FUNDAMENTALS OF IOT

Internet of Things (IoT) is the networking of physical objects that contain electronics embedded within their architecture in order to communicate and sense interactions amongst each other or with respect to the external environment. In the upcoming years, IoT-based technology will offer advanced levels of services and practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of the categorical examples where IoT is strongly established.

Over 9 billion 'Things' (physical objects) are currently connected to the Internet, as of now. In the near future, this number is expected to rise to a whopping 20 billion.

There are four main components used in IoT:

Low-power embedded systems: Less battery consumption, high performance are the inverse factors that play a significant role during the design of electronic systems.

Cloud computing: Data collected through IoT devices is massive and this data has to be stored on a reliable storage server. This is where cloud computing comes into play. The data is processed and learned, giving more room for us to discover where things like electrical faults/errors are within the system.

Availability of big data: We know that IoT relies heavily on sensors, especially in real-time. As these electronic devices spread throughout every field, their usage is going to trigger a massive flux of big data.

Networking connection: In order to communicate, internet connectivity is a must where each physical object is represented by an IP address. However, there are only a limited number of addresses available according to the IP naming. Due to the growing number of devices, this naming system will not be feasible anymore. Therefore, researchers are looking for another alternative naming system to represent each physical object.

There are two ways of building IoT:

- Form a separate internetwork including only physical objects.
- Make the Internet ever more expansive, but this requires hard-core technologies such as rigorous cloud computing and rapid big data storage (expensive).

IoT Enablers:

RFIDs: uses radio waves in order to electronically track the tags attached to each physical object.

Sensors: devices that are able to detect changes in an environment (ex: motion detectors).

Nanotechnology: as the name suggests, these are extremely small devices with dimensions usually less than a hundred nanometers.

Smart networks: (ex: mesh topology).

Characteristics of IoT:

Massively scalable and efficient

IP-based addressing will no longer be suitable in the upcoming future.

An abundance of physical objects is present that do not use IP, so IoT is made possible.

Devices typically consume less power. When not in use, they should be automatically programmed to sleep.

A device that is connected to another device right now may not be connected in another instant of time.

Intermittent connectivity – IoT devices aren't always connected. In order to save bandwidth and battery consumption, devices will be powered off periodically when not in use. Otherwise, connections might turn unreliable and thus prove to be inefficient.

As a quick note, IoT incorporates trillions of sensors, billions of smart systems, and millions of applications.

Modern Applications:

1. Smart Grids and energy saving
2. Smart cities
3. Smart homes
4. Healthcare
5. Earthquake detection
6. Radiation detection/hazardous gas detection
7. Smartphone detection
8. Water flow monitoring
9. Traffic monitoring
10. Wearables

Characteristics of the Internet of Things:

There are the following characteristics of IoT as follows. Let's discuss it one by one.

Connectivity –

Connectivity is an important requirement of the IoT infrastructure. Things of IoT should be connected to the IoT infrastructure. Anyone, anywhere, anytime can connectivity should be guaranteed at all times Without connection, nothing makes sense.

Intelligence and Identity –

The extraction of knowledge from the generated data is very important. For example, a sensor generates data, but that data will only be useful if it is interpreted properly. Each IoT device has a unique identity. This identification is helpful in tracking the equipment and at times for querying its status.

Scalability –

The number of elements connected to the IoT zone is increasing day by day. Hence, an IoT setup should be capable of handling the massive expansion. The

data generated as an outcome is enormous, and it should be handled appropriately.

Dynamic and Self-Adapting (Complexity) –

IoT devices should dynamically adapt themselves to the changing contexts and scenarios. Assume a camera meant for the surveillance. It should be adaptable to work in different conditions and different light situations (morning, afternoon, night).

Architecture –

IoT architecture cannot be homogeneous in nature. It should be hybrid, supporting different manufacturers ' products to function in the IoT network. IoT is not owned by anyone engineering branch. IoT is a reality when multiple domains come together.

Safety –

There is a danger of the sensitive personal details of the users getting compromised when all his/her devices are connected to the internet. This can cause a loss to the user. Hence, data security is the major challenge. Besides, the equipment involved is huge. IoT networks may also be at the risk. Therefore, equipment safety is also critical.

Physical and Logical Design of IoT

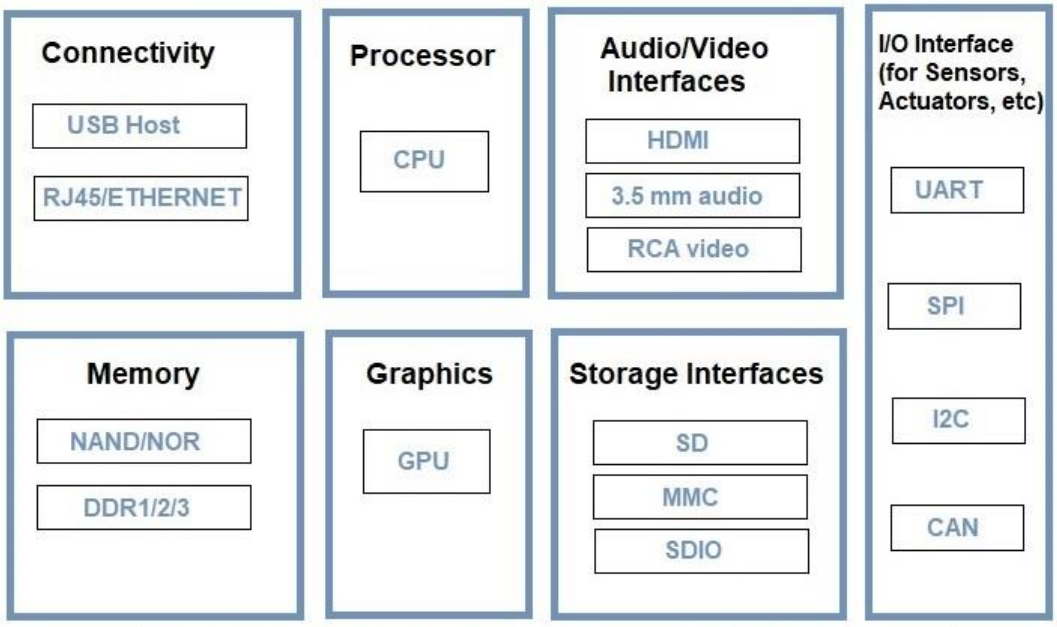
Physical and Logical Design of IoT. Physical Design of IoT system refers to IoT Devices and IoT Protocols. Things are Node device which have unique identities and can perform remote sensing, actuating and monitoring capabilities. Communication established between things and cloud based server over the Internet by various IoT protocols. Logical design of IoT system refers to an abstract representation of the entities & processes without going into the low-level specifies of the implementation.

Physical Design of IoT

Physical Design of IoT refers to IoT Devices and IoT Protocols. Things are Node device which have unique identities and can perform remote sensing, actuating and monitoring capabilities. IoT Protocols helps Communication established between things and cloud based server over the Internet.

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.

