access control (MAC) sub-layer of the data link layer for wired Ethernet networks. Ethernet is classified into two categories: classic Ethernet and switched Ethernet.
802.11 – WiFi : IEEE 802.11 is part of the IEEE 802 set of LAN protocols, and specifies the set of media access control (MAC) and physical layer (PHY) protocols for implementing wireless local area network (WLAN) Wi-Fi computer communication in various frequencies, including but not limited to 2.4 GHz, 5 GHz, and 60 GHz frequency bands.
802.16 – Wi-Max : The standard for WiMAX technology is a standard for Wireless Metropolitan Area Networks (WMANs) that has been developed by working group number 16 of IEEE 802, specializing in point-to-multipoint broadband wireless access.

802.15.4 -LR-WPAN: A collection of standards for Low-rate wireless personal area network. The IEEE's 802.15.4 standard defines the MAC and PHY layer used by, but not limited to, networking specifications such as ZigBee®, 6LoWPAN, Thread, WiSUN and MiWi protocols. The standards provide low-cost and low-speed communication for power constrained devices.
2G/3G/4G- Mobile Communication: These are different types of telecommunication generations. IoT devices are based on these standards can communicate over the cellular networks.

Network Layer
Responsible for sending of IP datagrams from the source network to the destination network. Network layer performs the host addressing and packet routing. We used IPv4 and IPv6 for Host identification. IPv4 and IPv6 are hierarchical IP addressing schemes.

IPv4:
An **Internet Protocol address** (**IP address**) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. An IP address serves two main functions: host or network interface identification and location addressing. Internet Protocol version 4 (IPv4) defines an IP address as a 32-bit number. However, because of the growth of the Internet and the depletion of available IPv4 addresses, a new version of IP (IPv6), using 128 bits for the IP address, was standardized in 1998. IPv6 deployment has been ongoing since the mid-2000s.

IPv6 : Internet Protocol version 6 (**IPv6**) is successor of IPv4. IPv6 was developed by the Internet Engineering Task Force (IETF) to deal with the long-anticipated problem of IPv4 address exhaustion. In December 1998, IPv6 became a Draft Standard for the IETF, who subsequently ratified it as an Internet Standard on 14 July 2017. IPv6 uses a 128-bit address, theoretically allowing 2^{128}, or approximately 3.4×10^{38} addresses.

6LoWPAN : It is an acronym of *IPv6 over Low-Power Wireless Personal Area Networks*. 6LoWPAN is the name of a concluded working group in the Internet area of the IETF. This protocol allows for the smallest devices with limited processing ability to transmit information wirelessly using an internet protocol. 6LoWPAN can communicate with 802.15.4 devices as well as other types of devices on an IP network link like WiFi.

Transport Layer
This layer provides functions such as error control, segmentation, flow control and congestion control. So this layer protocols provide end-to-end message transfer capability independent of the underlying network.

TCP : TCP (Transmission Control Protocol) is a standard that defines how to establish and maintain a network conversation through which application programs can exchange data. TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other. Together, TCP and IP are the basic rules defining the Internet. The Internet Engineering Task Force (IETF) defines TCP in the Request for Comment (RFC) standards document number 793.

UDP : User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of Internet Protocol suite, referred as UDP/IP suite. Unlike TCP, it is unreliable and connectionless protocol. So, there is no need to establish connection prior to data transfer.

Application Layer

Application layer protocols define how the applications interface with the lower layer protocols to send over the network.

HTTP : *Hypertext Transfer Protocol (HTTP)* is an application-layer protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes. HTTP follows a classical client-server model, with a client opening a connection to make a request, then waiting until it receives a response. HTTP is a stateless protocol, meaning that the server does not keep any data (state) between two requests.

CoAP : CoAP-Constrained Application Protocol is a specialized Internet Application Protocol for constrained devices, as defined in RFC 7252. It enables devices to communicate over the Internet. The protocol is especially targeted for constrained hardware such as 8-bits microcontrollers, low power sensors and similar devices that can't run on HTTP or TLS.

WebSocket : The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code. The security model used for this is the origin-based security model commonly used by web browsers.

MQTT : MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport and useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium

XMPP : Extensible Messaging and Presence Protocol (**XMPP**) is a communication protocol for message-oriented middleware based on XML (Extensible Markup Language). It enables the near-real-time exchange of structured yet extensible data between any two or more network entities.

DDS : The Data Distribution Service (DDS™) is a middleware protocol and API standard for data-centric connectivity from the Object Management Group® (OMG®). It integrates the components of a system together, providing low-latency data connectivity, extreme reliability, and a scalable architecture that business and mission-critical Internet of Things (IoT) applications need.

AMQP : The AMQP(Advanced Message Queuing Protocol) – IoT protocols consist of a hard and fast of components that route and save messages within a broker carrier, with a set of policies for wiring the components together. The AMQP protocol enables patron programs to talk to the dealer and engage with the AMQP model.