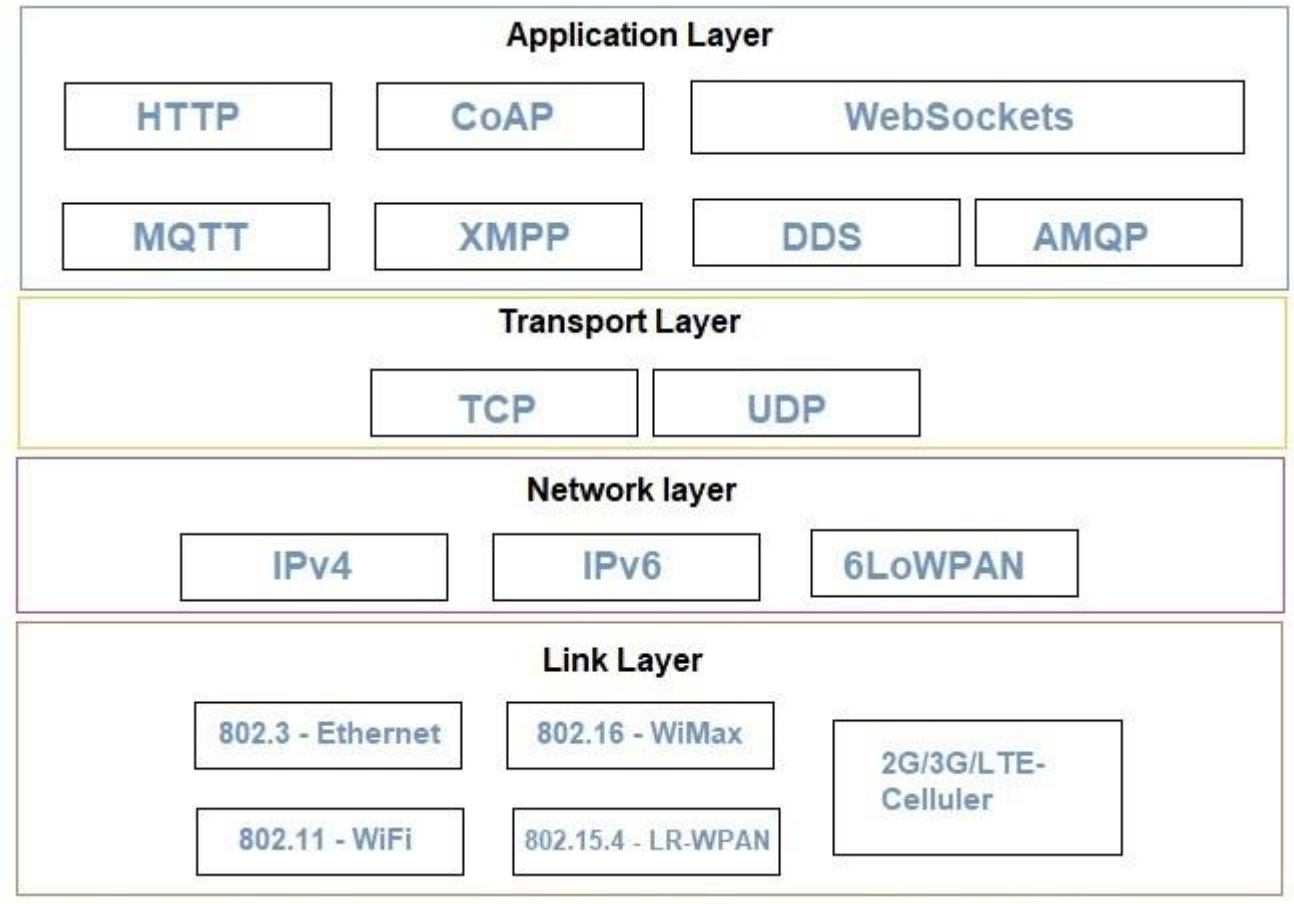
Above picture, shows a generic block diagram of IoT device. It may consist of several interfaces for connections to other devices. IoT Device has I/O interface for Sensors, Similarly for Internet connectivity, Storage and Audio/Video.

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 help to sent 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 (Coxial calbe or other or radio wave). Link Layer determines how the packets are coded and signaled by the hardware device over the medium to which the host is attached (eg. coxial 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 G

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. Initially 802.16a was developed and launched, but now it has been further refined. 802.16d or 802.16-2004 was released as a refined version of the 802.16a standard aimed at fixed applications. Another version of the standard, 802.16e or 802.16-2005 was also released and aimed at the roaming and mobile markets. Hz, and 60 GHz frequency bands.

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.

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 addrssing 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 the most recent version of the Internet Protocol (IP), the communications protocol that provides an identification and location system for computers on networks and routes traffic across the Internet. IPv6 was developed by the Internet Engineering Task Force (IETF) to deal with the long-anticipated problem of IPv4 address exhaustion. IPv6 is intended to replace IPv4. 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 2128, or approximately 3.4×1038 addresses.

6LoWPAN : 6LoWPAN 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. 6LoWPAN is a somewhat contorted acronym that combines the latest version of the Internet Protocol (IPv6) and Low-power Wireless Personal Area Networks (LoWPAN). 6LoWPAN, therefore, 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.

Application Layer

Application layer protocols define how the applications interface with the lower layer protocols to send over ther 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. Though often based on a TCP/IP layer, it can be used on any reliable transport layer, that is, a protocol that doesn't lose messages silently like UDP does. RUDP — the reliable update of UDP — is a suitable alternative.

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. It is defined as Contrained Application Protocol, and is a protocol intended to be used in very simple hardware. 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. It is a simplification of the HTTP protocol running on UDP, that helps save

bandwidth. It is designed for use between devices on the same constrained network (e.g., low-power, lossy networks), between devices and general nodes on the Internet, and between devices on different constrained networks both joined by an internet. CoAP is also being used via other mechanisms, such as SMS on mobile communication networks.

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. The protocol consists of an opening handshake followed by basic message framing, layered over TCP. The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers that does not rely on opening multiple HTTP connections (e.g., using XMLHttpRequest or <iframe>s and long polling).

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. For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with healthcare providers, and in a range of home automation and small device scenarios.

MQTT protocol runs on top of the TCP/IP networking stack. When clients connect and publish/subscribe, MQTT has different message

types that help with the handshaking of that process. The MQTT header is two bytes and first byte is constant. In the first byte, you specify the type of message being sent as well as the QoS level, retain, and DUP (duplication) flags. The second byte is the remaining length field.

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. Originally named Jabber, the protocol was developed by the eponymous open-source community in 1999 for near real-time instant messaging (IM), presence information, and contact list maintenance. Designed to be extensible, the protocol has been used also for publish-subscribe systems, signalling for VoIP, video, file transfer, gaming, the Internet of Things (IoT) applications such as the smart grid, and social networking services.

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.

In a distributed system, middleware is the software layer that lies between the operating system and applications. It enables the various components of a system to more easily communicate and share data. It simplifies the development of distributed systems by letting software developers focus on the specific purpose of their applications rather than the mechanics of passing information between applications and systems.

AMQP : The AMQP – 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. AMQP has the following three additives, which might link into processing chains in the server to create the favored capability.

Exchange: Receives messages from publisher primarily based programs and routes them to 'message queues'.

Message Queue: Stores messages until they may thoroughly process via the eating client software.

Binding: States the connection between the message queue and the change.

Logical Design of IoT

In this article we discuss 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 describes 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.

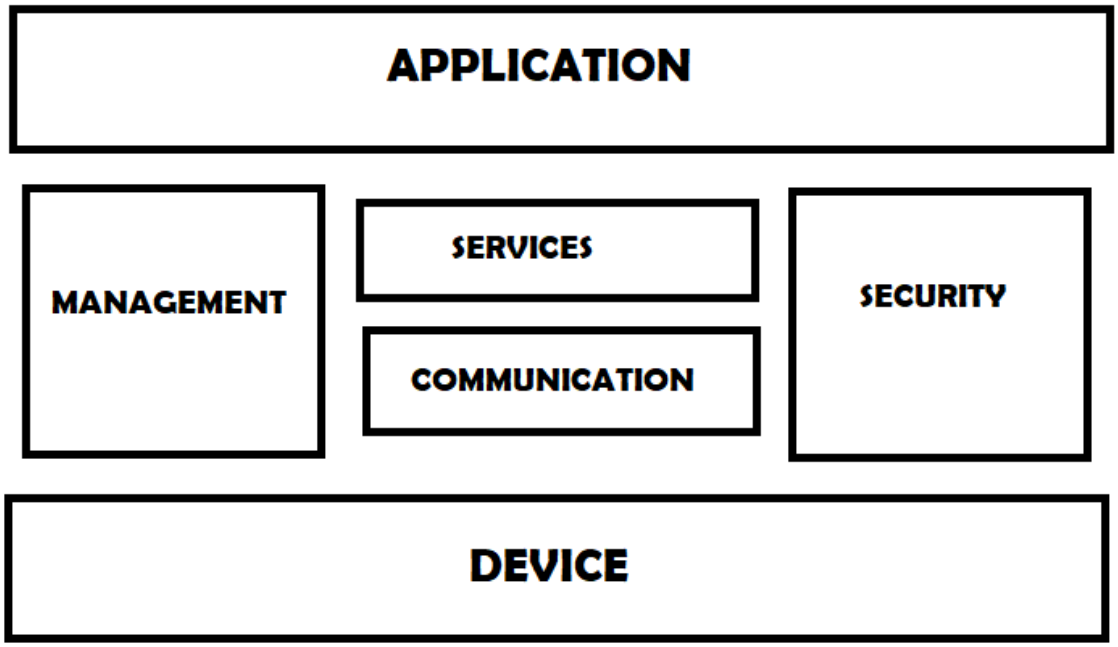
Communication: Handles the communication for the IoT system.

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.



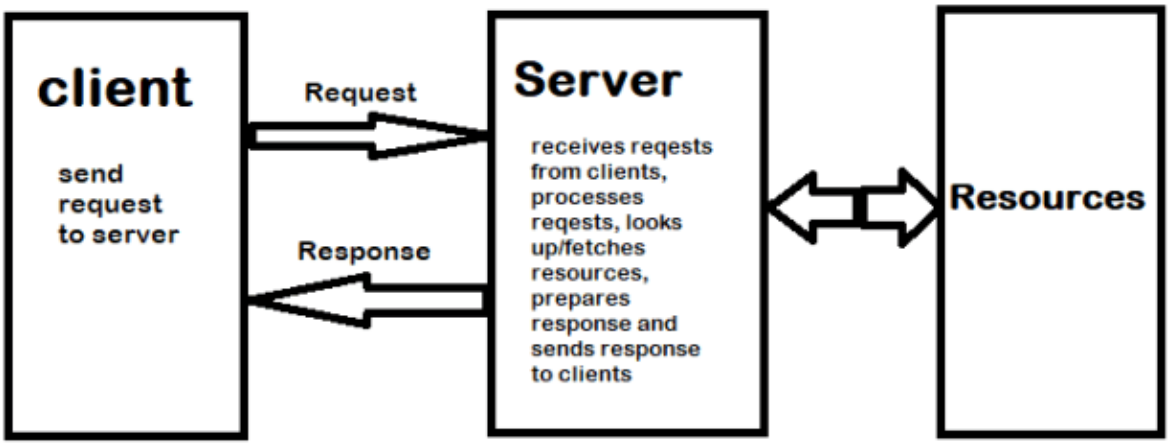
IoT Communication Models

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.

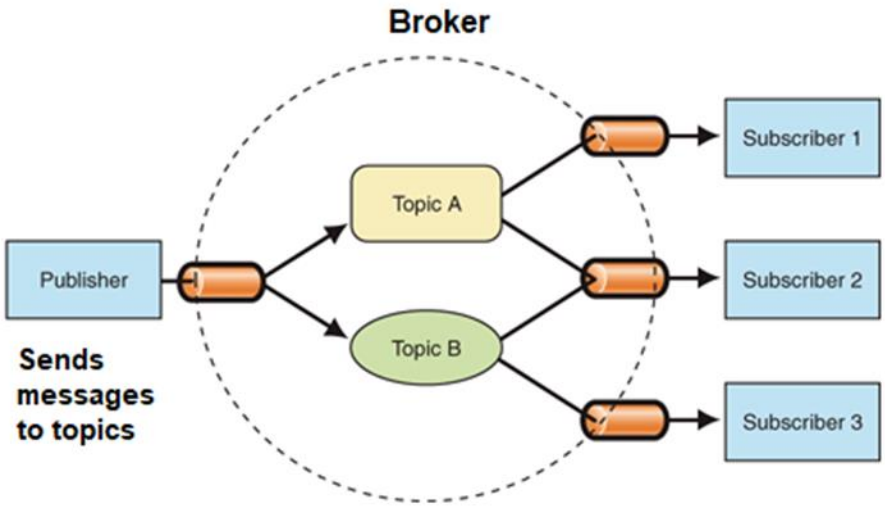


Request-Response Communication Model

Publish-Subscribe Model

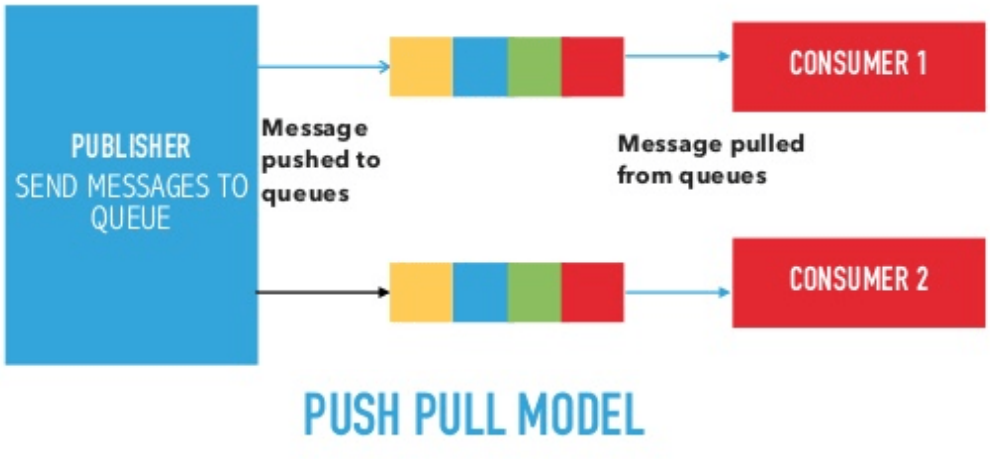
Publish-Subscribe is a communication model that involves publishers, brokers and consumers. Publishers are the source of data. Publishers send the data to the topics which are managed by the broker. Publishers are not aware of the consumers. Consumers subscribe to the topics which are managed by the broker.

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



Push-Pull Model

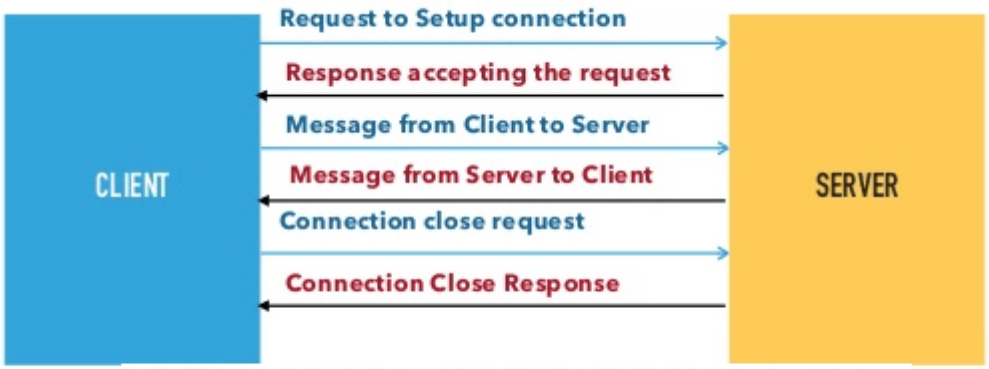
Push-Pull is a communication model in which the data producers push the data to queues and the consumers Pull the data from the Queues. Producers do not need to be aware of the consumers. Queues help in decoupling the messaging between the Producers and Consumers. Queues also act as a buffer which helps 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.