Logical Design of IoT | IoT Communication Models & APIs

Logical Design of IoT

In Logical design of Internet of things. Logical design of IoT system refers to an abstract representation of the entities & processes without going into the low-level specifies of the implementation. For understanding Logical Design of IoT, we describe given below terms.

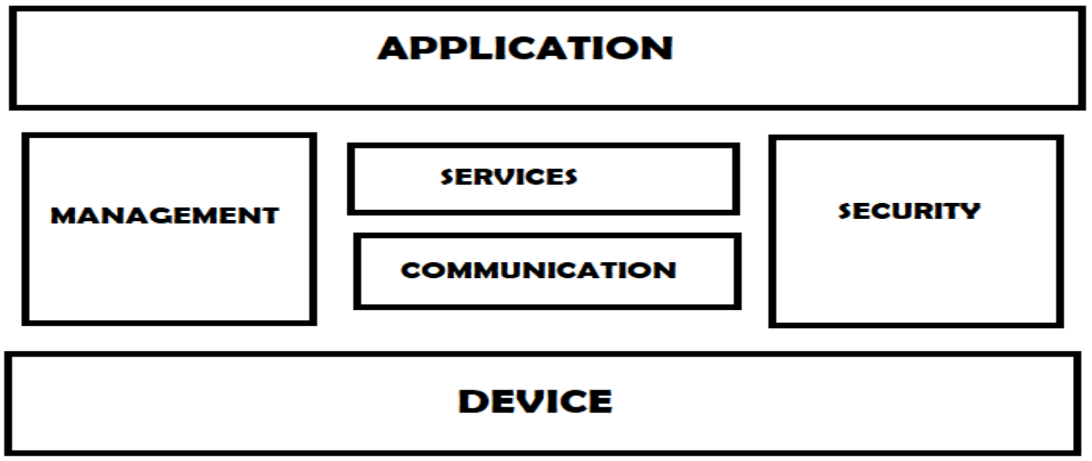
- IoT Functional Blocks
- IoT Communication Models
- IoT Communication APIs

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.

functional blocks are:

Device: An IoT system comprises of devices that provide sensing, actuation, monitoring and control functions.
Communication: Handles the communication for the IoT system.
Services: services for device monitoring, device control service, data publishing services and services for device discovery.
Management: this blocks provides various functions to govern the IoT system.
Security: this block secures the IoT system and by providing functions such as authentication, authorization, message and content integrity, and data security.
Application: This is an interface that the users can use to control and monitor various aspects of the IoT system. Application also allow users to view the system status and view or analyze the processed data.



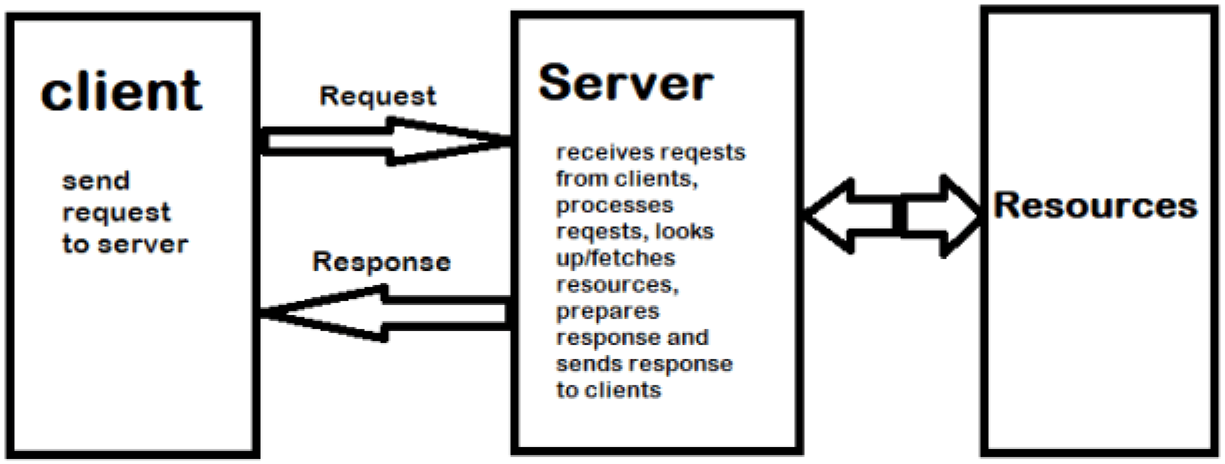
IoT Communication Models

Request-Response Model

Request-response model is 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 representation, prepares the response, and then sends the response to the client. Request-response is a stateless communication model and each request-response pair is independent of others.

HTTP works as a request-response protocol between a client and server. A web browser may be the client, and an application on a computer that hosts a web site may be the server.