PUSH PULL MODEL

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.



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's resources and how resource states are addressed and transferred. REST APIs that follow the request response communication model, the rest architectural constraint apply 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 repsonse 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 intermediaryalong 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.

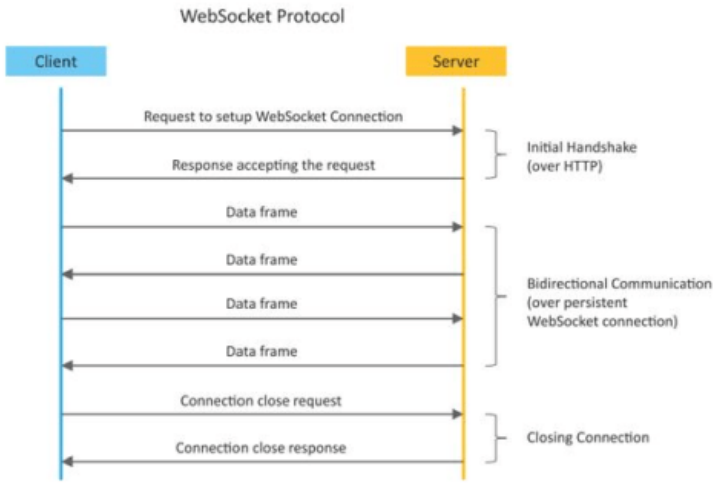
Uniform interface – uniform interface constraints requires that the method of communication between client and server must be uniform. Resources are identified in the requests (by URIsin web based systems) and are themselves is separate from the representations of the resources data returned to the client. 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.

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 coonection 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/mesages to each other in full duplex mode. Websocket API reduce the network traffic and letency 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.



Data Link Layer Communication Protocols in IoT

Several Communication Protocols are used in Internet of Things (IoT) to provide service to the network layer.

As we know 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.

Bluetooth :

Bluetooth is a PAN (Personal Area Network) or it is a short-range wireless communication network for exchanging data between the connected devices through that network. It is very cheaper in price and effective in a performance point of view for short-range distance. It is a 2.4GHz network that works well for personal wireless network communication. It provides a data transfer rate of 3 Mbps in a range of 50m to 150m. Nowadays Bluetooth is almost present in all smartphones and it is highly used in wearable devices connected with the mobile applications.

ZigBee :

Zigbee is similar to Bluetooth technology with 2.4Ghz frequency. It is a low power personal communication network. It is cheaper and is widely used for several applications. It is used for specific commercial and industrial applications. Its range varies from 10-100m. Mesh networking is one of the important advantages of Zigbee technology. Zigbee supports star or mesh network topology.

BLE (Bluetooth Low Energy) :

Bluetooth Low Energy is also known as Bluetooth smart which is a wireless PAN(Personal Area Network). The range is similar to that of Bluetooth but it consumes low power than Bluetooth. In 2011 BLE was introduced as Bluetooth 4.0. BLE goes to sleep mode when there is no transmission of data. It is a low-cost networking protocol. The smartphones operating systems like android, ios, etc uses this BLE technology and provide a Bluetooth network.

Wi-Fi (Wireless Fidelity) :

WiFi is a local area network which is a wireless network there is no wired connection. It is Proposed by Wi-Fi Alliance. WiFi provides Internet access to devices within a range of 60 feet to 100 feet. It uses high-frequency radio signals for sending and receiving data. It uses the IEEE 802.11 standard. Its data rate

varies from 2Mbps to 1.73Gbps. We can set up PAN (Personal Area Network) or LAN (Local Area Network) or WAN (Wide Area Network) in IoT systems. By Routing, we can increase the network area.

Z-Wave :

Z-wave technology is a wireless communication protocol that creates a wireless Mesh network. It is based on low power RF(Radio Frequency) based technology. It is mainly used for home automation applications and devices. It operates in 900 Mhz frequency bands. It is a more secure technology. It offers data transfer rates of 9.6Kbps, 40Kbps, or 100Kbps. Its range varies from 98 to 328 feet. It is low power and longer range IoT technology.

RFID (Radio-Frequency Identification) :

Radio Frequency Identification technology uses radiofrequency waves to transfer data between a reader and a movable item to identify and track. It does not require contact between reader and tagged item. RFID tag, RFID reader, RFID antenna are the key components of RFID technology. Tags operate depending upon their frequency bands of 13.56 and are mostly used.

Cellular :

Increased quantities of data can be sent over longer distances or range by using Cellular communications (GSM/3G/4G/5G etc). But it is very useful in sending a small quantity of data over the internet. Cellular carriers manage the infrastructure so when we use it we don’t need to worry about infrastructure costs and support costs.

Sigfox :

Sigfox is a form of wireless communications that provides low power and long-range wireless connectivity for devices. The messages are transmitted over the Sigfox global network. Sigfox provides one of the largest IoT networks. It is like

Cellular network type which sets up antennas on towers. It is a Low Power Wide Area Networks (LPWAN).

Ethernet:

Ethernet is used to connect the devices in a Local Area Network (LAN) which is based on IEEE 802.3 standard. Ethernet is a LAN technology in which the devices are wired connection which provides data transfer rates as high as 100 Mbps. Choosing Ethernet for IoT ecosystem is a little bit costly in terms of setup and management.

NFC (Near Field Communication) :

NFC is an IoT which helps to connect devices. It provides short-range wireless connectivity technology but NFC transmission is slower than Bluetooth. It is based on Radio Frequency Identity (RFID) technology. It can operate in low power. It operates at 13.56 MHz frequency.

LPWAN (Low Power Wide Area Network) :

LPWAN (Low Power Wide Area Network) is a wireless wide area network technology whose range varies from 2 km to 1000 km depending on the technology. Siggfox, LoRa is the examples involve all major LPWAN technology.

LoRaWAN :

LoRaWAN(Long Range Wide Area Network) is a wide area network protocol. It is a low power consumption protocol that targets the wide-area network (WAN) applications with better security and mobility. It supports a large network with millions and millions of low power devices deployed on public networks. It is along with range bidirectional communication which has a range of more than 15 km.

IoT Network Layer Protocols

The network layer is divided into two sublayers: routing layer which handles the transfer of packets from source to destination, and an encapsulation layer that forms the packets.

RPL Protocol

RPL stands for Routing Protocol for Low-Power and Lossy Network. It is a distance-vector protocol that supports a varity of Data Link Protocols. RPL builds a Destination Oriented Directed Acyclic Graph (DODAG) which has only one route from each leaf node to the root. All the traffic in this DODAG is routed through the root. Initially, each node sends a DODAG Information Object (DIO) announcing them self as a root. This information travels in the network, and complete DODAG is gradually built. When a new node wants to join the network, it sends a DODAG Information Solicitation (DIS) request and root responds back with a DAO Acknowledgment (DAO-ACK) confirming the join.

CORPL Protocol

CORPL protocol is the extension of the RPL protocol, which is termed as cognitive RPL. This network protocol is designed for cognitive networks and uses DODAG topology. CORPL protocol makes two new modifications in the RPL protocol. It uses opportunistic forwarding to forward a packet between the nodes. Each node of CORPL protocol keeps the information of forwarding set rather than parents only maintaining it. Each node updates its changes to its neighbor using DIO messages. On the basis of this updated message, each node frequently updates its neighbor for constant forwarder set.

CARP Protocol

CARP (Channel-Aware Routing Protocol) is a distributed routing protocol. It is designed for underwater communication. It has lightweight packets so that it can be used for Internet of Things (IoT). It performs two different functionalities: network initialization and data forwarding. CARP protocol does not support previously collected data. Hence, it is not beneficial for those IoT or other application where data is changed frequently. The upgradation of CARP is done in E-CARP which overcomes the limitation of CARP. The E-CARP allows the sink node to save previously received sensory data.

6LoWPAN

The 6LoWPAN protocol refers to IPv6 Low Power Personal Area Network which uses a lightweight IP-based communication to travel over low data rate networks. It has limited processing ability to transfer information wirelessly using

an internet protocol. So, it is mainly used for home and building automation. The 6LoWPAN protocol operates only within the 2.4 GHz frequency range with 250 kbps transfer rate. It has a maximum length of 128-bit header packets.

6LowPAN Security Measure

Security is a major issue for 6LowPAN communication Protocol. There are several attacks issues at the security level of 6LoWPAN which aim is to direct destruction of the network. Since it is the combination of two systems, so, there is a possibility of attack from two sides that targets all the layer of the 6LoWPAN stack (Physical layer, Data link layer, Adaptation layer, Network layer, Transport layer, Application layer).

Properties of 6LowPAN protocol

Standard: RFC6282

Frequency: Used over a variety of other networking media including Bluetooth Smart (2.4GHz) or ZigBee or low-power RF (sub-1GHz)

Range: NA

Data Rates: NA

Application layer refers to OSI Level 5, 6 and 7. It is application layer in the TCP-IP model. In IOT architecture, this layer lies above the service discovery layer. It is highest layer in the architecture extending from the client ends. It is the interface between the end devices and the network. This layer is implemented through a dedicated application at the device end. Like for a computer, application layer is implemented by the browser. It is the browser which implements application layer protocols like HTTP, HTTPS, SMTP and FTP. Same way, there are application layer protocols specified in context to IOT as well.

This layer is responsible for data formatting and presentation. The application layer in the Internet is typically based on HTTP protocol. However, HTTP is not suitable in resource constrained environment because it is extremely heavyweight and thus incurs a large parsing overhead. So, there are many alternate protocols

that have been developed for IOT environments. Some of the popular IOT application layer protocols are as follow –

- MQTT
- SMQTT
- CoAP
- DDS
- XMPP
- AMQP
- RESTful HTTP
- MQTT-SN
- STOMP
- SMCP
- LLAP
- SSI
- LWM2M
- M3DA
- XMPP-IOT
- ONS 2.0
- SOAP
- Websocket
- Reactive Streams
- HTTP/2
- JavaScript IOT

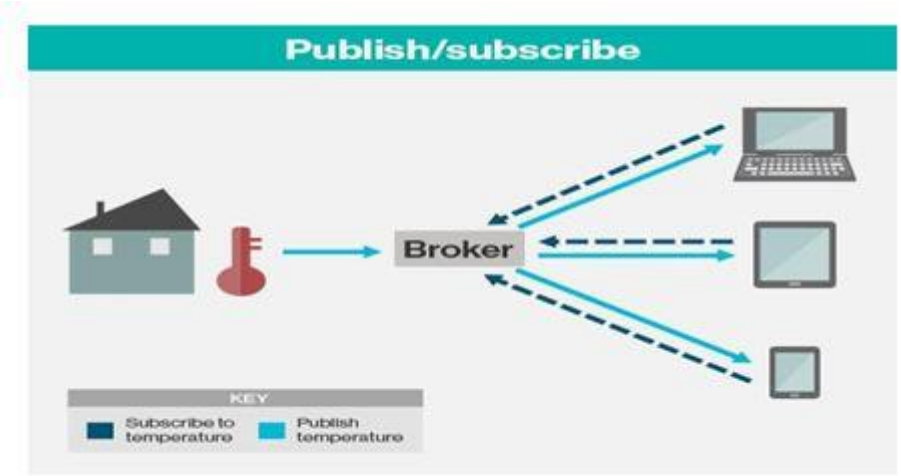
MQTT – Message Queuing Telemetry Transport is a lightweight messaging protocol. It uses publish-subscribe communication way and that's why it is used for M2M (machine to machine) communication. It is based on TCP-IP protocol and is designed to operate in limited bandwidth. In the protocol terminology, the

limited network bandwidth is referred as ‘small code footprint’. However, the exact meaning of limited network bandwidth is not clear in the specification.

This protocol has been specially designed for sensor networks and wireless sensor networks. MQTT allows devices to send or publish data information on a given topic to a server. There is a MQTT broker (Broker- Mosquitto) in between publisher and subscriber. The broker then transfers the information to the clients that are previously subscribed.

The sensors are interfaced with a broker which is an IOT device or server that reads and publish sensor data. The other devices that subscribe for and request sensor data are called clients. The sensors themselves are referred as publishers in the network. The client can be a laptop, smart phone, tablet or other mobile device. The client devices need to subscribe with the broker in the network to receive sensor data. For receiving data, the subscribed client devices have to establish connection with the broker and request data. The broker takes data from the publisher (wireless sensors) and send it to the client requesting for it. Even if the connection with the client device is broken after the request has been made, the broker saves the data in a cache so that when the client device reconnects with the broker, it could receive the requested sensor data. Similarly, if the connection between the publisher and the broker is broken after the request has been made, the broker forwards appropriate instructions sent by the publisher, so that client device can reconnect and receive requested data.

So, MQTT fares well even when connection between the broker and the publisher or broker and the client is broken due to limited network bandwidth. This ability to deal with delay or latency in network makes this protocol quite suitable for wireless networks.



An MQTT session can be divided into four stages –
1) Connection
2) Authentication
3) Communication
4) Termination

Connection and Authentication – In this stage, the client (like a mobile device or laptop) initiates a TCP-IP connection with the broker (server) using a standard port or a port defined by the network operator. The standard ports are generally 1883 for non-encrypted communication and 8883 for encrypted communication through SSL or TLS. In encrypted communication, the server sends server certificate to authenticate itself by the client and client may also send a certificate to the server to authenticate itself. Though even in encrypted communication, the client authentication is not part of the specification and also not common. Still, usually, the client authentication is done by passing a username and password by the client to the broker (server).

Communication – The sensor data is communicated to client using small code footprints. Though the term 'small code footprint' is not exactly specified in the protocol, it usually contains 2-byte long header, an optional variable length header, sensor data as message payload limited to 250 Mb in size and Quality of Service Level Indication. There can be three Quality of Service (QoS) levels – Unacknowledged Service (QoS Level 0), Acknowledged Service (QoS Level 1) and Assured Service (QoS Level 2). Each exceeding QoS level requires more network bandwidth and tolerance to latency.

There can be four types of operations that can be performed by a client – publish, subscribe, unsubscribe and ping. The client can subscribe for specific topics in the network, like a mobile device might subscribe with a broker to read current temperature of a city. For subscribing, the client needs to send a SUBSCRIBE/SUBACK packet pair to the broker. Similarly, for unsubscribing (from a topic), the client needs to send a UBSUBSCRIBE/UNSUBACK packet pair to the server. The ping operation can be performed by a client device to check the live connection with the broker. In the publish operation, the data is communicated between the broker and the client on a specific topic.