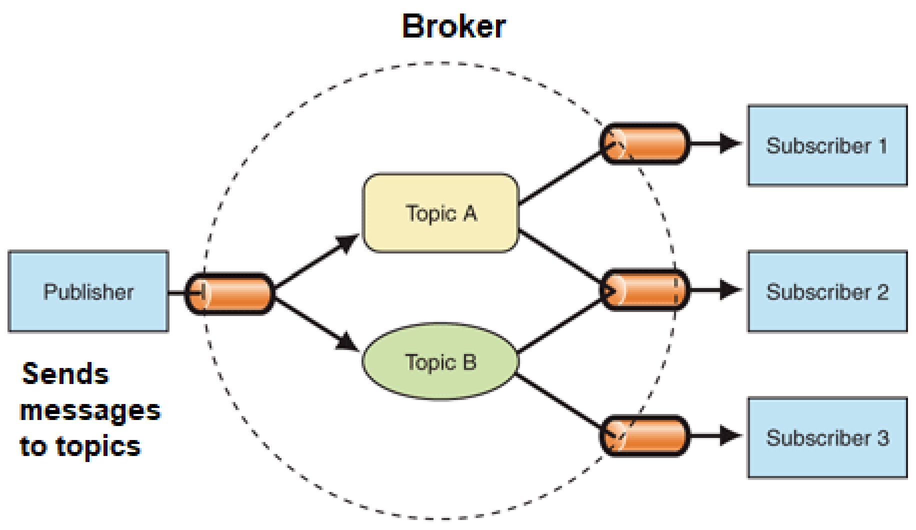
Example: A client (browser) submits an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.


client
send request to server
Request
Response
Server
receives reqests from clients, processes reqests, looks up/fetches resources, prepares response and sends response to clients
Resources

Request-Response Communication Model

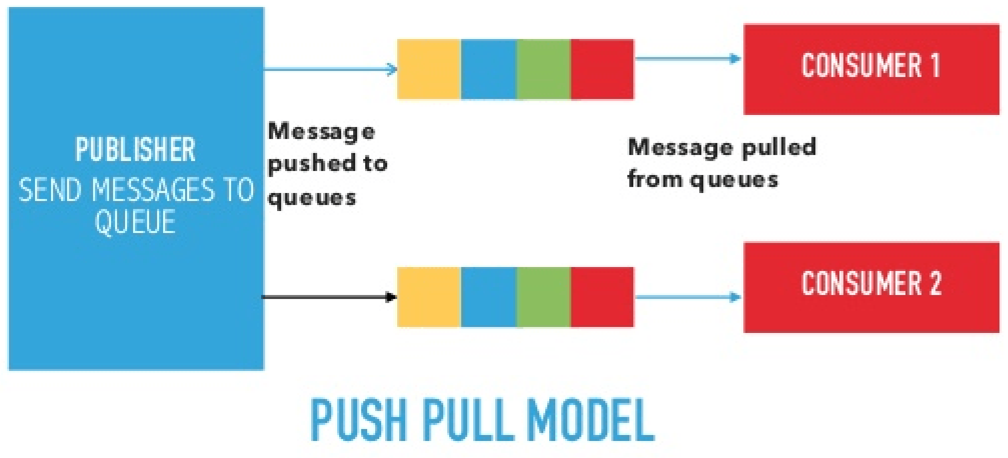
Publish-Subscribe Model

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 receive data for a topic from the publisher, it sends the data to all the subscribed consumers.



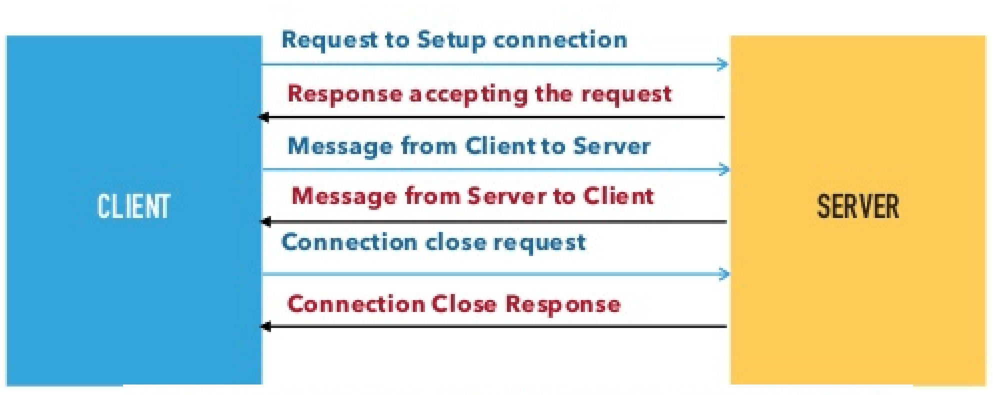
Push-Pull Model

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 rate at which the consumer pull data.