4) Termination – The publisher or client can terminate the connection by sending a DISCONNECT message to the broker, after which, the TCP-IP connection is closed. If the connection is broken suddenly (due to limited network bandwidth) without termination request, the broker sends the cached messages from the publisher to the client.

SMQTT – Based on MQTT, Secure Message Queue Telemetry Transport (SMQTT) protocol is an encryption based light weight messaging protocol. Compared to an MQTT session, SMQTT session has four stages – setup, encryption, publish and decryption. It has broker based architecture similar to MQTT, however, in this protocol both subscriber and publisher need to register with the broker using a secret master key. The data is also encrypted before being published by the publisher and then is decrypted at the subscriber end. There can be any encryption algorithm can be used by the developer.

CoAP – Constrained Application Protocol is specifically designed for constrained (limited) Hardware. The hardware that doesn't support HTTP or TCP/IP can use CoAP Protocol. So, basically the designers of this protocol taking inspiration by the HTTP had designed the CoAP protocol using UDP and IP protocol. It is a lightweight protocol that needs low power IOT application like for communication between battery powered IOT devices. Like HTTP, it also follows client-server model. The clients can GET, PUT, DELETE or POST informational resources over the network.

CoAP makes use of two message types – requests and responses. The messages contain a base header followed by message body whose length is specified by the datagram. The messages or data packets are small in size, so that they can be communicated among constraint devices without data losses. The messages can be confirmable or non-confirmable. For confirmable messages, the client need to respond with an acknowledgement after receiving the data packet. The request messages can contain query strings to implement additional functionalities like search, sort or paging.

For security, the protocol uses Datagram Transport Layer Security (DTLS) over UDP. In HTTP, it is very complicated to know the new state on a variable. in HTTP, client has to do polling again and again which means client has to ask every time every second to see if there is any new state of variable it is observing. In CoAP, the CoAP services try to solve the problem by creating an observe flag, so if the original device sends observe flag with GET command, every time the sever or the other devices see that there is a change in the state of the variable then the server or the devices will send the push notification to the original device who is actually finding the observer flag.

The protocol also provides an additional mechanism for resource discovery by the client devices. The server has to provide a list of resources along with meta data which can be accessed as link or application media type. By navigating through the list, a client can find available resources (information or data) and discover their media types.

The sensor nodes in the network themselves act as server instead of clients. The sensors are directly routable and the data communication is one to one between sensor and the client devices. For implementation of this protocols, the sensors must be able to receive data packets and able to respond them.

DDS – Designed by Object Management Group (OMG), Data Distribution Service is a M2M application layer protocol for real-time systems. Based on a publish – subscribe pattern like MQTT, the protocol architecture has nodes configured as publisher, subscriber or both. The protocol does not require any networking middleware, so the publishers can release information on specific topics (like a temperature sensor may release current temperature of a location) and the protocol itself manages to deliver it to the subscriber (like a mobile device showing current temperature of a location). The implementation of the protocol does not require any network programming as the protocol does not require verifying existence or location of the nodes and also does not need confirmation of the message delivery.

XMPP – Extensible Messaging and Presence Protocol (XMPP) is an XML based messaging protocol. XML is a mark up language for encoding documents which are both human readable as well as machine readable. XML evolved to extend HTML and allow addition of custom tags and elements to the web. As an extension of HTML, it allows structuring of data along with extensionality. Traditionally, XMPP has been used for real time communication like instant messaging, presence, multi-party chat, voice and video calls, collaboration, content syndication etc.

The use of XMPP for IOT allows real-time and scalable networking between devices or things. The things (devices) have one or more nodes and each node has several (informational) fields. Each field contains a readable and writable value. The nodes need to friend each other by sending and accepting friend requests. Once the friend request from one node is accepted by the other, it can receive updates from the other node. If other node also need to get updates from the first node, it also needs to send a friend request and need confirmation. If both nodes become mutually friend over the network, it is called dual subscription, otherwise, it is called single sided subscription. The data is communicated between the nodes on a one to one basis where a node can read or write field values in the other node.

AMQP – Like XMPP, Advanced Message Queue Protocol (AMQP) is also an open standard application layer protocol for message-oriented middleware. It is used for passing business messages between applications or organizations. It connects systems, feeds business processes with the information they need and reliably transmits the instructions that achieve their goals.

This is an interoperable and cross platform messaging standard. In the protocol architecture, the message along with a header is passed by the client to a broker or exchange. So, there is a single queue to which the message is passed by a producer. From the exchange or broker, the message can be passed on to one or many queues. The header contains information about each byte of the message. It also contains the routing information. The broker or exchange remains responsible to read headers and receive, route and deliver messages to the client applications. The communication in this protocol remains one to one between two nodes.

RESTful HTTP – Representational State Transfer (REST) or RESTful is a stateless and interoperable communication protocol. The protocol allows identifying web resources (informational resources) by unique URLs and let deliver them as HTTP, JSON or XML file. The resources are exposed as directory structure URIs. A client can access resources to GET, PUT, DELETE or POST messages to representations where representations are data objects and their attributes in HTML, XML or JSON format. The GET, PUT, DELETE and POST all are the HTTP request methods of receiving, modifying and sending data. Being a stateless communication, no acknowledgement is sent or received for confirmation of the message delivery. However, the HTTP request may be responded by a status code indicating success, redirection, informational, client error or server error.

MQTT-SN – MQTT-Sensor Network is an open and light-weight publish/subscription protocol designed specifically for constrained devices i.e. wireless sensor network (WSN). The wireless sensor network which does not have TCP/IP stack on the top of it like Zigbee based wireless sensor network or Bluetooth based WSN, they can send their data to the Internet with MQTT-SN protocol.

MQTT-SN is designed to be as close as possible to MQTT, but is adapted to the peculiarities of a wireless communication environment such as low bandwidth, high link failures, short message length, etc. It is also optimized for the implementation on low-cost, battery-operated devices with limited processing and storage resources.

Compared to MQTT, the following changes have been introduced in MQTT-SN –

- The topic names are replaced by topic IDs, which reduce the overheads of transmission.
- Topics do not need registration as they are preregistered.
- Messages are also split so that only the necessary information is sent.
- For power conservation, there is an offline procedure for clients who are in a sleep state. Messages can be buffered and later read by clients when they wake up.
- Clients connect to the broker through a gateway device, which resides within the sensor network and connects to the broker.

In the architecture of MQTT-SN, there are three kinds of MQTT-SN components – MQTT-SN clients, MQTT-SN gateways (GW), and MQTT-SN forwarders. MQTT-SN clients connect themselves to a MQTT server via a MQTT-SN GW using the MQTT-SN protocol. A MQTT-SN GW may or may not be integrated with a MQTT server. In case of a stand-alone GW, the MQTT protocol is used between the MQTT server and the MQTT-SN GW. Its main function is the translation between MQTT and MQTT-SN.

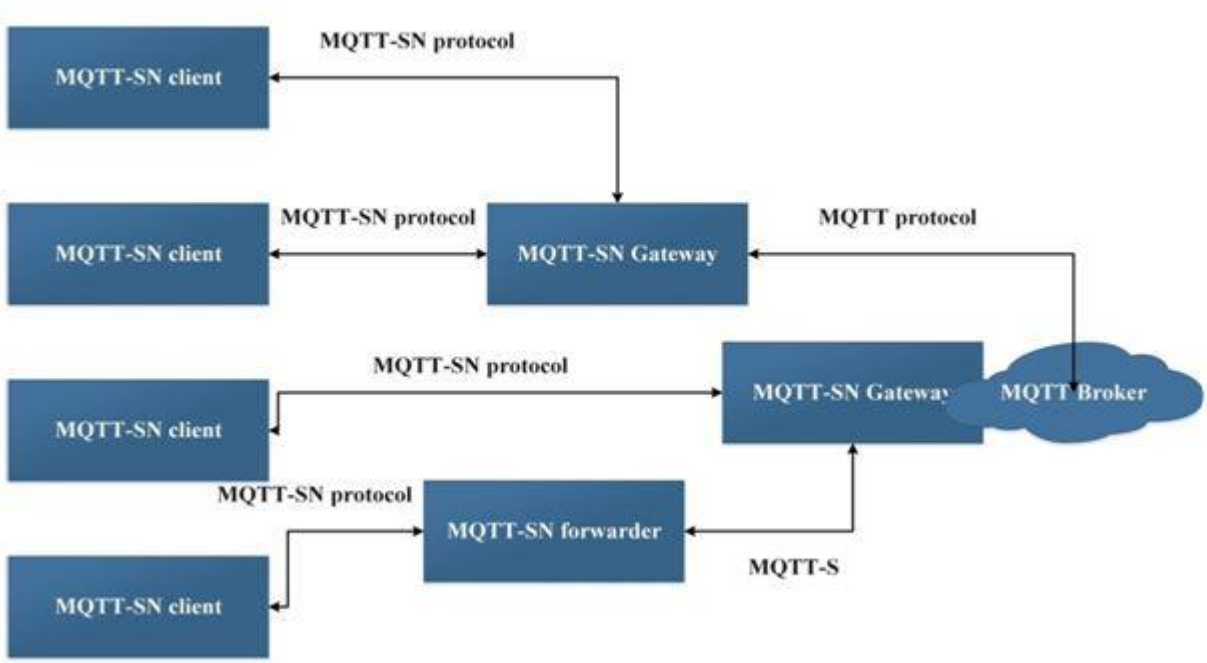


Fig. 2: Image showing Architecture of MQTT-SN Protocol

MQTT-SN clients can also access a GW via a forwarder in case the GW is not directly attached to their network. The forwarder simply encapsulates the MQTT-SN frames it receives on the wireless side and forwards them unchanged to the GW; in the opposite direction, it releases the frames it receives from the gateway and sends them to the clients, unchanged too.

STOMP – Simple or Streaming Text Orientated Messaging Protocol (STOMP) is a text based protocol for message oriented middleware. It is based on TCP and uses HTTP like commands. It has been designed to provide interoperability among platforms, languages and brokers. The data is communicated between a client and broker in multi-line frames containing command, header and content. The command can be CONNECT, DISCONNECT, ACK, NACK, SUBSCRIBE, UNSUBSCRIBE, SEND, BEGIN, COMMIT or ABORT.

SMCP – Simple Media Control Protocol (SMCP) is a CoAP protocol stack for embedded devices. It is developed in C language and can be used for sending and receiving asynchronous CoAP responses.

LLAP – Lightweight Local Automation Protocol is a simple and short messaging protocol which can run on any communication medium. the protocol has been designed for direct messaging between embedded devices independent of the low level physical layer protocol used by the devices.

SSI – Simple Sensor Interface (SSI) is an application layer protocol designed for communication between computers and sensors. Based on UART, the protocol allows polling sensors and streaming sensor data. The data is communicated by the sensors to the computers in small code footprints. In the protocol, the message between sensor and the computer contains a 2-byte header and the message payload. The header contains a single byte address and a single byte command or message type.

LWM2M – Lightweight M2M (LWM2M) is a device management protocol for sensor networks. Open Mobile Alliance (OMA) has developed this protocol to manage lightweight and low power devices on a variety of networks. The protocol stack is based on REST and CoAP.

M3DA – M3DA is a protocol stack for communication of binary data between M2M servers and applications running on embedded gateways. It is designed for device and asset management so that the network bandwidth is best used in optimum manner.

XMPP-IOT – Like XMPP is used for interoperable communication, XMPP-IOT is protocol stack for machine independent M2M communication.

ONS 2.0 – Object Name Service (ONS) is a protocol stack for locating metadata and services by a given GS1 identification key. GS1 is an electronic business messaging standard for automation of business transactions in a supply chain.

SOAP – Simple Object Access Protocol (SOAP) is an XML based protocol stack used for implementation of web services over internet. It uses HTTP request methods for messaging between clients over a network.

Websocket – WebSocket JavaScript interface defines a full-duplex single socket connection over which messages can be sent between client and server. The standard simplifies the complexity around bi-directional web communication and connection management.

Reactive Streams – Reactive Streams is a protocol stack for asynchronous stream processing with non-blocking back pressure. It is a specification which may be implemented on Java or JavaScript. It is used for handling live data streams whose volume is not known.

HTTP/2 – HTTP version 2 will be the next version of the HTTP protocol. It will be using header field compression to reduce latency and will allow multiple concurrent exchanges on the same connection.

JavaScript IOT – This is not any standard or protocol. It refers to the use of JavaScript and particularly Node.js (a server side JavaScript) with IOT boards for various applications. Some of the JavaScript based IOT projects include NodeMCU, Jerryscript, Socket.IO, Ruff, NodeRed, KinomaJS, CHIRIMEN, Mosca, Nodebots, heimcontrol.js, Resin, UPM, i2C, node-http2, onoff, node-serialport, mdns, IoT.js, Favor, Serverless, noduino, duino, Pijs.io, etc.

What Are The 6 Levels Of IoT?

Level Description

Sounding level: The layer is integrated with existing IoT hardware (RFID, sensors, actuators, etc.) in order to recognize/control the physical world and collect relevant info.

Network layer: The layer provides basic network support and information transmission over a wireless or wired network.

Service level: Services are created and managed at this level.

Interface level: The layer provides interaction between users and with third-party applications.

Scalability: How many IoT hardware devices are supported?

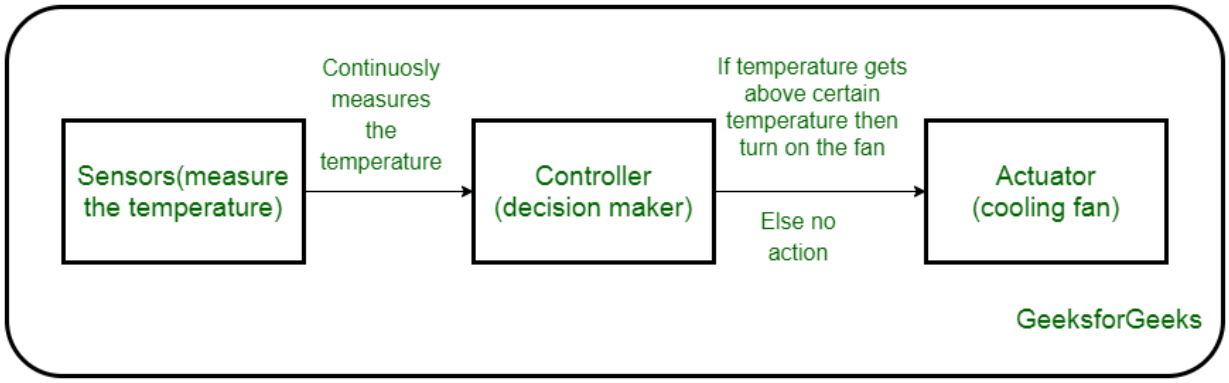
Cloud-Based Edge Computing: multilayer software solution like Arduino Uno

Features	IoT	M2M

Abbreviation	IoT stands for the Internet of Things.	M2M stands for Machine-to-Machine communication.
Intelligence	Devices include objects that are responsible for decision-making processes.	In M2M, there is a limited amount of intelligence observed.
Communication Protocol Used	IoT has used internet protocols like FTP, Telnet, and HTTP.	Communication technology and Traditional protocols are uses in M2M technology.
Connection Type Used	The connection of IoT is through the network and using various types of communication.	M2M uses a point to point connection.
Scope	It has a wide range of devices, yet the scope is large.	It has a limited Scope for devices.
Business Type Used	It has Business to Consumer (B2C) and Business to Business (B2B).	It has Business to Business (B2B) communication.
Data Sharing	In IoT, data sharing depends on the Internet protocol network.	In M2M, devices may be connected through mobile or any other network.
Open API Support	IoT technology supports Open API integrations.	In M2M technology, there is no Open API support.
Example	Big Data, Cloud, Smart wearables, and many more.	Data and Information, sensors, and many more.