Exclusive Pair Model

Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and server. 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 stateful communication model and the server is aware of all the open connections.


Request to Setup connection
Response accepting the request
Message from Client to Server
Message from Server to Client
Connection close request
Connection Close Response
CLIENT
SERVER

EXCLUSIVE PAIR COMMUNICATION MODEL

IoT Communication APIs

Generally we used Two APIs For IoT Communication. These IoT Communication APIs are:

- REST-based Communication APIs
- WebSocket-based Communication APIs
-

REST-based Communication APIs

Representational state transfer (REST) is a set of architectural principles by which you can design Web services the Web APIs that focus on systems resources and how resource states are addressed and transferred. REST APIs that follow the request response communication model, the rest architectural constraint applies to the components, connector and data elements, within a distributed hypermedia system. The rest architectural constraint 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 concern of the serve. Similarly the server should not be concerned about the user interface, which is concern of the clien. Separation allows client and server to be independently developed and updated.

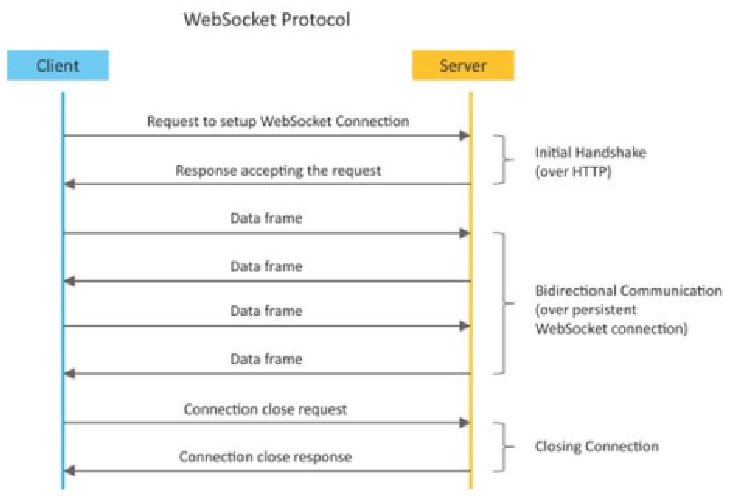
Stateless – Each request from client to server must contain all the information necessary to understand the request, and cannot take advantage of any stored context on the server. The session state is kept entirely on the client.
Cache-able – Cache constraints requires that the data within a response to a request be implicitly or explicitly leveled 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 instructions and improve efficiency and scalability.
Layered system – layered system constraints, constrains the behavior of components such that each component cannot see beyond the immediate layer with they are interacting. For example, the client cannot tell whether it is connected directly to the end server or two 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 constraints requires that the method of communication between client and server must be uniform. Resources are identified in the requests (by URIsin web based systems) and are themselves is separate from the representations of the resources data returned to the client. When a client holds a representation of resources it has all the information required to update or delete the resource you (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. REST is most popular IoT Communication APIs.

Uniform Resource Identifier (URI)	**GET**	**PUT**	**PATCH**	**POST**	**DELETE**
Collection, such as `https://api.example.com/resources/`	*List* the URIs and perhaps other details of the collection's members.	*Replace* the entire collection with another collection.	Not generally used	*Create* a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the	*Delete* the entire collection.

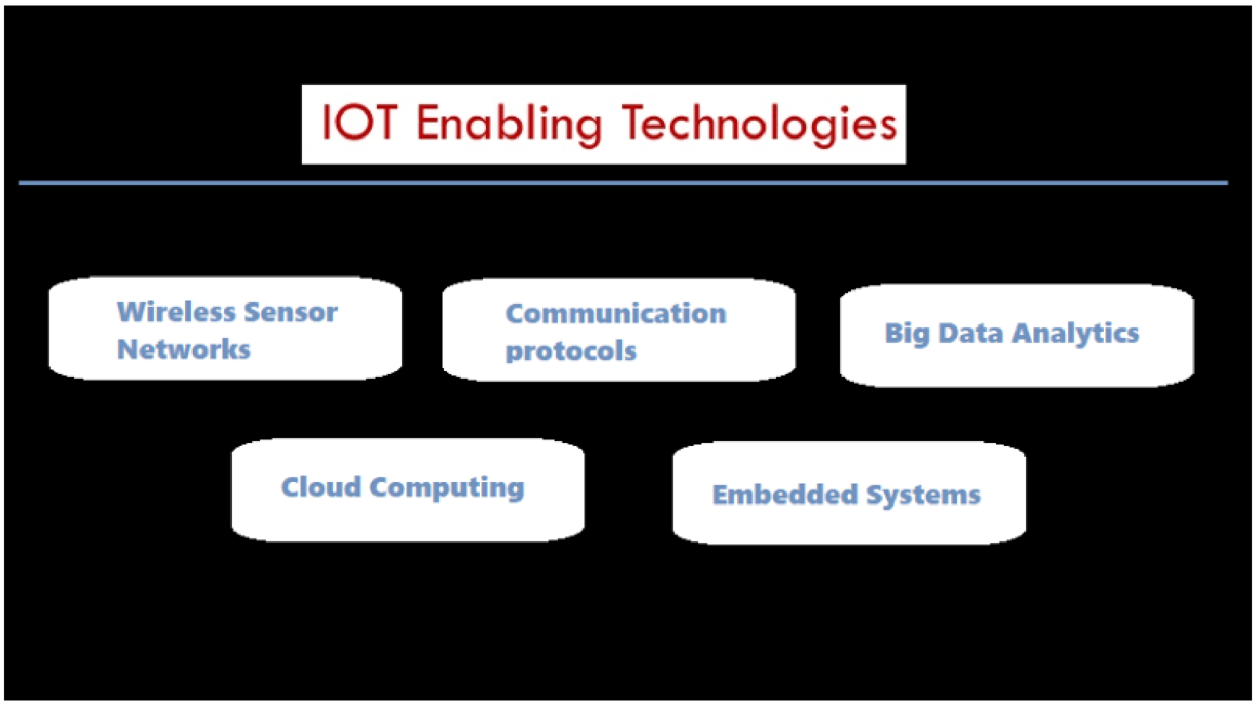
				operation.	
Element, such as https://api.example.com/resources/item5	*Retrieve* a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	*Replace* the addressed member of the collection, or if it does not exist, create it.	*Update* the addressed member of the collection.	Not generally used. Treat the addressed member as a collection in its own right and create a new entry within it.	*Delete* the addressed member of the collection.

WebSocket based communication API

Websocket APIs allow bi-directional, full duplex communication between clients and servers. Websocket APIs follow the exclusive pair communication model. Unlike request-response model such as REST, the Websocket APIs allow full duplex communication and do not require 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. The request (called websocket handshake) is sent over HTTP and the server interprets it is an upgrade request. If the server supports websocket protocol, the server responds to the websocket handshake response. After the connection setup client and server can send data/messages to each other in full duplex mode. Websocket API reduce the network traffic and latency as there is no overhead for connection setup and termination requests for each message. Websocket suitable for IoT applications that have low latency or high throughput requirements. So Web socket is most suitable IoT Communication APIs for IoT System.


WebSocket Protocol
Client
Server
Request to setup WebSocket Connection
Response accepting the request
Initial Handshake (over HTTP)
Data frame
Data frame
Data frame
Data frame
Bidirectional Communication (over persistent WebSocket connection)
Connection close request
Connection close response
Closing Connection

IoT Enabling Technologies



IoT is enabled by several technologies including wireless sensor networks, cloud computing, Big data analytics, Embedded Systems, Security Protocols and architectures, communication protocols, web services, Mobile Internet, and Semantic Search engines.