Actuators in IOT

An IoT device is made up of a Physical object ("thing") + Controller ("brain") + Sensors + Actuators + Networks (Internet). An actuator is a machine component or system that moves or controls the mechanism or the system. Sensors in the device sense the environment, then control signals are generated for the actuators according to the actions needed to perform.



The control system acts upon an environment through the actuator. It requires a source of energy and a control signal. When it receives a control signal, it converts the source of energy to a mechanical operation. On this basis, on which form of energy it uses, it has different types given below.

Types of Actuators :

1. Hydraulic Actuators –

A hydraulic actuator uses hydraulic power to perform a mechanical operation. They are actuated by a cylinder or fluid motor. The mechanical motion is converted to rotary, linear, or oscillatory motion, according to the need of the IoT device. Ex- construction equipment uses hydraulic actuators because hydraulic actuators can generate a large amount of force.

Advantages :

- Hydraulic actuators can produce a large magnitude of force and high speed.
- Used in welding, clamping, etc.
- Used for lowering or raising the vehicles in car transport carriers.

Disadvantages :

- Hydraulic fluid leaks can cause efficiency loss and issues of cleaning.
- It is expensive.
- It requires noise reduction equipment, heat exchangers, and high maintenance systems.

2. Pneumatic Actuators –

A pneumatic actuator uses energy formed by vacuum or compressed air at high pressure to convert into either linear or rotary motion. Example- Used in robotics, use sensors that work like human fingers by using compressed air.

Advantages :

- They are a low-cost option and are used at extreme temperatures where using air is a safer option than chemicals.
- They need low maintenance, are durable, and have a long operational life.
- It is very quick in starting and stopping the motion.

Disadvantages :

- Loss of pressure can make it less efficient.
- The air compressor should be running continuously.
- Air can be polluted, and it needs maintenance.

3. Electrical Actuators –

An electric actuator uses electrical energy, is usually actuated by a motor that converts electrical energy into mechanical torque. An example of an electric actuator is a solenoid based electric bell.

Advantages :

- It has many applications in various industries as it can automate industrial valves.
- It produces less noise and is safe to use since there are no fluid leakages.
- It can be re-programmed and it provides the highest control precision positioning.

Disadvantages :

- It is expensive.
- It depends a lot on environmental conditions.

Other actuators are –

- **Thermal/Magnetic Actuators –**
  These are actuated by thermal or mechanical energy. Shape Memory Alloys (SMAs) or Magnetic Shape-Memory Alloys (MSMAs) are used by these actuators. An example of a thermal/magnetic actuator can be a piezo motor using SMA.
- **Mechanical Actuators –**
  A mechanical actuator executes movement by converting rotary motion into linear motion. It involves pulleys, chains, gears, rails, and other devices to operate. Example – A crankshaft.
- Soft Actuators
- Shape Memory Polymers
- Light Activated Polymers
- With the expanding world of IoT, sensors and actuators will find more usage in commercial and domestic applications along with the pre-existing use in industry.

Sensors in IOT

1. Temperature Sensors

Temperature sensors measure the amount of heat energy in a source, allowing them to detect temperature changes and convert these changes to data. Machinery used in manufacturing often requires environmental and device temperatures to be at specific levels. Similarly, within agriculture, soil temperature is a key factor for crop growth.

2. Humidity Sensors

These types of sensors measure the amount of water vapor in the atmosphere of air or other gases. Humidity sensors are commonly found in heating, vents and air conditioning (HVAC) systems in both industrial and residential domains. They can be found in many other areas including hospitals, and meteorology stations to report and predict weather.

3. Pressure Sensors

A pressure sensor senses changes in gases and liquids. When the pressure changes, the sensor detects these changes, and communicates them to connected systems. Common use cases include leak testing which can be a result of decay. Pressure sensors are also useful in the manufacturing of water systems as it is easy to detect fluctuations or drops in pressure.

4. Proximity Sensors

Proximity sensors are used for non-contact detection of objects near the sensor. These types of sensors often emit electromagnetic fields or beams of radiation such as infrared. Proximity sensors have some interesting use cases. In retail, a proximity sensor can detect the motion between a customer and a product in which he or she is interested. The user can be notified of any discounts or special offers of products located near the sensor. Proximity sensors are also used in the parking lots of malls, stadiums and airports to indicate parking availability. They can also be used on the assembly lines of chemical, food and many other types of industries.

5. Level Sensors

Level sensors are used to detect the level of substances including liquids, powders and granular materials. Many industries including oil manufacturing, water treatment and beverage and food manufacturing factories use level sensors. Waste management systems provide a common use case as level sensors can detect the level of waste in a garbage can or dumpster.

6. Accelerometers

Accelerometers detect an object's acceleration i.e. the rate of change of the object's velocity with respect to time. Accelerometers can also detect changes to gravity. Use cases for accelerometers include smart pedometers and monitoring driving fleets. They can also be used as anti-theft protection alerting the system if an object that should be stationary is moved.

7. Gyroscope

Gyroscope sensors measure the angular rate or velocity, often defined as a measurement of speed and rotation around an axis. Use cases include automotive, such as car navigation and electronic stability control (anti-skid) systems. Additional use cases include motion sensing for video games, and camera-shake detection systems.

8. Gas Sensors

These types of sensors monitor and detect changes in air quality, including the presence of toxic, combustible or hazardous gasses. Industries using gas sensors include mining, oil and gas, chemical research andmanufacturing. A common consumer use case is the familiar carbon dioxide detectors used in many homes.

9. Infrared Sensors

These types of sensors sense characteristics in their surroundings by either emitting or detecting infrared radiation. They can also measure the heat emitted by objects. Infrared sensors are used in a variety of different IoT projects including healthcare as they simplify the monitoring of blood flow and blood pressure. Televisions use infrared sensors to interpret the signals sent from a remote control. Another interesting application is that of art historians using infrared sensors to see hidden layers in paintings to help determine whether a work of art is original or fake or has been altered by a restoration process.

10. Optical Sensors

Optical sensors convert rays of light into electrical signals. There are many applications and use cases for optical sensors. In the auto industry, vehicles use optical sensors to recognize signs, obstacles, and other things that a driver would notice when driving or parking. Optical sensors play a big role in the development of driverless cars. Optical sensors are very common in smart phones. For example, ambient light sensors can extend battery life. Optical sensors are also used in the biomedical field including breath analysis and heart-rate monitors.

MYTHINGS IoT Sensor

The MYTHINGS Smart Sensor is a self-contained, battery-powered multi-purpose IoT sensor that allows you to capture critical data points like acceleration, temperature, humidity, pressure and GPS. The smart sensor is integrated with the MYTHINGS Library – a hardware independent, small-footprint and power-optimized library of code, featuring the MIOTY (TS-UNB) low-power wide area network protocol.