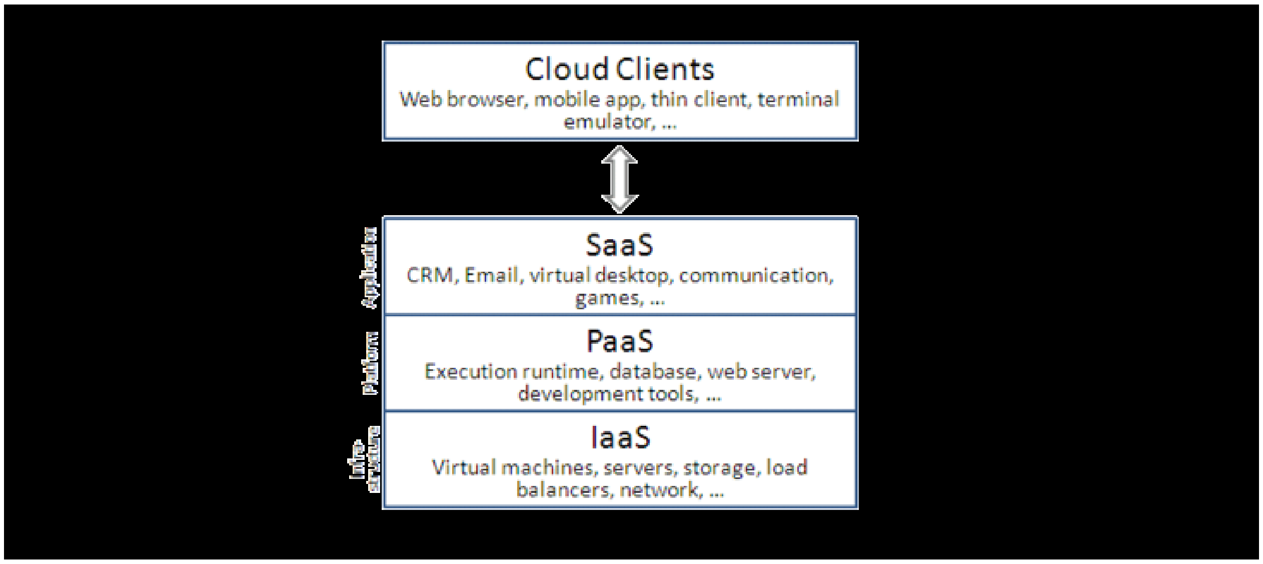
Wireless Sensor Networks

A wireless sensor network comprises of distributed device with sensor which are used to monitor the environmental and physical conditions. A WSN consists of a number of end-nodes and routers and a coordinator. End Nodes have several sensors attached to them in node can also act as routers. Routers are responsible for routing the data packets from end-nodes to the coordinator. The coordinator collects the data from all the nodes. Coordinator also act as a gateway that connects the WSN to the internet. Some examples of WSNs used in IoT systems are described as follows:

- Weather monitoring system use WSNs in which the nodes collect temperature humidity and other data which is aggregated and analyzed.
- Indoor air quality monitoring systems use WSNs to collect data on the indoor air quality and concentration of various gases
- Soil moisture monitoring system use WSNs to monitor soil moisture at various locations.
- Surveillance system use WSNs for collecting Surveillance data (such as motion detection data)
- Smart grid use WSNs for monitoring the grid at various points.
- Structural health monitoring system use WSNs to monitor the health of structures (buildings, bridges) by collecting vibration data from sensor nodes de deployed at various points in the structure.

Cloud Computing

Cloud computing is a trans-formative computing paradigm that involves delivering applications and services over the Internet Cloud computing involves provisioning of computing, networking and storage resources on demand and providing these resources as metered services to the users, in a “pay as you go” model. C loud computing resources can be provisioned on demand by the users, without requiring interactions with the cloud service Provider. The process of provisioning resources is automated. Cloud computing resources can be accessed over The network using standard access mechanisms that provide platform independent access through the use of heterogeneous client platforms such as the workstations, laptops, tablets and smartphones.


Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...
SaaS
CRM, Email, virtual desktop, communication, games, ...
PaaS
Execution runtime, database, web server, development tools, ...
IaaS
Virtual machines, servers, storage, load balancers, network, ...
Application
Platform
Infra-structure

Cloud computing services are offered to users in different forms:

Infrastructure as a Service (IaaS): hardware is provided by an external provider and managed for you
Platform as a Service (PaaS): in addition to hardware, your operating system layer is managed for you
Software as a Service (SaaS): further to the above, an application layer is provided and managed for you – you won't see or have to worry about the first two layers.

Big Data Analytics

Big Data analytics is the process of collecting, organizing and analyzing large sets of data (*called* Big Data) to discover patterns and other useful information. Big Data analytics can help organizations to better understand the information contained within the data and will also help identify the data that is most important to the business and future business decisions. Analysts working with Big Data typically want the *knowledge* that comes from analyzing the data.
Some examples of big data generated by IoT systems are described as follows:

- Sensor data generated by IoT system such as weather monitoring stations.
- Machine sensor data collected from sensors embedded in industrial and energy systems for monitoring their health and detecting Failures.
- Health and fitness data generated by IoT devices such as wearable fitness bands
- Data generated by ioT systems for location and tracking of vehicles
- Data generated by retail inventory monitoring systems

Characteristics
Big data can be described by the following characteristics:

- **Volume –** The quantity of generated and stored data. The size of the data determines the value and potential insight, and whether it can be considered big data or not.

- **Variety –** The type and nature of the data. This helps people who analyze it to effectively use the resulting insight. Big data draws from text, images, audio, video; plus it completes missing pieces through data fusion.
- **Velocity –** In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development. Big data is often available in real-time. Compared to small data, big data are produced more continually. Two kinds of velocity related to Big Data are the frequency of generation and the frequency of handling, recording, and publishing.
- **Veracity –** It is the extended definition for big data, which refers to the data quality and the data value. The data quality of captured data can vary greatly, affecting the accurate analysis.

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. Multiple protocols often describe different aspects of a single communication. A group of protocols designed to work together are known as a protocol suite; when implemented in software they are a protocol stack.

Internet communication protocols are published by the Internet Engineering Task Force (IETF). The IEEE handles wired and wireless networking, and the International Organization for Standardization (ISO) handles other types. The ITU-T handles telecommunication protocols and formats for the public switched telephone network (PSTN). As the PSTN and Internet converge, the standards are also being driven towards convergence.

In IoT we used MQTT, COAP, AMQP etc. protocols. You can read in detail by given below links.

Embedded Systems

As its name suggests, Embedded means something that is attached to another thing. An embedded system can be thought of as a computer hardware system having software embedded in it. An embedded system can be an independent system or it can be a part of a large system. An embedded system is a controller programmed and controlled by a real-time operating system (RTOS) with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is *embedded* as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today. Ninety-eight percent of all microprocessors are manufactured to serve as embedded system component.
An embedded system has three components –

- It has hardware.
- It has application software.
- It has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies. RTOS defines the way the system works. It sets the rules during the execution of application program. A small scale embedded system may not have RTOS.

IoT Levels and Deployment Templates

1.IoT Levels and Deployment Templates An IoT system comprises the following components: Device, Resource, Controller Service, Database, Web service, Analysis Component and Application. Device: An IoT device allows identification, remote sensing, remote monitoring capabilities. Resource:

• Software components on the IoT device for -accessing, processing and storing sensor information, -controlling actuators connected to the device. - enabling network access for the device. Controller Service:

• Controller service is a native service that runs on the device and interacts with the web services.

• It sends data from the device to the web service and receives commands from the application (via web services) for controlling the device.

2. 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.

•It can be implemented using HTTP and REST principles (REST service) or using the WebSocket protocol (WebSocket service). Analysis Component:

• Analysis Component is responsible for analyzing the IoT data and generating results in a form that is easy for the user to understand. 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 the processed data.

3. 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 modelling low- cost and low-complexity solutions where the data involved is not big and the analysis requirements are not computationally intensive.

IoT – Level 1 Example : Home Automation System

4. 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 the 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.

IoT – Level 2 Example: Smart Irrigation

5. IoT Level-3

A level-3 IoT system has a single node. Data is stored and analyzed in the cloud and the 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.

IoT – Level 3 Example: Tracking Package Handling Sensors used Gives orientation infosense movement or vibrations Websocket service is used because sensor data can be sent in real time. Accelrometer Gyroscope

6. IoT Level-4

A level-4 IoT system has multiple nodes that perform local analysis. Data is stored in the cloud and the application is cloud-based. Level-4 contains local and cloud- based observer nodes which can subscribe 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.

IoT – Level 4 Example: Noise Monitoring Sound Sensors are used

7. IoT Level-5

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

IoT – Level 5 Example: Forest Fire Detection Detect forest fire in early stages to take action while the fire is still controllable. Sensors measure the temperature, smoke, weather, slope of the earth, wind speed, speed of fire spread, flame length

8.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 the 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.

IoT – Level 6 Example: Weather Monitoring System Wind speed and direction Solar radiation Temperature (air, water, soil) Relative humidity Sensors used Precipitation Snow depth Barometric pressure Soil moisture

IoT Issues and Challenges Security • Cyber Attacks, Data Theft Privacy • Controlling access and ownership of data. Interoperability • Integration Inflexibility Legality and Rights • Data Protection laws be followed, Data Retention and destruction policies Economy and Development • Investment Incentives, Technical Skill Requirement

IoT security challenges

1.Secure constrained devices

Many IoT devices have limited amounts of storage, memory, and processing capability and they often need to be able to operate on lower power, for example, when running on batteries.

Security approaches that rely heavily on encryption are not a good fit for these constrained devices, because they are not capable of performing complex encryption and decryption quickly enough to be able to transmit data securely in real-time.

These devices are often vulnerable to side-channel attacks, such as power analysis attacks, that can be used to reverse engineer these algorithms. Instead, constrained devices typically only employ fast, lightweight encryption algorithms.

IoT systems should make use of multiple layers of defense, for example, segregating devices onto separate networks and using firewalls, to compensate for these device limitations.

2 Authorize and authenticate devices

With so many devices offering potential points of entry within an IoT system, device authentication and authorization is critical for securing IoT systems.

Devices must establish their identity before they can access gateways and upstream services and apps. However, there are many IoT devices that fall down when it comes to device authentication, for example, by using weak basic password authentication or using passwords unchanged from their default values.

Adopting an IoT Platform that provides security by default helps to resolve these issues, for example by enabling two factor authentication (2FA) and enforcing the use of strong passwords or certificates. IoT Platforms also provide device authorization services used to determine which services, apps, or resources that each device has access to throughout the system.

3 Manage device updates

Applying updates, including security patches, to firmware or software that runs on IoT devices and gateways presents a number of challenges. For example, you need to keep track of which updates are available, apply them synchronously across distributed environments with heterogeneous devices that communicate through a range of different networking protocols. You have to implement a graceful rollback-strategy in case the update fails.

Not all devices support over-the-air updates, or updates without downtime, so devices might need to be physically accessed or temporarily pulled from production to apply updates. Also, updates might not be available for all devices, particularly older devices or those devices that are no longer supported by their manufacturer.

Even when updates are available, the owners of a device might opt out of applying an update, so maintaining backward-compatibility is important. As part of your device management, you need to keep track of the versions that are deployed on each device and which devices are candidates for retirement after updates are no longer available.

Device manager systems often support pushing out updates automatically to devices as well as managing rollbacks if the update process fails. They can also help to ensure that only legitimate updates are applied, for example through the use of digital signing.

4 Secure communication

Once the devices themselves are secured, the next IoT security challenge is to ensure that communication across the network between devices and cloud services or apps is secure.

Many IoT devices don't encrypt messages before sending them over the network. However, best practice is to use transport encryption and to adopt standards like TLS. Using separate networks to isolate devices also helps with establishing secure, private communication, so that data transmitted remains confidential. Common measures include using Firewalls, restricting physical access to gateway devices, using randomly generated one-time-password, and turn off the OS features that aren't required by the device.

5 Ensure data privacy and integrity

It is also important that wherever the data ends up after it has been transmitted across the network, it is stored and processed securely. Implementing data privacy includes redacting or anonymizing sensitive data before it is stored or using data separation to decouple personally identifiable information from IoT data payloads. Data that is no longer required should be disposed of securely, and if data is stored, maintaining compliance with legal and regulatory frameworks is also an important challenge. There has been an increased interest in developing systems that enable federated machine learning, where the data stays local to the device and the machine learning work happens at edge, with only the learnings/insights being shared with the cloud. The essence of such systems is that the algorithm should come to the data instead of the conventional methods where all the data goes to the algorithm.

Ensuring data integrity, which may involve employing checksums or digital signatures to ensure data has not been modified. Blockchain – as a decentralized distributed ledger for IoT data – offers a scalable and resilient approach for ensuring the integrity of IoT data.

6 Secure web, mobile, and cloud applications

Web, mobile, and cloud apps and services are used to manage, access, and process IoT devices and data, so they must also be secured as part of a multi-layered approach to IoT security. All the standard practices to secure mobile and web apps should be followed.

When developing IoT applications, be sure to apply secure engineering practices to avoid vulnerabilities such as the OWASP top 10 vulnerabilities. Just like devices, apps should also support secure authentication, both for the apps themselves and the users of the applications, by providing options such as 2FA and secure password recovery options.

7 Ensure high availability

As we come to rely more on IoT within our day-to-day lives, IoT developers must consider the availability of IoT data and the web and mobile apps that rely on that data as well as our access to the physical things managed by IoT systems. The potential for disruption as a result of connectivity outages or device failures, or arising as a result of attacks like denial of service attacks, is more than just inconvenience. In some applications, the impact of the lack of availability could mean loss of revenue, damage to equipment, or even loss of life.

For example, in connected cities, IoT infrastructure is responsible for essential services such as traffic control, and in healthcare, IoT devices include pacemakers and insulin pumps. To ensure high availability, IoT devices must be protected against cyber-attacks as well as physical tampering. IoT systems must include redundancy to eliminate single points of failure, and should also be designed to be resilient and fault tolerant, so that they can adapt and recover quickly when problems do arise. Since most IoT systems are distributed in nature, an important thing to keep in mind is the CAP theorem (A distributed computer system can have at most two properties out of high availability, consistency, and tolerance to network partition).

8 Prevent incidents by detecting vulnerabilities

Despite best efforts, security vulnerabilities and breaches are inevitable. How do you know if your IoT system has been compromised? In large scale IoT systems, the complexity of the system in terms of the number of devices connected, and the variety of devices, apps, services, and communication protocols involved, can make it difficult to identify when an incident has occurred. Strategies for detecting vulnerabilities and breaches include monitoring network communications and activity logs for anomalies, engaging in penetration testing and ethical hacking to expose vulnerabilities, and applying security intelligence and analytics to identify and notify when incidents occur.

9 Manage vulnerabilities

The complexity of IoT systems also makes it challenging to assess the repercussions of a vulnerability or the extent of a breach in order to manage its impact. Challenges include identifying which devices were affected, what data or services were accessed or compromised and which users were impacted, and then taking immediate actions to resolve the situation.

Device managers maintain a register of devices, which can be used to temporarily disable or isolate affected devices until they can be patched. This feature is particularly important for key devices (such as gateway devices) in order to limit their potential to cause harm or disruption, such as by flooding the system with fake data if they have been compromised. Actions can be applied automatically using a rules engine with rules based on vulnerability management policies.

10 Predict and preempt security issues

A longer-term IoT security challenge is to apply security intelligence not only for detecting and mitigating issues as they occur, but also to predict and proactively protect against potential security threats. Threat modeling is one approach used to predict security issues. Other approaches include applying monitoring and analytics tools to correlate events and visualize unfolding threats in real-time, as well as applying AI to adaptively adjust security strategies applied based on the effectiveness of previous actions. The essence is to minimize human intervention and offload as much work as possible to algorithms since continuous manual examination is almost impossible.